XEROX XEROX

Alto II/Orbit/Dover Press file printer

Spruce version 7.0

File: pictureorganization.st

Creation date: 25-APR-78 15:41:26

Name: Weyer, Steve

3 total sheets = 2 pages, 1 copy.

```
Problems encountered:
Character code 36b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 36b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 36b not found in Font set 0, font 0.
... more problems not listed ...
```

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 15 April 1978 at 4:49:03 am.'
**


PictureOrganization insert: 'test' with: (Traveller new image: (255 255 255 255 255 255 255 255 255 255
** 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 ) hull: (255
** 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
**255 255 255 255 255 255 ) bitsPerLine: 16).
**

PictureOrganization insert: 'lady' with: (Traveller new image: (7 192 12 96 30 240 52 88 37 72 36 72 39
** 200 61 120 2 128 126 252 32 8 60 120 8 32 31 240 2 128 14 224 ) hull: (7 192 15 224 31 240 63 248 63
** 248 63 248 63 248 61 120 3 128 127 252 63 248 63 248 15 224 31 240 14 224 14 224 ) bitsPerLine: 16).
**
**

PictureOrganization insert: 'bigtruck' with: (Traveller new image: (255 255 131 255 128 0 131 255 128 0
** 131 255 128 0 131 255 128 0 131 255 128 0 131 255 128 0 131 255 128 0 255 255 128 0 7 255 128 0 7 25
**5 128 0 7 255 128 0 7 255 255 255 255 255 36 0 147 255 36 0 147 255 24 0 99 255 ) hull: (255 255 131
**255 255 255 131 255 255 255 131 255 255 255 131 255 255 255 131 255 255 255 131 255 255 255 131 255 2
**55 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 60 0 2
**43 255 60 0 243 255 24 0 99 255 ) bitsPerLine: 22).
**

PictureOrganization insert: 'man' with: (Traveller new image: (7 192 12 96 10 160 8 32 12 96 7 192 60 1
**20 96 12 192 6 136 34 216 54 113 28 18 144 34 136 66 132 124 124 ) hull: (7 192 15 224 15 224 15 224
**15 224 7 192 63 248 127 252 255 254 255 254 255 254 127 252 30 240 62 248 126 252 124 124 ) bitsPerLi
**ne: 16).
**

PictureOrganization insert: '7000' with: (Traveller new image: (0 248 0 136 255 255 128 169 188 81 255
**255 128 1 128 1 128 1 128 1 128 1 128 1 128 1 128 1 128 1 255 255 ) hull: (0 248 0 248 255 255 255 25
**5 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 ) b
**itsPerLine: 16).
**

PictureOrganization insert: 'binder' with: (Traveller new image: (0 0 0 0 21 80 63 251 32 14 32 8 63 24
**8 255 254 255 254 64 4 64 4 64 4 64 4 64 4 64 4 64 4 ) hull: (0 0 0 0 31 240 127 255 127 255 127 255
**127 252 255 254 255 254 127 252 127 252 127 252 127 252 127 252 127 252 127 252 ) bitsPerLine: 16).
**

PictureOrganization insert: 'papers' with: (Traveller new image: (255 248 63 255 128 14 63 255 128 11 1
**91 255 128 10 191 255 128 10 191 255 128 10 191 255 128 10 191 255 128 10 191 255 128 10 191 255 128
**10 191 255 128 10 191 255 128 10 191 255 128 10 191 255 255 250 191 255 63 254 191 255 15 255 191 255
** ) hull: (255 248 63 255 255 254 63 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 2
**55 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 255 25
**5 255 191 255 255 255 191 255 63 255 191 255 15 255 191 255 ) bitsPerLine: 18).
**

PictureOrganization insert: 'waxer' with: (Traveller new image: (7 192 12 96 8 32 8 47 12 105 7 201 60
**71 96 57 192 1 128 15 216 56 115 144 18 144 18 144 114 156 124 124 ) hull: (7 192 15 224 15 224 15 23
**9 15 239 7 207 63 199 127 255 255 255 255 255 255 248 127 240 30 240 30 240 126 252 124 124 ) bitsPer
**Line: 16).
**

PictureOrganization insert: 'washer' with: (Traveller new image: (7 192 15 248 8 32 8 47 12 102 7 207 6
**0 121 96 1 192 1 128 15 216 56 113 16 17 16 17 16 17 16 126 252 ) hull: (7 192 15 248 15 239 15 239 1
**5 230 7 207 63 255 127 255 255 255 255 255 255 248 127 240 31 240 31 240 127 252 126 252 ) bitsPerLin
**e: 16).
**

PictureOrganization insert: 'cashier' with: (Traveller new image: (7 192 7 192 60 120 44 104 44 104 44
**104 43 168 57 56 2 128 126 252 32 8 60 120 8 32 31 240 2 128 14 224 ) hull: (7 192 7 192 63 248 63 24
**8 63 248 63 248 59 184 57 56 3 128 127 252 63 248 63 248 15 224 31 240 14 224 14 224 ) bitsPerLine: 1
**6).
**

PictureOrganization insert: '3100' with: (Traveller new image: (0 0 0 0 0 0 0 0 0 0 0 0 56 0 63 128 96
**192 32 192 32 128 32 128 32 128 63 128 192 96 192 96 ) hull: (0 0 0 0 0 0 0 0 0 0 124 0 127 192 127 1
**92 127 192 127 192 127 192 127 192 127 192 127 192 255 224 224 224 ) bitsPerLine: 16).
**

PictureOrganization insert: 'mediumtruck' with: (Traveller new image: (0 0 63 255 0 0 63 255 255 248 63
** 255 128 8 63 255 128 8 63 255 128 8 63 255 128 15 255 255 128 0 127 255 128 0 127 255 128 0 127 255
**128 0 127 255 128 0 127 255 255 255 255 255 36 36 63 255 36 36 63 255 24 24 63 255 ) hull: (0 0 63 25
**5 0 0 63 255 255 248 63 255 255 248 63 255 255 248 63 255 255 248 63 255 255 255 255 255 255 255 255
**255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 60 60 63 255 60 6
**0 63 255 24 24 63 255 ) bitsPerLine: 16).
**

PictureOrganization insert: 'fileclerk' with: (Traveller new image: (0 56 0 68 0 68 0 68 0 56 0 16 126
**124 65 150 65 19 65 146 65 156 65 188 65 172 127 230 32 195 31 193 ) hull: (0 56 0 124 0 124 0 124 0
**56 0 16 126 124 127 150 127 19 127 146 127 156 127 188 127 172 127 230 63 195 31 193 ) bitsPerLine: 1
**6).
**

PictureOrganization insert: 'dryer' with: (Traveller new image: (30 0 63 192 33 0 33 0 18 63 12 97 51 1
**93 96 33 193 217 128 137 160 143 100 128 36 128 36 128 36 128 255 224 ) hull: (30 0 63 192 63 0 63 0
**30 63 12 127 63 255 127 255 255 223 255 143 255 143 127 128 63 128 63 128 255 224 255 224 ) bitsPerLi
**ne: 16).
**

PictureOrganization insert: '4500' with: (Traveller new image: (0 0 0 0 0 0 62 0 34 0 62 7 162 4 190
**127 226 192 62 64 34 64 62 64 34 64 34 64 34 127 254 ) hull: (0 0 0 0 0 127 0 127 0 127 15 255 15 255
** 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 ) bitsPerLine: 16).
**

PictureOrganization insert: 'smalltruck' with: (Traveller new image: (0 0 0 0 0 0 0 63 0 33 0 33 0 33 2
**55 225 128 1 128 1 128 1 128 1 255 255 36 72 36 72 24 48 ) hull: (0 0 0 0 0 0 0 63 0 63 0 63 0 63 255
** 255 255 255 255 255 255 255 255 255 255 255 60 120 60 120 24 48 ) bitsPerLine: 16).
**

PictureOrganization insert: 'default' with: (Traveller new image: (0 0 0 0 0 0 0 1 0 3 0 7 0 13 0 25 0
**49 255 225 255 255 192 3 192 3 192 3 255 255 255 255 ) hull: (0 0 0 0 0 0 0 1 0 3 0 7 0 15 0 31 0 63
**255 255 255 255 255 255 255 255 255 255 255 255 255 255 ) bitsPerLine: 16).
**
```

XEROX XEROX

EARS

Filename: simulationcategories.st

Creation Date: April 25, 1978 2:35 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.2n on 17 January 1976 at 3:07:39 pm.'⌋
                                                                "Examples"
Class new title: 'Examples'
    subclasso|: Simulation
    fields: ''
    declare: '';
    asFollows⌋

This class does nothing but tell its name

Access
title
    [⇑'Examples']
⌋
SystemOrganization classify: ↪Examples under: 'SimulationCategories'.⌋
SystemOrganization insert: 'Examples'.⌋
                                                                "Vehicles"
Class new title: 'Vehicles'
    subclassof: Simulation
    fields: ''
    declare: '';
    asFollows⌋

This class does nothing but tell its name

Access
title
    [⇑'Vehicles']
⌋
SystemOrganization classify: ↪Vehicles under: 'SimulationCategories'.⌋
SystemOrganization insert: 'Vehicles'.⌋
```

```
Class new title: 'Factory'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows⌋

This class does nothing but tell its name

Access
title
   [⇑'Factory']
⌋
SystemOrganization classify: ↪Factory under: 'SimulationCategories'.⌋
SystemOrganization insert: 'Factory'.⌋
```

```
Class new title: 'FieldService'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows⌋

This class does nothing but tell its name

Access
title
   [⇑'FieldService']
⌋
SystemOrganization classify: ↪FieldService under: 'SimulationCategories'.⌋
SystemOrganization insert: 'FieldService'.⌋
```

```
Class new title: 'Office'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows┘

This class does nothing but tell its name

Access
title
   [⇑'Office']
┘
SystemOrganization classify: ↪Office under: 'SimulationCategories'.┘
SystemOrganization insert: 'Office'.┘
```

XEROX XEROX

EARS

Filename: statistical-distributions.sim,

Creation Date: April 25, 1978 2:57 PM

Printed by: Weyer, Steve

XEROX XEROX

'From Smalltalk 5.2k on 27 March 1978 at 2:27:15 pm.'⌟

"ProbabilityDistribution"

```
Class new title: 'ProbabilityDistribution'
   subclassof: Object
   fields: ''
   declare: 'U ';
   asFollows⌟

General superclass for probability distributions

Initialization
classInit ["this is really done by Uniform"]

Aspects
error: s [user notify: [s⇒ [s] 'message not implemented']]
mean [self error: false]
variance [self error: false]

Probability Function
distribution: int [self error: false]
inverseDistribution: x [self error: false]
plot: int [
   "need file: 'graphchanges.st' "
   Projector new
      projecting: (int start asFloat⊙0.0 rect: int stop asFloat⊙1.0) of: nil
      onto: (100.0⊙100.0 rect: 300.0⊙300.0) of: nil
      reflect: 0,2 "reflect y";
   clean;
   graph: self message: ↪y: on: int]
probability: x [
   "x is Integer for discrete probability laws,
   x is Float for continuous probability laws"
   self error: false]
y: x ["simulate other functions" ⇑x ⊙ (self probability: x)]

Random Sampling
generateSample: n | i s [
   s ← Vector new: n.
   for⦂ i to: n do⦂ [s◦i ← self random].
   ⇑s]
random [
   "general random number generation method for any probability law:
   use a (0.0, 1.0) uniformly distributed random number as the
   value of th: law's distribution function, and solve for the inverse"

   ⇑self inverseDistribution: U random01]
⌟
SystemOrganization classify: ↪ProbabilityDistribution under: 'Statistical
Distributions'.⌟
ProbabilityDistribution classInit⌟
```

```
Class new title: 'ContinuousProbability'
   subclassof: ProbabilityDistribution
   fields: ''
   declare: '';
   asFollows⌋
```

*superclass for continuous probability laws,
e.g. Uniform, Normal, Exponential, Gamma*

Initialization

Aspects

Probability Function

```
distribution: int | t x1 x2 y1 y2 [
   "slow and dirty trapezoidal integration"
   t ← 0.0.
   int ← int asStream.
   x2 ← int next.
   y2 ← self prcbability: x2.
   while⦂ [x1 ← x2. x2 ← int next] do⦂ [
      y1 ← y2.
      y2 ← self probability: x2.
      t ← t + ((x2-x1)*(y2+y1))].
   ↑t*0.5]
```

Random Sampling

```
⌋
SystemOrganization classify: ↱ContinuousProbability under: 'Statistical
Distributions'.⌋
```

```
                                                  "DiscreteProbability
                                                  "

Class new title: 'DiscreteProbability'
   subclassof: ProbabilityDistribution
   fields: ''
   declare: '';
   asFollows⌋

superclass for discrete probability laws,
   e.g. Bernoulli, Binomial, Poisson, Geometric

Initialization

Aspects

Probability Function
distribution: int | i t [
   t ← 0.0.
   for⦂ i from: int do⦂ [t ← t + (self probability: i)].
   ⇑t]

Random Sampling
⌋
SystemOrganization classify: ⤴DiscreteProbability under: 'Statistical
Distributions'.⌋
```

```
Class new title: 'Bernoulli'
   subclassof: DiscreteProbability
   fields: '
      p  "0.0 ≤ p ≤ 1.0"
      q  "q ← 1.0 − p" '
   declare: '';
   asFollows⌋
```

An example of a numerical valued random phenomena obeying the Bernoulli probability law with parameter p is the outcome of a Bernoulli trial in which the probability of success is p, where we denote success and failure by 1 and 0, respectively

Initialization

```
p: p [
   p between: 0.0 and: 1.0⇒[q ← 1.0 − p]
   self error: 'p must be between 0.0 and 1.0']
```

Aspects

```
mean [⇑p]
variance [⇑p * q]
```

Probability Function

```
inverseDistribution: x [x ≤ p⇒ [⇑1] ⇑0]
probability: x [
   x = 1⇒ [⇑p];
      =0⇒ [⇑q]
   ⇑0.0]
```

Random Sampling

```
⌋
SystemOrganization classify: ↪Bernoulli under: 'Statistical Distributions'.⌋
```

"Binomial"

```
Class new title: 'Binomial'
   subclassof: Bernoulli
   fields: '
      n  "n is Integer, > 0"'
   declare: '';
   asFollows⌋
```

An important example of a numerical valued random phenomenon obeying the binomial probability law with parameters n and p is the number of successes in n independent repeated Bernoulli trials in which the probability of success at each trial is p

Initialization

```
p: p n: n [
   n < 1⇒ [self error: 'n must be Integer > 0']
   self p: p]
```

Aspects

```
mean [⇑super mean * n]
variance [⇑super variance * n]
```

Probability Function

```
probability: x [
   x between: 0 and: n⇒ [
      "binomial coefficient (N x)"
      ⇑((n asFloat ! x) / (x asFloat ! x)) * (p ipow: x) * (q ipow: n-x)]
   ⇑0.0]
```

Random Sampling

```
random | t i [
   "a surefire but slow method: sample a Bernoulli n times"
   t ← 0.
   for§ i to: n do§ [t ← t + super random].
   ⇑t
   "for large n, we can approximate a binomial with a normal with mean
np
   and standard dev. (np(1-p)) sqrt"]
⌋
SystemOrganization classify: ↪Binomial under: 'Statistical Distributions'.⌋
```

"Exponential"

```
Class new title: 'Exponential'
   subclassof: ContinuousProbability
   fields: '
      mu    "mu > 0.0" '
   declare: '';
   asFollows⌟
```

The waiting time to the first event (in a series of events which happens in accordance with a Poisson probability law at the rate of mu events per unit of time (or space)) obeys the exponential probability law with parameter mu

Initialization

```
mean: mu [mu ≤ 0.0⇒ [self error: 'mu must be > 0.0']]
```

Aspects

```
mean [⇑1.0 / mu]
variance [⇑1.0 / (mu*mu)]
```

Probability Function

```
distribution: int [
   int stop ≤ 0.0⇒ [⇑0.0]
   ⇑1.0 - (mu * int stop) neg exp -
      [int start > 0.0⇒ [self distribution: (0.0 to: int start)] 0.0]]
inverseDistribution: x [
   "see Knuth, Vol. 2, p. 114"
   ⇑x ln neg / mu]
probability: x [
   x > 0.0⇒ [⇑mu * (mu*x) neg exp]
   ⇑0.0]
```

Random Sampling

```
⌟
SystemOrganization classify: ↪Exponential under: 'Statistical
Distributions'.⌟
```

```
Class new title: 'Gamma'
   subclassof: Exponential
   fields: '
      r  "r is Integer > 0" '
   declare: '';
   asFollows⌋
```

see Exponential. The waiting time to the rth event in a series of events ...obeys a gamma probability law with parameter r and mu

Initialization

```
mean: mu n: r [
   r ≤ 0⇒ [self error: 'r must be > 0']
   self mean: mu]
```

Aspects

```
mean [⇑super mean * r]
variance [⇑super variance * r]
```

Probability Function

```
probability: x | t [
   x > 0.0⇒ [
      t ← mu * x.
      ⇑(mu / r gamma) * (t ipow: r-1) * t neg exp]
   ⇑0.0]
```

Random Sampling

```
⌋
SystemOrganization classify: ↪Gamma under: 'Statistical Distributions'.⌋
```

```
Class new title: 'Geometric'
   subclassof: Bernoulli
   fields: ''
   declare: '';
   asFollows⌋
```

An important examrle of a numerical valued random phenomenon obeying the geometric probability law with parameter p is the number of trials required to obtain the first success in a sequence of independent repeated Bernoulli trials in which the probability of success at each trial is p

Initialization

Aspects

```
mean [⇑1.0/p]
variance [⇑q / (p*p)]
```

Probability Function

```
inverseDistribution: x [
   "Knuth, Vol. 2, pp.116-117"
   ⇑(x ln / q ln) ceiling]
probability: x [
   x > 0⇒ [⇑p * (q ipow: x-1)]
   ⇑0.0]
```

Random Sampling

```
⌋
SystemOrganization classify: ↪Geometric under: 'Statistical
Distributions'.⌋
```

```
Class new title: 'Normal'
   subclassof: ContinuousProbability
   fields: '
      mu   "mu real"
      sigma       "sigma > 0.0" '
   declare: '';
   asFollows⌟

A Normal distribution

Initialization
mean: mu sdev: sigma [sigma ≤ 0.0⇒ [self error: 'sigma must be > 0.0']]

Aspects
mean [⇑mu]
variance [⇑sigma*sigma]

Probability Function
probability: x [
   ⇑(¯0.5 * ((x - mu / sigma) ipow: 2)) exp / (sigma * (Float◦↪twopi)
sqrt)]

Random Sampling
random | v1 v2 s u [
   "Polar method for normal deviates, Knuth vol. 2, p.104, 113"
   u ← Uniform new from: ¯1.0 to: 1.0.
   while⦂ ([
      v1 ← u random. v2 ← u random.
      s ← v1*v1 + (v2*v2)]) ≥ 1 do⦂ [].
   u ← (¯2.0 * s ln / s) sqrt.
   "(v1 * u), (v2 * u) are normally distributed with mean 0, sdev 1"
   ⇑mu + (sigma * v1*u)]
⌟
SystemOrganization classify: ↪Normal under: 'Statistical Distributions'.⌟
```

```
Class new title: 'Poisson'
   subclassof: DiscreteProbability
   fields: '
      mu   "mu > 0.0" '
   declare: '';
   asFollows┘
```

*The Poisson law describes the probability that exactly k events occur in a unit time interval, when the mean rate of occurrence per unit time is the parameter mu. For a time interval of t, the parameter is mu*t*

Initialization

```
mean: mu [
   "average number of events happening per unit interval"
   mu ≤ 0.0⇒ [self error: 'mu must be > 0.0']]
```

Aspects

```
mean [⇑mu]
variance [⇑mu]
```

Probability Function

```
probability: n  [
   "the probability that in a unit interval, n events will occur"
   n ≥ 0⇒ [
      ⇑(mu neg exp * (mu ipow: n))/ (n asFloat ! n)]
   ⇑0.0]
```

Random Sampling

```
random | p n q [
   p ← mu neg exp.
   n ← 0.
   q ← 1.0.
   while⦂ (q ← q*U random) ≥ p do⦂ [n ← n+1].
   ⇑n]
┘
SystemOrganization classify: ↪Poisson under: 'Statistical Distributions'.┘
```

```
Class new title: 'SampleSpace'
   subclassof: ProbabilityDistribution
   fields: 'data rgen'
   declare: '';
   asFollows⌋

sample description space of a random phenomenon

Initialization
data: data [rgen ← Uniform new from: 1 to: data length]

Aspects

Probability Function

Random Sampling
random [⇑data◦rgen random]
⌋
SystemOrganization classify: ↪SampleSpace under: 'Statistical
Distribut.ons'.⌋
```

```
Class new title: 'Uniform'
   subclassof: ContinuousProbability
   fields: 'start stop step length'
   declare: 'L13849 L27181 ';
   asFollows⌋
```

A Uniform distribution

Initialization

```
classInit [
   "two Large numbers used frequently in random01"
   L13849 ← 13849 asLarge.
   L27181 ← 27181 asLarge.
   "class variable in ProbabilityDistribution"
   U ← Uniform new from: 0.0 to: 1.0]
from: start to: stop [
   length ← stop - start.
   [length is: Integer⇒ [length ← 1.0 + length]].
   length ≤ 0.0⇒ [self error: 'illegal uniform interval']
   self randomInit: mem◦0430 "Alto clock"]
```

Aspects

```
mean [⇑0.5 ∗ (start+stop)]
variance [⇑(length∗length) / 12.0]
```

Probability Function

```
inverseDistribution: x [⇑start + (x ∗ length)]
probability: x [
   x between: start and: stop⇒ [⇑1.0 / length]
   ⇑0.0]
```

Random Sampling

```
random01 [
   "Lehmers linear congruential method, Knuth Vol. 2.
   modulus m=2↑16
   a=27181 odd, and 5 = a mod 8
   c=13849 odd, and c/m around 0.21132"

   "generate a 0.0 < random number < 1.0, uniformly distributed"
   step← (L13849 + (L27181 ∗ step asLarge)) asSmall.
   step=0100000⇒ [⇑self random01 "omit 0.0. try again"]
   ⇑(32768.0 + step asFloat) / 65536.0]
randomInit: step "Call with constant to get repeatable sequence"
⌋
SystemOrganization classify: ↪Uniform under: 'Stat:stical Distributions'.⌋
Uniform classInit⌋
```

EARS

Filename: inputschedule.st

Creation Date: April 25, 1978 2:57 PM

Printed by: Weyer, Steve

```
'From Smalltalk 5.2k on 24 March 1978 at 12:23:43 pm.'⌋
```

```
Class new title: 'Inputschedule'
   subclassof: Object
   fields: 'constant orderedSet distribution'
   declare: '';
   asFollows⌋
```

This class returns arrival times according to stored or computed distributions

```
Distribution Initialization
distribution: distribution
   [constant ← orderedSet ← false]
exponential: m "m > 0.0"
   [self distribution: (Exponential new mean: m)]
from: dist size: n
   ["example: ...from: (Poisson new mean: 0.5) size: 200"
   "get a fixed random sample from some distribution"
   self data: (dist generateSample: n)]
geometric: p "0.0 < p ≤ 1.0"
   [self distribution: (Geometric new p: p)]
normal: m sdev: s "¯1.0e4000 < m < 1.0e4000. s > 0.0"
   [self distribution: (No~~rmal new mean: m~~; sdev: s)]
                                        ((Normal new) mean: m; deviation: s))
poisson: m "m > 0.0"
   [self distribution: (Poisson new mean: m)]
uniform: a to: b
   [self distribution: (Uniform new from: a to: b)]
uniform: a to: b by: c
   [self distribution: (SampleSpace new data: (a to: b by: c))]

Other Initialization
constant: constant
   [orderedSet ← distribution ← false]
data: vec "Vector/Interval/Set... (anything indexable with ◦)"
   ["random sampling with replacement'
   self distribution: (SampleSpace new data: vec)]
orderedSet: vec "Vector/Interval/Set...(anything recognizing asStream and
next)"
   [constant ← distribution ← false.
   "sampling will be done without replacement"
   orderedSet ← vec asStream]

Aspects
set
   [orderedSet⇒ [⇑true] ⇑false]

Sampling
next | t
   [constant⇒ [⇑constant]
   orderedSet⇒ [⇑orderedSet next]
   distribution⇒
      [t ← distribution random.
      t is: Float⇒ [⇑t round]
      ⇑t]
```

```
    ⇑false]
⌋
SystemOrganization classify: ↪Inputschedule under: 'Simulator'.⌋
```

XEROX XEROX

EARS

Filename: floatchanges.st,

Creation Date: April 25, 1978 2:58 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.2k on 24 March 1978 at 9:50:35 am.'⌋


Float asFollows⌋


Initialization
classInit
    [pi ← 3.1415926536.
    halfpi ← pi/2.0.
    fourthpi ← pi/4.0.
    twopi ← 2.0*pi.
    degreesPerRadian ← 180.0/pi.
    ln2 ←0.69314718 "0.6931471805599453094172321214581765 7".
    sqrt2 ← 1.4142136 "1.4142135623730950488016887242096980 8"]

Math functions
! n | t [
    "number of ways in which one can draw a sample (without
    replacement) of size n from a set of size M (self).
    partial product: self*(self-1)*...*(self-n+1)
    factorial = self ! self"

    n > self⇒ [⇑0.0]
    t ← 1.0.
    for⦂ n from: (self-n+1 to: self by: 1.0) do⦂ [t ← t * n]
    ⇑t]
ceiling ["(least integer) min k, k≥self"
    ⇑([self = self ipart⇒ [self] self+1.0]) asInteger]
exp | a n1 x x2 P Q [
    "see Computer Approximations, pp. 96-104, p. 205 (EXPB 1065)"

    self abs > 9212.0 "1.0e4001 ln"⇒ [user notify: 'exp overflow']
    x ← self / ln2.
    (n1 ← Float new "2.0 ipow: x asInteger")
        instfield: 1 ← x asInteger * 2.
    [(x ← x fpart) ≥ 0.5⇒ [
        n1 ← n1 * sqrt2.
        x ← x - 0.5]].
    x2 ← x*x.
    "compute 2.0 power: x"
    P ← Q ← 0.0.
    "(0.25250428525576241933744e4 0.28875563776168927289e2) reverse copy"
    for⦂ a from: ↪(28.875564 2525.0429) do⦂ [
        P ← (P*x2) + a].
    "(0.72857336028361108885189e4 0.37502165422086660021 3e3 0.1e1) reverse
copy"
    for⦂ a from: ↪(1.0 375.02165 7285.7336) do⦂ [
        Q ← (Q*x2) + a].
    ⇑n1 * ((Q + (x*P))/(Q - (x*P)))]
floor ["(greatest integer) max k, k≤self" ⇑self asInteger]
gamma | t [
    "eventually make work for non-integer values.
    use Stirling's formula?"
    t ← self - 1.0.
    ⇑t ! t]
log: base [⇑self ln / base asFloat ln]
ln | a x x2 n P [
```

n ! x ⊂ n/m ⊂ M/r

```
"see Computer Approximations, pp. 105-111, p. 227 (LOGE 2663)"

self ≤ 0.0⇒ [user notify: 'ln not valid for ' + self asString]

"exponent"
n ← ln2 * (((self instfield: 1) / 2) asFloat - 0.5).
"mantissa between 0.5 and 1.0".
(x ← self + 0.0)
    instfield: 1 ← 0.
x ← x * sqrt2.
x ← (x - 1.0) / (x + 1.0).
x2 ← x*x.
P ← 0.0.
"↪(0.2000000000046727e1 0.6666666635059382 0.4000059794795
    0.28525381498 0.2376245609) reverse copy"
for§ a from: ↪(0.23762456 0.28525381 0.40000598 0.66666664 2.0) do§ [
    P ← (P*x2) + a].
⇑n + (x * P)]
⌟
```

E A R S

Filename: office.st

Creation Date: April 25, 1978 2:48 PM

Printed by: Weyer, Steve

'From Smalltalk 5.3b/xm on 15 April 1978 at 4:51:24 am.'⌟

"AddCopier"

Environment

```
Class new title: 'AddCopier'
    subclassof: Simulation
    fields: ''
    declare: '';
    asFollows⌟
```

This class has not yet been commented

Parts

```
arrivalSchedule
      "State the job, input schedule,
      and initial assignments (the
      names of stations the job will
      visit). For example"

    [ self use: Papers new
              startAt: 0
              schedule:
                 (Inputschedule new
                                  constant: 1.5)
              assignments: 'Copying'.]
layout | var workers stns
    ["Create the stations with workers."
      "Provide space for the station."
 workers←Set new default.
 stns←Set new default.
 var ← Copying init
          of: (100⊙100 rect: 200⊙250).
 workers next←X4500 init
        id: 1 in: var.
 stns next ← var.
 var ← Branch init
          of: (300⊙100 rect: 400⊙250).        workers next←X7000 init
        id: 2 in: var.
 stns next ← var.
      "Get the workers and construct the station."
 for⦂ var from: stns do⦂
        [var use: workers.
        self constructStation: var].
 (workers asStream∘2)
        setTaskSchedule]
⌟
SystemOrganization classify: ↪AddCopier under: 'Office'.⌟
```

customer Arrivals

```
[ self schedule: Papers
         agenda: 'Copying'
         entersAt: 0
         nextEntersWithFrequency:
             (Inputschedule new
                 constant: 1.5). ]
```

places

```
[self newPlace: Copying
         name: 'copying'
         area: (100⊙100 rect: 200⊙250).
   self newPlace: Branch
         name: 'Branch'
         area: (300⊙100 rect: 400⊙250).
]
```

workers

```
[ self worker: X4500
         workingIn: 'Copying', 'Branch'
         startingAt: 'Copying'.
    self worker: X7000
         workingIn: 'Copying', 'Branch'
         startingAt: 'Branch' ]
```

paths

```
[ self pathFrom: 'Copying' to: 'Branch'
         takes: .
   self pathFrom: 'Branch' to: 'Copying'
         takes: (newPlace: 'Copying').
]
```

```
X7000
setTasks "what Tasks"
[ ↑'Copying' ]

setTaskSchedule "when to Do"
[ ↑ InputSchedule new constant: 8 ]
```

```
Class new title: 'Binder'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋

This class has not yet been commented

Parts
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."

    [ ⇑ 'binder']
serviceTime: job
    "The time the worker spends
     giving service to the job.  It is
     possible that job is a set of jobs.
     Time might a function of the
     job feature."

    [ ⇑ 0.5*(job feature∘2)]
⌋
SystemOrganization classify: ↪Binder under: 'Office'.⌋
```

```
Class new title: 'Binding'
    subclassof: Station
    fields: ''
    declare: '';
    asFollows┘
```

This class has not yet been commented

As yet unclassified

```
┘
SystemOrganization classify: ↪Binding under: 'Office'.┘
```

```
Class new title: 'Branch'
    subclassof: Station
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

As yet unclassified

```
⌋
SystemOrganization classify: ↪Branch under: 'Office'.⌋
```

```
Class new title: 'CopyFlow'
    subclassof: Simulation
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

```
Parts
arrivalSchedule
    "State the job, input schedule,
     and initial assignments (the
     names of stations the job will
     visit). For example"

    [ self use: Papers new
                startAt: 0
                schedule:
                    (Inputschedule new
                        constant: 3)
                assignments: 'Copying'.]
layout | var workers
    ["Create the stations with workers."
     "Provide space for the station."
 var ← Copying init
            of: (100⊙100 rect: 200⊙250).                    "Get the workers."
 var with: (X3100 init id: 1).
     "Now construct the station."
 self constructStation: var.]
⌋
SystemOrganization classify: ↪CopyFlow under: 'Office'.⌋
```

```
Class new title: 'Copying'
    subclassof: Station
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
    "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"

    [ ]
⌋
SystemOrganization classify: ↪Copying under: 'Office'.⌋
```

"FourCopiers"

```
Class new title: 'FourCopiers'
    subclassof: Simulation
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Parts

```
arrivalSchedule
     "State the job, input schedule,
     and initia assignments (the
     names of stations the job will
     visit). For example"

    [ self use: Papers new
            startAt: 0
            schedule:
              (Inputschedule new
                         constant: 1.25)
            assignments:
    (Inputschedule new data:
    'Area1', 'Area2', 'Area3', 'Area4').]
layout | var workers
 [ var ← Copying init
           name: 'Area1'
        of: (200⊙20 rect: 300⊙170).              var with: (X3100 init id: 1).
 self constructStation: var.
 var ← Copying init
           name: 'Area2'
        of: (340⊙20 rect: 440⊙170).              var with: (X3100 init id: 2).
 self constructStation: var.
 var ← Copying init
           name: 'Area3'
        of: (200⊙210 rect: 300⊙360).                 var with: (X3100 init id: 3).
 self constructStation: var.
var ← Copying init
           name: 'Area4'
        of: (340⊙210 rect: 440⊙360).                 var with: (X3100 init id: 4).
 self constructStation: var]
⌋
SystemOrganization classify: ⤴FourCopiers under: 'Office'.⌋
```

```
Class new title: 'Papers'
    subclassof: Job
    fields: ''
    declare: '';
    asFollows⌟

This class has not yet been commented

Parts
picture
    "The name of the picture for the
    job. The default is a square
    shape."

    [ ⇑ 'papers']
setFeature
    "Each Papers will be initialized with a unique pair of numbers: the
number of pages (average: 4) in it, and the number of copies (average: 5)
to be made of it."

    [ ⇑ (Inputschedule new
            uniform: 1 to: 30)next,
    (Inputschedule new
            normal: 5 sdev: 2)next abs.]
speed
    "The number of display bits per
    travel time"

    [ ⇑ 8]
travelTime
    "The amount of time it takes the
    job to travel one display bit if it
    travel speed=1."

    [ ⇑ 0]
⌟
SystemOrganization classify: ☞Papers under: 'Office'.⌟
```

"X3100"

```
Class new title: 'X3100'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Parts

```
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."

    [ ⇑ '3100']
serviceTime: job
"The time the copier spends on a single job. Job-specific information is
combined with known setup time (0.3 min. for entire job, plus 0.13 min for
each page of original) and machine performance rate (0.05 min per sheet of
output)."
    [⇑(0.05*(job feature∘1)+0.13) *
        (job feature∘2) +
        0.3]
speed
    "Number of display bits per travel
     time"

    [ ⇑ 8]
travelTime
   "The amount of time it takes the
    worker to travel one display bit if
    its travel speed=1."

    [ ⇑ 1]
⌋
SystemOrganization classify: ↪X3100 under: 'Office'.⌋
```

```
Class new title: 'X4500'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Parts

```
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."

    [ ⇑ '4500']
serviceTime: job
"The time the copier spends on a single job. Job-specific information is
combined with known setup time (0.3 min. for entire job, plus 0.13 min for
each page of original) and machine performance rate (0.022 min per sheet
of output)."
    [⇑(0.022*(job feature∘1)+0.13) *
        (job feature∘2) +
        0.3]
speed
    "Number of display bits per travel
     time"

    [ ⇑ 8]
travelTime
    "The amount of time it takes the
     worker to travel one display bit if
     its travel speed=1."

    [ ⇑ 1]
⌋
SystemOrganization classify: ↪X4500 under: 'Office'.⌋
```

```
Class new title: 'X7000'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows_|
```

This class has not yet been commented

Parts

```
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."

    [ ⇑ '7000']
serviceTime: job
"The time the copier spends on a single job. Job-specific information is
combined with known setup time (0.3 min. for entire job, plus 0.13 min for
each page of original) and machine performance rate (0.017 min per sheet
of output)."
    [⇑(0.017*(job feature◦1)+0.13) *
        (job feature◦2) +
        0.3]
setTaskSchedule  |n
     "Initialize the worker's travelling
      schedule. This example has the
      worker move every hour. "

    [ n← (station layout) whichStation: 'Copying'.
     (station layout) schedule: self
             todo: ⤴travelto:, n
             after: 8.]
speed
     "Number of display bits per travel
     time"

    [ ⇑ 16]
taskSchedule | task
     "It is possible that the worker can
     service more than one station.  At
     that time, the worker should be sent
     the message setTaskSchedule in order
     to initialize the schedule.  Example:
     worker changes stations each hour.
     Ordered list of stations is given here!"
        [ [tasksToDo empty ⇒
             [self setTasks: 'Copying']].
        task ← self nextTask.
        (station layout) schedule: self
                     todo: ⤴(travelto:, task)
                     after: 0.
        self setTaskSchedule. ]
travelTime
    "The amount of time it takes the
     worker to travel one display bit if
     its travel speed=1."
```

```
        [ ⇑ 0.5]
⌋
SystemOrganization classify: ↪X7000 under: 'Office'.⌋
SystemOrganization classify: ↪AddCopier alsounder: 'Simulation'⌋
SystemOrganization classify: ↪CopyFlow alsounder: 'Simulation'⌋
SystemOrganization classify: ↪FourCopiers alsounder: 'Simulation'⌋
SystemOrganization classify: ↪X3100 alsounder: 'Worker'⌋
SystemOrganization classify: ↪X4500 alsounder: 'Worker'⌋
SystemOrganization classify: ↪X7000 alsounder: 'Worker'⌋
SystemOrganization classify: ↪Binder alsounder: 'Worker'⌋
SystemOrganization classify: ↪Copying alsounder: 'Station'⌋
SystemOrganization classify: ↪Binding alsounder: 'Station'⌋
SystemOrganization classify: ↪Branch alsounder: 'Station'⌋
SystemOrganization classify: ↪Papers alsounder: 'Job'⌋
```

EARS

Filename: systemchanges.st.

Creation Date: May 4, 1978 3:04 PM

Printed by: Weyer, Steve

```
Class new title: 'ScrollBar'
   subclassof: Object
   fields: 'rect bitstr owner position'
   declare: 'JumpCursor DownCursor UpCursor ';
   asFollows⌋
```

I am a bar to the left of an awake window. With the cursor in me I can make that window scroll.

Initialization

```
classInit        "ScrollBar classInit"
   [UpCursor ← Cursor new fromtext: '
0000000110000000
0000001111000000
0000011111100000
0000111111110000
0001111111111000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000'.
   DownCursor ← Cursor new fromtext: '
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0001111111111000
0000111111110000
0000011111100000
0000001111000000
0000000110000000'.
   JumpCursor ← Cursor new fromtext: '
0000001000000000
0110001100000000
1111111110000000
0110001100000000
0000001000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000']
on: f from: o
    [self on: f from: o at: o scrollPos]
on: frame from: o at: f | w
    [owner ← o asCitation.
    rect ← Rectangle new
        origin: frame origin-((w←24)⊙2)
        extent: w⊙(frame height+4).
    position ← Rectangle new origin: 0⊙0 extent: 8⊙6.
    self position: f]
```

Scheduling

```
close
    [owner←nil]
eachtime | p h t oldCursor
    [(rect has: user mp)≡false⇒[⇑false]
    t← rect minus: (rect inset: 0⊙(h← rect width min: rect height/4)).
    oldCursor← Cursor new frompage1.
    whiles⦂ [rect has: (p← user mp)] do⦂
        [t◦1 has: p⇒
            [DownCursor topage1. whiles⦂ user redbug do⦂
                [self repositions⦂ [owner value scrollUp: ¯40]]]
        t◦2 has: p⇒
            [UpCursor topage1. whiles⦂ use· redbug do⦂
                [self repositions⦂ [owner value scrollUp: 40]]]
        jumpCursor topage1. whiles⦂ user redbug do⦂
                [self repositions⦂ [owner value scrollTo: ((0.0 max:
                    (user mp y-h-rect minY-4) asFloat/(rect height-12-(2*h)))
                    min: 1.0)]]].
    oldCursor topage1]

firsttime
    [⇑rect has: user mp]
lasttime
```

Image

```
hide        "restore background"
    [bitstr≡nil⇒ [user notify: 'Attemp: to hide unshown scrollbar']
    rect bitsFromString: bitstr.  bitstr← nil]
hidewhiles⦂ expr  | v
    [self hide. v ← expr eval. self show. ⇑v]
position: f | t
    [t← rect width min: rect height/4.
    position moveto: rect origin+
            ((rect width-position width/2)⊙(3+t+(f*((rect height-(2*t))-12))))]
repositions⦂ expr
    [self reshows⦂
        [expr eval. self position: owner value scrollPos]]
reshows⦂ expr | r
    [r ← position inset: ¯1.  expr eval.
    r clear: white.  position outline: 1]
```

```
show | r        "Save background and show"
    [bitstr ← rect bitsIntoString.
    rect clear: black.
    (r← rect inset: 2⊙2 and: 1⊙2) clear: ltgray.
    (r inset: 0⊙(rect width min: rect height/4)) outline: 1.
    position outline: 1]

Reclamation
keepCitations
    [owner keep]
⌋
SystemOrganization classify: ↪ScrollBar under: 'Panes and Menus'.⌋
ScrollBar classInit⌋
```

```
'from Smalltalk 5.2j on 9 January 1976 at 10:49:29 pm.'⌋


CodePane asFollows⌋

As stream
append: text        "append text (a string or paragraph) to my end; don't show"
   [pared append: text]
display      "scroll to the end of the paragraph and show it"
   [pared display]
⌋

CodeWindow asFollows⌋

As stream
append: text
   [(panes∘1) append: text]
cr
   [(panes∘1) append: 015 inString]
display
   [(panes∘1) display]
next ← char
   [(panes∘1) append: char inString]
print: x
   [(panes∘1) append: x asString]
space
   [(panes∘1) append: ' ']
⌋

ParagraphEditor asFollows⌋

Public Messages
append: text  "append text (a string or paragraph) to my end; don't show"
   [self fintype.
    [oldpara⇒ [] oldpara ← oldpara copy].
    para ← para replace: para length+1 to: para length by: text]
display  "scroll to the end of the paragraph and show it"
   [self fintype; select: para length+1; show]
⌋

ParagraphEditor asFollows⌋

Public Messages
selectAndScroll | l dY t
   [l ← self lineheight. self select.
   dY ← p1 y - window origin y.
    [dY≥0⇒ [dY ← (p1 y +l-1) - (window corner y) max: 0]].
   dY≠0⇒[self scrollby: (dY abs+l-1)/l*dY sign]
   ]
⌋

Point asFollows⌋

Arithmetic
+ delta
```

```
   ["Return a Point that is the sum of me and delta (which is a Point or
Number)"
   ⇑Point new x: (x + delta asPtX) y: (y + delta asPtY)
   ]
- delta
   ["Return a Point that is the difference of me and delta (which is a Point
or Number)"
   ⇑Point new x: (x - delta asPtX) y: (y - delta asPtY)
   ]
⌋

SystemOrganizer asFollows⌋

Mapping
classesIn: cat | c v                "⇑Vector of all classes in this category (-ies)"
   [cat is: String⇒
      [⇑(self category: cat) transforms⦂ c to⦂ Smalltalk◦c]
   v← Vector new: 0.
   for⦂ c from: cat do⦂
      [v← v+(self classesIn: c)]
   ⇑v]
whoIn: cat sends: message | x t
   [⇑((self classesIn: cat) transforms⦂ x to⦂
      [(t← x whosends: message) length=0⇒[nil]
      '

'+x title+' '+t]) notNil]
"
SystemOrganization whoIn: ↪('Forms' 'Simulator' 'Car Wash' 'Playground')
sends: ↪hide
"
⌋

ClassOrganizer asFollows⌋

Access to parts
classify: selector alsounder: heading | s
   [selector is: Vector⇒
      [for⦂ s from: selector do⦂
         [self classify: s alsounder: heading]]
   s ← commentVector find: heading.
   s>0 and⦂ (groupVector◦s has: selector)⇒[⇑self]
   [s=0⇒ [s ← self insert: heading]].
   groupVector◦s ← groupVector◦s insertSorted: selector.
   ]
⌋

UserView asFollows⌋

Special windows
notify: errorString | notifyWindow s
   ["Create a notify window looking at the Context stack"
   Top currentPriority≠1⇒
      [notifyWindow ← self notifier: errorString stack: thisContext sender
interrupt: false.
      notifyWindow⇒
         [thisContext sender ← nil.
```

```
            self scheduleOnBottom: notifyWindow.
            Top errorReset]
        ⇑nil]
    s ← thisContext sender.
    thisContext sender ← nil.
    (user sched∘1) showError: errorString stack: s interrupt: false]
⌋

Object asFollows⌋

System Primitives
showError: errorString stack: context interrupt: flag | notifyWindow
    [notifyWindow ← user notifier: errorString stack: context interrupt: flag.
     notifyWindow ⇒ [user restartup: notifyWindow]]
⌋

PriorityScheduler asFollows⌋

Top level
init11     " Top terminate: 11; init11. "
    [GRODSK ← Top
        install⦂
            [user displayoffwhile⦂
                [dp0 growSmalltalkBy: 100].
            GRODSK deepsleep]
        at: 11.
    GRODSK enable]
⌋
```

```
ParagraphEditor asFollows⌋

Public Messages
again | t
    [ [self fintype⇒[Scrap← Scrap text]].
    t← para findString: Deletion startingAt: loc2.
    t=0⇒[frame flash]
    loc1← t.  loc2← loc1+Deletion length.
    self paste]

Private Messages:
fintype
    [typein⇒
        [   [typein<loc1⇒
                [Scrap ← para copy: typein to: loc1-1.
                loc1 ← typein]].
        typein ← false]
    ⇑false]

Public Messages
paste [self fintype; replace: Scrap; selectAndScroll]

Private Messages
replace: t
    [   [oldpara⇒[] oldpara ← para copy].
        [typein≡false⇒[Deletion ← self selection]].
    para ← para replace: loc1 to: loc2-1 by: t.
    loc2 ← loc1 + t length.
    self show]

Public Messages
scrollby: n | l w
    [n ← n max: (frame origin y-4-window origin y)/(l← self lineheight).
    n ← n*l.
    frame moveby: 0⊙(0-n).
    w ← [n<0⇒[window inset: 0⊙0 and: 0⊙(0-n)]
            window inset: 0⊙n and: 0⊙0].
    w empty⇒[self show; select]
    w blt: w origin-(0⊙n) mode: storing.
        [n<0⇒[w corner y ← w origin y - n]
            w origin y ← w corner y - n].
    self showin: w; selectin: w]
scrollPos | t
    [t← (self ptofchar: para length+1) y - frame minY.
    t=0⇒[⇑0.0]
    ⇑(window minY-frame minY) asFloat / t]
scrollTo: f
    [self scrollUp:
            (f*(self lineheight + (self ptofchar: para length+1) y - frame minY))
asInteger -
            (window minY+4-frame minY)]
typing | more char
    [[loc1<loc2⇒[self checklooks⇒[self show. ⇑self select]]].
    more ← Stream default.
    [typein⇒[] Deletion← self selection. typein ← loc1].
    while⦂ user kbck do⦂
        [(char ← user kbd)
        =bs⇒ [more empty⇒[loc1 ← 1 max: loc1-1. typein ← typein min: loc1]
                more skip: ¯1];                                  "backspace"
        =ctlw⇒ [more reset. loc1 ← 1 max: loc1-1.                        "ctl-w for backspace
word"
                while⦂ [loc1>1⇒[(para◦(loc1-1)) tokenish] false] do⦂ [loc1 ← loc1-1]
                typein ← typein min: loc1];
        =cut⇒ [⇑self cut];
        =paste⇒ [⇑self paste];
        =esc⇒ [       [more empty⇒[] self replace: more contents. loc1 ← loc2].
                self fintype. "select previous type-in"
                loc1 ← loc2-Scrap length. ⇑self select]
        more next← char].
    self replace: more contents. loc1←loc2.
    user kbck⇒[] self selectAndScroll]
⌋
```

```
ListPane asFollows⌋

Private
scrollControl⦂ expr
        | dY onlyFirst butFirst onlyLast butLast x1 x2 y1 y2 y3 y4 k
    ["Selection is highlighted.  Unhighlight it.  Invalidate my saved para if I
scroll.  Then reselect selection, or deselect if it is no longer displayed."
    self compselection. dY ← self lineheight.
    x1 ← window origin x. x2 ← wincow corner x.
    y1 ← window origin y+2. y4 ← window height-4 |dY + y1. y2←y1+dY.
y3←y4-dY.
    onlyFirst ← x1+2⊙y1 rect: 2000⊙y2. butFirst ← x1⊙y2 rect: x2⊙y4.
    onlyLast ← x1+2⊙y3 rect: 2000⊙y4. butLast ← x1⊙y1 rect: x2⊙y3.
    while⦂ (k←expr eval)≠0 do⦂
        [k>0⇒[self scrollBy⦂ expr eval copying: butFirst into: butLast showing:
lastShown
                in: onlyLast direction: 1]
        self scrollBy⦂ expr eval copying: butLast into: butFirst showing:
firstShown
                in: onlyFirst direction: ¯1].
    self select: selection]

As yet unclassified
scrollPos
    [firstShown≡nil or⦂ list length=0⇒[⇑0.0]
    ⇑firstShown asFloat / list length]

Private
scrollUp: n
    [self scrollControl⦂
        [user buttons=4⇒
            [n sign*[mem∘0177036 nomask: 0100⇒[2] 1]]
        0]]
⌋

Textframe asFollows⌋

Image
scrollPos | t
    [t← (self ptofchar: para length+1) y - frame minY.
    t=0⇒[⇑0.0]
    ⇑(window minY-frame minY) asFloat / t]
⌋

ListPane asFollows⌋

Private
scrollTo: f | t
    [scl' scrollControl⦂
        [t← (f*list length) asInteger-firstShown.
        t<0⇒[firstShown≤0⇒[0] t]
        lastShown>list length⇒[0] t]]
⌋
```

```
Class asFollows⌋
printSubclassesOn: strm indent: n | x
    [strm cr. for⦂ x to: n do⦂ [strm tab].
    strm append: title.
    for⦂ x from: AllClassNames do⦂
        [(Smalltalk◦x) superclass≡self⇒
            [Smalltalk◦x printSubclassesOn: strm indent: n+1]]]
"
 | f. user displayoff while⦂
    [f← dp0 file: 'subclasses'.
    Object printSubclassesOn: f indent: 0.
    f shorten; close].
"
⌋

Dispframe asFollows⌋

Scheduler
hardcopy: p [text hardcopy: p]
⌋

ListPane asFollows⌋

As yet unclassified
pressSelect: p
    [selection=0⇒[]
    ColorPrint≡false⇒[]
    p hue: 40; saturation: 200.
    p showrect: self selectionRect color: 255]

Private
refresh | i s                "If para is nil, then recompute it"
    [para≡nil⇒
        [s ← (String new: 256) asStream.
        for⦂ i from: (firstShown to: lastShown) do⦂
            [[0< and⦂ i≤list length⇒ [(list◦i) printon: s] self dummy copyto: s].
            s cr].
        para ← s contents]]
⌋

PanedWindow asFollows⌋

Window protocol
hardcopy | p t
    [user displayoff while⦂
        [p ← dp0 pressfile: (t← (self title + '.press.') asFileName).
        self hardcopy: p.
        p close.
        user quitThen:
'Impress ' + t + '
Resume small.boot
']]
⌋
```

```
Paragraph asFollows⌋

Press printing
presson: strm in: r
   [⇑self presson: strm in: r style: DefaultTextStyle]
presson: strm in: r style: style                    "Output paragraph inside rectangle (page
coordinates)"
         | char s1 s2 s3 y chop
   [s3 ← ParagraphScanner new of: self to: strm style: style.
   s3 init in: r.  y ← r corner y.
   s1 ← s3 copy. s2 ← s3 copy.
   chop ← [alignment=1⇒ [0] alignment].
   while⦂ (char ← s3 scan) do⦂
      [char ≡ true ⇒                "Exceeded max width"
         [[s2 position = s1 position ⇒                    "No blanks in line"
            [y ← s3 printfrom: s1 align: 0 backup: 1. s3 backup. s3 copyto: s2]
          y ← s2 printfrom: s1 align: alignment backup: 1].
         y ⇒
            [s2 init in: (r origin rect: r corner x ⊙ y). s2 copyto: s1. s2 copyto:
s3]
         ⇑ self copy: s1 position + 1 to: text length]
      char = 040 ⇒
         [s3 copyto: s2. s3 space]
      char = 015 ⇒
         [y ← s3 printfrom: s1 align: chop backup: 1 ⇒
            [s3 init in: (r origin rect: r corner x ⊙ y). s3 copyto: s1. s3 copyto:
s2]
         ⇑ self copy: s1 position + 1 to: text length]
      char = 011 ⇒
         [s3 tab]
      "user notify: 'unimplemented control char'"]
   "Put out trailing text if any"
   s3 width=0⇒ [⇑y]
   y ← s3 printfrom: s1 align: chop backup: 0 ⇒ [⇑y]
   ⇑ self copy: s1 position + 1 to: text length]

⌋

ParagraphScanner asFollows⌋

Scanning
scan "Scan up to a zero-width character"
         | char w maxw cd len pos
   [maxw ← rect width asInteger.
   [textstrm end ⇒ []
    ascent ← ascent max: font ascent.
    descent ← descent min: font descent.
   ].
   while⦂ (char ← textstrm next) do⦂
      [w ← font widthof: char.
      w = 0 ⇒ [⇑char]
      width ← width + w.
      width > maxw ⇒ [⇑true]].
   while⦂ (len  ← runstrm next) do⦂
      [cd ← runstrm next.
      len = 0 ⇒ []                  "Go to next run"
      pos ← textstrm position.
```

```
        textstrm of: para text from: pos+1 to: pos+len.
        font ← press codefont: cd style: style.
        ⇑self scan].
    ⇑false]
⌋
```

```
PressFile asFollows⌋
```

Bitmap support

```
bitmap: rect bits: bits | r dlpos w w1 h len "w1 is for Swinehart"
    [boundbox ← boundbox max: (r ← self transrect: rect).
    dlpos ← file wordpos.
        [dlpos*2=file position⇒[]
        self skipchars: 1. file positioneven: 0. dlpos←dlpos+1].
    w ← rect width. h ← rect height.
    self setcodingdots: (w1 ← w+15|16) lines: h;
        setmode;
        setsizewidth: (scale * w1) height: (scale * h);
        setwindowwidth: w1 height: h;
        dotsfollow.
    [bits⇒[file append: bits]                    "bits supplied"
        rect bitsOntoStream: file].                  "else from screen"
    self setp: r origin;
        showdots: (file wordpos - dlpos)]
```

File structure commands

```
close | p i fp
    [   [EL empty⇒[] self closePage].
    "put out font directory, part directory, document directory"
    self fontdir.
    fp ← file pages.
    parts puton: file.
    file padpage.
    p ← file pages.
    file nextwordvector ← (27183, "press password"
        (p + 1) "number of records",
        (parts position / 8) "number of parts",
        fp, (p - fp), "part dir and length"
        ¯1, ¯1, ¯1, "junk"
        1, 1, "first and last copies"
        ('S'◦1 "solid color")).
    for⦂ i from: (10 to: 0177) do⦂ [file nextword← ¯1].
    file append: (file title asBCPL: 52);
        append: (dp0 username asBCPL: 32);
        append: (Date default asString+' '+Time default asString asBCPL: 40);
        padpage; shorten; close]
closePage | padding dllength st
    ["Assumes entity trailer has been put on"
    dllength ← file position - dlstart.
    file nextword ← 0.
    EL puton: file; reset.
    padding ← file padpage. st ← dlstart/512.
    parts nextwordvector ← (0, st, (file position/512 - st), padding).
    dlstart ← file position]
entity: box containing⦂ expr | fp1 fp2 EL1 v
    [fp1← file position. EL1← EL position/2.
    v← expr eval.
    EL positioneven: 255.               "word-pad EL with ‹Nop›"
    file positioneven: 0.               "word-pad DL with 0"
    fp2 ← file position - fp1.
    "Put a trailer into the EL"
    EL nextwordvector ← (0, "type and fontset"
        0, fp1 "dlstart relative to DL location in file", 0, fp2,
        800, 1200, "paper origin"
        (box origin x), (box origin y),
        (box width), (box height)).
    EL nextword ← EL position/2 - EL1 + 1.
    ⇑v]
```

Initialization

```
of: f
    [file ← f.  self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.
    fontcodes ← Vector new: 0.
    fontdefs ← Vector new: 0.
    dlstart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]
```

Bitmap support

```
screenout: rect scale: scale
    ["puts a bit map image onto the PressFile.  The standard
    scaling is 32 micas per  Alto dot.  22 looks better, Dover only
    works with 32"
    user displayoffwhile⦂
        [self somefont.
        self bitmap: rect bits: false.
        self page.
        self close]]
⌋
```

Textframe asFollows⌋

Printing

```
brightness [⇑255]
hardcopy | p
    [p ← dp0 pressfile: 'frame.press'.
    self hardcopy: p.
    p close]
hardcopy: p | e w
    [p entity: (p transrect: (w← window inset: ¯2)) containing⦂
        [   [ColorPrint⇒[p hue: self hue; saturation: self saturation.
                p showrect: window color: self brightness]].
        self pressSelect: p.
        for⦂ e from: (w minus: window) do⦂
            [p showrect: e color: 0].
        para asParagraph presson: p
            in: (p transrect: ((window inset: 0) width← frame width))
            style: style]]
hue [⇑120]
pressSelect: p              "ignored"
saturation [⇑191]
⌋
```

```
UserView asFollows⌋

Display
displayoffwhile⦂ expr | i t v
    [t ← mem◦067.
    for⦂ i from: t to: 58 by: ¯2 do⦂
        [mem◦067 ← i. 1/1/1/1/1 "slow"].
    v ← expr eval.
    for⦂ i from: 58 to: t by: 2 do⦂
        [mem◦067 ← i. 1/1/1/1/1 "slow"].
    ⇑v]
⌋
```

```
UserView asFollows⌋

Window Scheduling
restore | w
    ["screenrect clear. d sp outline.
    for: w from: (sched length to: 1 by: ¯1) do:
        [(sched◦w) show]"
     (sched◦1) refresh]
⌋

Window asFollows⌋

Framing
refresh
    [self show]
⌋

ScrollBar asFollows⌋

Image
hide        "restore background"
    [bitstr=nil⇒ []
    rect bitsFromString: bitstr.  bitstr← nil]
⌋

ScrollBar asFollows⌋

Scheduling
firsttime
    [⇑ (rect inset: ¯5⊙ ¯5) has: user mp]
⌋

ListPane asFollows⌋

Pane protocol
hardcopy: p [self refill. ⇑super hardcopy: p]
outline
    [window outline: 1]
⌋

ScrollBar asFollows⌋

Image
position: f | t
    [t← rect width min: rect height/4.
    position moveto: rect origin+
            ((rect width-6/2)⊙(3+t+(f*((rect height-(2*t))-12)))).
    ⇑f]
⌋

Textframe asFollows⌋
```

```
Image
outline
    [window border: 1 color: black]
⌋

WidthTable asFollows⌋

Initialization
lockup | key font file i
    [key ← name + pointsize asString + (↪('' 'I' 'B' 'BI')◦(face+1)).
    font ← WidthDict lookup: key⇒ [⇑font]
    file ← dp0 file: 'fonts.widths'.
    self fontfrom: file.
    file close.
    for⦂ i from: ↪(011 015 040) do⦂
        [i≥min and: i≤max ⇒ [widths◦(i-min+1) ← 0]].
    WidthDict insert: key with: self.
    user show: 'OK.'; cr.
    ⇑self]
⌋

SystemPane asFollows⌋

Window protocol
enter       "be sure I am up to date"
    [mySysOrgVersion≡user classNames⇒ [super enter]
    window outline. self update.
    super enter]
⌋
```

```
Class asFollows⌋
printSubclassesOn: strm indent: n | x
    [strm cr. for⦂ x to: n do⦂ [strm tab].
    strm append: title.
    for⦂ x from: AllClassNames do⦂
        [(Smalltalk∘x) superclass≡self⇒
            [Smalltalk∘x printSubclassesOn: strm indent: n+1]]]
"
 | f. user displayoffwhile⦂
    [f← dp0 file: 'subclasses'.
    Object printSubclassesOn: f indent: 0.
    f shorten; close].
"
⌋

Dispframe asFollows⌋

Scheduler
hardcopy: p [text hardcopy: p]
⌋

ListPane asFollows⌋

As yet unclassified
pressSelect: p
    [selection=0⇒[]
    ColorPrint≡false⇒[]
    p hue: 40; saturation: 200.
    p showrect: self selectionRect color: 255]

Private
refresh | i s              "If para is nil, then recompute it"
    [para≡nil⇒
        [s ← (String new: 256) asStream.
        for⦂ i from: (firstShown to: lastShown) do⦂
            [[0<. and⦂ i≤list length⇒ [(list∘i) printon: s] self dummy copyto: s].
            s cr].
        para ← s contents]]
⌋

PanedWindow asFollows⌋

Window protocol
hardcopy | p t
    [user displayoff while⦂
        [p ← dp0 pressfile: (t← (self title + '.press.') asFileName).
        self hardcopy: p.
        p close.
        user quitThen:
'Empress ' + t + '
Resume small.boot.
']]
⌋
```

```
Paragraph asFollows┘

Press printing
presson: strm in: r
    [⇑self presson: strm in: r style: DefaultTextStyle]
presson: strm in: r style: style                "Output paragraph inside rectangle (page
coordinates)"
        | char s1 s2 s3 y chop
    [s3 ← ParagraphScanner new of: self to: strm style: style.
    s3 init in: r.  y ← r corner y.
    s1 ← s3 copy. s2 ← s3 copy.
    chop ← [alignment=1⇒ [0] alignment].
    whiles⦂ (char ← s3 scan) do⦂
        [char ≡ true ⇒              "Exceeded max width"
            [[s2 position = s1 position ⇒                   "No blanks in line"
                [y ← s3 printfrom: s1 align: 0 backup: 1. s3 backup. s3 copyto: s2]
             y ← s2 printfrom: s1 align: alignment backup: 1].
            y ⇒
                [s2 init in: (r origin rect: r corner x ⊙ y). s2 copyto: s1. s2 copyto:
s3]
            ⇑ self copy: s1 position + 1 to: text length]
        char = 040 ⇒
            [s3 copyto: s2. s3 space]
        char = 015 ⇒
            [y ← s3 printfrom: s1 align: chop backup: 1 ⇒
                [s3 init in: (r origin rect: r corner x ⊙ y). s3 copyto: s1. s3 copyto:
s2]
            ⇑ self copy: s1 position + 1 to: text length]
        char = 011 ⇒
            [s3 tab]
        "user notify: 'unimplemented control char'"]
    "Put out trailing text if any"
    s3 width=0⇒ [⇑y]
    y ← s3 printfrom: s1 align: chop backup: 0 ⇒ [⇑y]
    ⇑ self copy: s1 position + 1 to: text length]

┘

ParagraphScanner asFollows┘

Scanning
scan "Scan up to a zero-width character"
        | char w maxw cd len pos
    [maxw ← rect width asInteger.
    [textstrm end ⇒ []
     ascent ← ascent max: font ascent.
     descent ← descent min: font descent.
    ].
    whiles⦂ (char ← textstrm next) do⦂
        [w ← font widthof: char.
        w = 0 ⇒ [⇑char]
        width ← width + w.
        width > maxw ⇒ [⇑true]].
    whiles⦂ (len  ← runstrm next) do⦂
        [cd ← runstrm next.
        len = 0 ⇒ []                "Go to next run"
        pos ← textstrm position.
        textstrm of: para text from: pos+1 to: pos+len.
        font ← press codefont: cd style: style.
        ⇑self scan].
    ⇑false]
⌋
```

PressFile asFollows⌋

Bitmap support

```
bitmap: rect bits: bits | r dlpos w w1 h len "w1 is for Swinehart"
    [boundbox ← boundbox max: (r ← self transrect: rect).
    dlpos ← file wordpos.
        [dlpos*2=file position⇒[]
        self skipchars: 1. file positioneven: 0. dlpos←dlpos+1].
    w ← rect width. h ← rect height.
    self setcodingdots: (w1 ← w+15|16) lines: h;
        setmode;
        setsizewidth: (scale * w1) height: (scale * h);
        setwindowwidth: w1 height: h;
        dotsfollow.
    [bits⇒[file append: bits]                    "bits supplied"
        rect bitsOntoStream: file].                      "else from screen"
    self setp: r origin;
        showdots: (file wordpos - dlpos)]
```

File structure commands

```
close | p i fp
    [   [EL empty⇒[] self closePage].
    "put out font directory, part directory, document directory"
    self fontdir.
    fp ← file pages.
    parts puton: file.
    file padpage.
    p ← file pages.
    file nextwordvector ← (27183, "press password"
        (p + 1) "number of records",
        (parts position / 8) "number of parts",
        fp, (p - fp), "part dir and length"
        ¯1, ¯1, ¯1, "junk"
        1, 1, "first and last copies"
        ('S'◦1 "solid color")).
    for⦂ i from: (10 to: 0177) do⦂ [file nextword← ¯1].
    file append: (file title asBCPL: 52);
        append: (dp0 username asBCPL: 32);
        append: (Date default asString+' '+Time default asString asBCPL: 40);
        padpage; shorten; close]
closePage | padding dllength st
    ["Assumes entity trailer has been put on"
    dllength ← file position - dlstart.
    file nextword ← 0.
    EL puton: file; reset.
    padding ← file padpage. st ← dlstart/512.
    parts nextwordvector ← (0, st, (file position/512 - st), padding).
    dlstart ← file position]
entity: box containing⦂ expr | fp1 fp2 EL1 v
    [fp1← file position. EL1← EL position/2.
    v← expr eval.
    EL positioneven: 255.             "word-pad EL with ‹Nop›"
    file positioneven: 0.             "word-pad DL with 0"
    fp2 ← file position - fp1.
    "Put a trailer into the EL"
    EL nextwordvector ← (0, "type and fontset"
        0, fp1 "dlstart relative to DL location in file", 0, fp2,
        800, 1200, "paper origin"
        (box origin x), (box origin y),
        (box width), (box height)).
    EL nextword ← EL position/2 - EL1 + 1.
    ⇑v]

Initialization
of: f
    [file ← f.  self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.
    fontcodes ← Vector new: 0.
    fontdefs ← Vector new: 0.
    dlstart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]

Bitmap support
screenout: rect scale: scale
    ["puts a bit map image onto the PressFile.  The standard
    scaling is 32 micas per Alto dot.  22 looks better, Dover only
    works with 32"
    user displayoffwhile⦂
        [self somefont.
        self bitmap: rect bits: false.
        self page.
        self close]]
⌋

Textframe asFollows⌋

Printing
brightness [⇑255]
hardcopy | p
    [p ← dp0 pressfile: 'frame.press'.
    self hardcopy: p.
    p close]
hardcopy: p | e w
    [p entity: (p transrect: (w← window inset: ¯2)) containing⦂
        [   [ColorPrint⇒[p hue: self hue; saturation: self saturation.
                p showrect: window color: self brightness]].
        self pressSelect: p.
        for⦂ e from: (w minus: window) do⦂
            [p showrect: e color: 0].
        para asParagraph presson: p
            in: (p transrect: ((window inset: 0) width← frame width))
            style: style]]
hue [⇑120]
pressSelect: p            "ignored"
saturation [⇑191]
⌋
```

```
UserView asFollows⌟

Display
displayoffwhile⦂ expr | i t v
    [t ← mem◦067.
  ·for⦂ i from: t to: 58 by: ¯2 do⦂
        [mem◦067 ← i. 1/1/1/1/1 "slow"].
    v ← expr eval.
    for⦂ i from: 58 to: t by: 2 do⦂
        [mem◦067 ← i. 1/1/1/1/1 "slow"].
    ⇑v]
⌟
```

```
PanedWindow asFollows⌋

Window protocol
hardcopy: p | pane
    [self showtitle.
    titleframe hardcopy: p.
    for⦂ pane from: panes do⦂ [pane hardcopy: p]]
⌋

PressFile asFollows⌋

Bitmap support
bitmap: rect bits: bits | r dlpos w w1 h len "w1 is for Swinehart"
    [boundbox ← boundbox max: (r ← self transrect: rect).
    dlpos ← file wordpos.
        [dlpos*2=file position⇒[]
        self skipchars: 1. file positioneven: 0. dlpos←dlpos+1].
    w ← rect width. h ← rect height.
    self setcodingdots: (w1 ← w+15|16) lines: h;
        setmode;
        setsizewidth: (scale * w1) height: (scale * h);
        setwindowwidth: w1 height: h;
        dotsfollow.
    [bits⇒[file append: bits]                    "bits supplied"
        rect bitsOntoStream: file].                      "else from screen"
    self setp: r origin;
        showdots: (file wordpos - dlpos)]

File structure commands
box: rect hue: hue sat: sat bright: bright containing⦂ expr | w r
    [self entity: (self transrect: (w← rect inset: ¯2)) containing⦂
        [for⦂ r from: (w minus: rect) do⦂
            [self showrect: r color: 0].
        [ColorPrint⇒
            [self hue: hue; saturation: sat;
                showrect: rect color: bright; brightness: 0]].
        expr eval]]
⌋

Textframe asFollows⌋

Printing
hardcopy: p | e w
    [p box: window hue: self hue sat: self saturation
        bright: self brightness containing⦂
        [self pressSelect: p.
        p brightness: 0.
        para asParagraph presson: p
            in: (p transrect:
                (((window intersect: frame) inset: 0⊙¯1) width← frame width))
            style: style]]
⌋

Window asFollows⌋
```

Framing

```
show
   [self outline. growing⇒[]
   self showtitle]
showtitle
   [titleframe put:
      (Paragraph new text: self title runs: titlerun alignment: 0)
      at: frame origin+titleloc.
   titleframe outline]
⌟
```

```
LargeInteger asFollows⌋

Conversion
asInteger | t i
    [bytes length>3⇒[⇑self]
    t ← 0.
    for⦂ i from: bytes length to: 1 by: ¯1 do⦂
        [t ← t*0200+(bytes◦i)].
    neg⇒[⇑0-t]
    ⇑t]
⌋

ListPane asFollows⌋

Private
fill | i        "Given firstShown, compute lastShown and show me."
    [lastShown ← firstShown-1 + (window extent y-4/self lineheight)
        min: list length+1.
    [self locked⇒
        [i ← (selection-lastShown max: 0) + (selection-firstShown min: 0).
        i≠0⇒ [para←nil. firstShown ← firstShown + i. lastShown ← lastShown
+ i]]].
    (frame ← window inset: 2) width ← 999.
    [para≡nil⇒[self refill]].
    self show]

Pane protocol
hardcopy: p
    [(frame ← window inset: 2) width ← 999.
    self refill. ⇑super hardcopy: p]

Private
refill | i s            "Recompute para from list"
    [s ← (String new: 256) asStream.
    for⦂ i from: (firstShown to: lastShown) do⦂
        [   [0<i and⦂ i≤list length⇒ [(list◦i) printon: s]
                self dummy copyto: s].
        s cr].
    para ← s contents asParagraph]
⌋

Paragraph asFollows⌋

Normal access
+ c         "Concatenates two paragraphs"
    [c← c asParagraph.
    ⇑Paragraph new
        text: text + c text
        runs: (self runcat: self runs to: c runs)
        alignment: alignment]

Manipulation of format runs
allBold [self allStyle: 1]
allStyle: val
```

```
    [runs← self makerun: text length val: val]
⌋

ParagraphEditor asFollows⌋

Public Messages
append: text  "append text (a string or paragraph) to my end; don't show"
    [self fintype.
        [oldpara⇒ [] oldpara ← oldpara copy].
     para ← para + text]
⌋

ParagraphScanner asFollows⌋

Printing
printfrom: ps align: align backup: n "Returns false if goes below bottom"
        | ybot a b p xs sp rs c len tpos ts
    [ybot ← rect corner y - (style lineheight*32).
    ybot < rect origin y ⇒ [⇑false]                         "Won't fit"
    a ← ps position + 1. b ← textstrm position - n.
    a > b ⇒ [⇑ybot]               "No text"
    p ← rect origin x ⊙ (rect corner y - (style maxascent*32)).
    xs ← rect width - width.
    sp ← font space.               "Kludge"
    ts ← tabpos contents asStream.
    [align
        = 2 ⇒ [press setp: xs/2⊙0+p; setspacex: sp];
        = 4 ⇒ [press setp: xs⊙0+p; setspacex: sp].
        press setp: p; setspacex: sp].
    press append: para text◦(a to: b).
    rs ← (para run: a to: b) asStream.
    tpos ← ts next.
    while⦂ (len ← rs next) do⦂
        [c ← rs next land: 0363.                     "Remove underline and strikeout"
        len = 0 ⇒ []
        press selectfont: (press fontindex: c style: style) - 1.
        b ← a+len.
        while⦂ (tpos and⦂ tpos<b) do⦂                       "Put out tabs"
            [press showchars: tpos-a;
                skipchars: 1;
                setx: p x + ts next.
            a ← tpos+1. tpos ← ts next].
        [align=1 and⦂ tpos=false ⇒                       "Reset space width"
            [press setspacex: xs/spaces+sp.
            align ← 0]].
        press showchars: b-a.
        a ← b].
    ⇑ybot]
⌋

PressFile asFollows⌋

Font handling stuff
fontindex: code style: style | ix font n
    ["return index if in font dictionary"
    code ← code land: 0363.                     "Remove underline and strikeout"
```

```
        [style=prevstyle⇒
            [ix ← fontcodes find: code.
            ix > 0 ⇒ [⇑ix]]
        fontcodes all← nil. "invalid across style change"
        prevstyle← style].
    n ← code / 16 + 1.
    font ← (WidthTable new
        named: (style fontfamily: n)
        pointsize: (style fontsize: n)
        face: (self pressface: code))
     lookup.
    (ix← fontdefs find: font)>0⇒
        [fontcodes◦ix← code. ⇑ix]
    "add entry to font dictionary"
    fontdefs length=16⇒[user notify: 'too many fonts'. ⇑1]
    fontcodes ← fontcodes, code.
    fontdefs ← fontdefs, font.
    ⇑fontcodes length]
```

Initialization

```
of: f
    [file ← f.  self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.
    fontcodes ← Vector new: 0.
    fontdefs ← Vector new: 0.
    prevstyle← nil.
    dlstart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]
⌋
```

Textframe asFollows⌋

Printing

```
hardcopy: p | t w
    [p box: window hue: self hue sat: self saturation
        bright: self brightness containing⦂
        [self pressSelect: p.
        p brightness: 0.
        w← (window intersect: frame) width← frame width.
        t← self charnearpt: w origin+1.
        ([t>1⇒[para copy: t to: para length] para])
            asParagraph presson: p
            in: (p transrect: w)
            style: style]]
```

Image

```
inset: d            "put some leading inside window"
    [frame← window inset: d]
```

Initialization

```
style← st [style← st]
"

    | each.
    MenuStyle setfont: 0 name: 'CREAM10'.
    Menu allInstances notNil transform⦂ each to⦂
```

```
        (each rescan).
    MenuPaneStyle setfont: 0 name: 'CREAM10'.
    MenuPane allInstances notNil transform⦂ each to⦂
        (each style← MenuPaneStyle).
    DefaultTextStyle setfont: 0 name: 'CREAM12'.
"
⌋

TextStyle asFollows⌋

Access
maxascent
    [maxascent≡nil⇒[⇑(fonts∘1) word: 6] ⇑maxascent]
maxdescent
    [maxdescent≡nil⇒[⇑(fonts∘1) word: 7] ⇑maxdescent]
lineheight
    [⇑self maxascent + self maxdescent]
⌋

WidthTable asFollows⌋

Initialization
lookup | key font file i
    [key ← name + pointsize asString + (↪('' 'I' 'B' 'BI')∘(face+1)).
    font ← WidthDict lookup: key⇒ [⇑font]
    file ← dp0 file: 'fonts.widths'.
    self fontfrom: file.
    file close.
    for⦂ i from: ↪(011 015 040) do⦂
        [i≥min and: i≤max ⇒ [widths∘(i-min+1) ← 0]].
    WidthDict insert: key with: self.
    ⇑self]
⌋
```

ListPane asFollows⌋

Private

```
scrollBy⦂ expr copying: src into: dest showing: item in: frame direction: n
        | strm final stop pt delay chars locked t
    [strm ← Stream new. chars ← 2*frame width/self lineheight. para ←
String new: chars.
    pt ← dest origin. final ← [n<0⇒ [0] list length+1].
    stop ← [locked←self locked⇒ [0 max: (list length+1 min: (lastShown -
firstShown * n sign + selection))] final].
    while⦂ item≠stop do⦂
        [firstShown ← firstShown + n. lastShown ← lastShown + n. item ←
item + n.
        strm of: para from: 1 to: chars.
        [item≠final⇒ [(list◦item) printon: strm] self dummy copyto: strm].
        strm cr. src blt: pt mode: storing. self show.
        (t← expr eval) abs ≤1⇒ [for⦂ delay to: chars/4 do⦂ [strm myend]. para
← nil. ⇑false]
            t*n<0⇒[⇑false]].
    para ← nil. locked and: stop≠final⇒ [locked flash. ⇑false]]
scrollControl⦂ expr
        | dY onlyFirst butFirst onlyLast butLast x1 x2 y1 y2 y3 y4 k
    ["Selection is highlighted. Unhighlight it. Invalidate my saved para if I
scroll. Then reselect selection, or deselect if it is no longer displayed."
    self compselection. dY ← self lineheight.
    x1 ← window origin x. x2 ← window corner x.
    y1 ← window origin y+2. y4 ← window height-4 |dY + y1. y2←y1+dY.
y3←y4-dY.
    onlyFirst ← x1+2⊙y1 rect: 2000⊙y2. butFirst ← x1⊙y2 rect: x2⊙y4.
    onlyLast ← x1+2⊙y3 rect: 2000⊙y4. butLast ← x1⊙y1 rect: x2⊙y3.
    while⦂ (k←expr eval)≠0 do⦂
        [k>0⇒[self scrollBy⦂ expr eval copying: butFirst into: butLast showing:
lastShown
                in: onlyLast direction: 1⇒[] ⇑self select: selection]
        self scrollBy⦂ expr eval copying: butLast into: butFirst showing:
firstShown
                in: onlyFirst direction: ¯1⇒[] ⇑self select: selection].
    self select: selection]
scrollPos
    [list≡nil⇒[⇑0.0]
    list length=0⇒[⇑0.0]
    ⇑firstShown asFloat/list length]
⌋
```

Menu asFollows⌋

Initialization

```
string: str |  i pt tpara
    [[str last≠13⇒[str←str+'
']].        "make sure str ends with CR"
    text ← Textframe new para: (tpara ← str asParagraph)
                frame:  (Rectangle new origin: (pt ←  0 ⊙ 0)
                                                corner: 1000 ⊙ 1000)
                style: MenuStyle.
    pt ← text maxx: str length+1.
    text frame growto: pt + (4  ⊙ 0).
    tpara center.
    frame ← text frame inset: ¯2 ⊙ ¯2.
    thisline ← Rectangle new origin: text frame origin
            corner: text frame corner x ⊙ text lineheight]
⌋

ScrollBar asFollows⌋

Scheduling
eachtime | p h t oldCursor
    [(rect has: user mp)≡false⇒[⇑false]
    t← rect minus: (rect inset: 0⊙(h← rect width min: rect height/4)).
    oldCursor← Cursor new frompage1.
    while⦂ [rect has: (p← user mp)] do⦂
        [t◦1 has: p⇒
            [DownCursor topage1. while⦂ user redbug do⦂
                [self reposition⦂ [owner value scrollUp: ¯42]]]
        t◦2 has: p⇒
            [UpCursor topage1. while⦂ user redbug do⦂
                [self reposition⦂ [owner value scrollUp: 42]]]
        JumpCursor topage1. while⦂ user redbug do⦂
                [self reshow⦂ [owner value scrollTo: (self position: ((0.0 max:
                    (user mp y-h-rect minY-4) asFloat/(rect height-12-(2*h)))
                    min: 1.0))]]].
    oldCursor topage1]


Image
position: f | t
    [t← rect width min: rect height/4.
    position moveto: rect origin+
            (12⊙(3+t+(f*((rect height-(2*t))-12)))).
    ⇑f]
⌋

Textframe asFollows⌋

Initialization
style← st [style← st]
"
    | each. Smalltalk define: ↪MenuStyle as: DefaultTextStyle copy.
    Smalltalk define: ↪MenuPaneStyle as: DefaultTextStyle copy.
    Menu allInstances notNil transform⦂ each to⦂ (each rescan).
    MenuPane allInstances notNil transform⦂ each to⦂ (each style←
MenuPaneStyle).
    MenuPane allInstances notNil transform⦂ each to⦂ (each style←
MenuPaneStyle).
    DefaultTextStyle setfont: 0 name: 'CREAM12'.
"
⌋

TextStyle asFollows⌋

Initialization
copy | t
    [⇑super copy fonts: fonts copy fontnames: fontnames copy]
```

```
fonts: fonts fontnames: fontnames
⌋

ParagraphEditor asFollows⌋

Private Messages
checklooks | t val mask runvals
   [t ← ↪(248 232 219 233   312 296 283 297
      217 241 226 225 210 209 211 213 262 231
      281 305 290 289 274 273 322 308) find: user kbck.
   t=0⇒[⇑false]
   user kbd.
   t=25⇒[self toBravo]; "ctl-T"
    =26⇒[self fromBravo]. "ctl-F"
      [oldpara⇒[] oldpara ← para recopy].
   runvals ← ↪(1 2 4 256   ¯1 ¯2 ¯4 256  "ctl-b i - x   B I ¯ X"
      0 16 32 48 64 80 96 112 128 144  "ctl-0 1 ... 9"
      160 176 192 208 224 240).  "ctl-shift-0 1 ... 5"
   [(val← runvals◦t)=256⇒[mask← 0377.  val← 0]                              "reset all"
      val<0⇒[mask← 0-val.  val← 0]                   "reset emphasis"
      val>0 and⦂ val<16⇒[mask← val]                  "set emphasis"
      mask← 0360].               "set font"
   para maskrun: loc1 to: loc2-1 under: mask to: val]

" Set up kbmap for unmapped ctl chars:
    | x. ↪(248 232 219 233   312 296 283 297
      217 241 226 225 210 209 211 213 262 231
      281 305 290 289 274 273 0502 0464) transforms⦂ x to⦂ (kbMap◦x← x).
   ParagraphEditor classvars delete: ↪(ctlchars runvals).
"

classInit
   [bs ← 010. ctlw ← 013. esc ← 020.
   cut ← 127. paste ← 2]
⌋

Rectangle asFollows⌋

Conversion
bitsOntoStream: strm | len rec s
   [rec ← origin rect: origin + (self width ⌾ (16 min: self height)).
   s← rec bitsIntoString all← 0.  len← 0.
   while⦂ rec maxY ≤ corner y do⦂
      [rec bitsIntoString: s.
      strm append: s.
      len ← len + (s length / 2).
      rec moveby: 0⌾16].
   rec minY<corner y⇒
      [s← (rec origin rect: corner) bitsIntoString.
      strm append: s.  ⇑len+s length]
   ⇑len]
⌋

TextStyle asFollows⌋

Fonts
fontfamily: n | s char
```

```
    ["return the family name taken out of fontnames"
    s ← Stream default.
    for⦂ char from: fontnames ◦ n do⦂
        [char isletter or⦂ char=('-'◦1)⇒
            [s next ← char]
        ⇑s contents]]
fontsize: n | s c size
    ["return size from fontname"
    s ← (fontnames ◦ n) asStream.
    until⦂ (c ← s next) isdigit do⦂ [].
    size← c-060.
    while⦂ [c ← s next] do⦂
        [size ← size*10 + (c - 060). ].
    ⇑size]
⌋

PanedWindow asFollows⌋

Window protocol
hardcopy: p | pane w
    [self showtitle.
    for⦂ w from: (titleframe window, titleframe frame) do⦂
        [w width← w width*11/10].
    titleframe hardcopy: p.
    for⦂ pane from: panes do⦂ [pane hardcopy: p]]
⌋

PressFile asFollows⌋

Scaling and Transformations
transpt: p
    [⇑ (p x * scale) asInteger ⌾ (24500 - (p y * scale)) asInteger]
⌋
```

EARS

Filename: systemchanges.st,

Creation Date: April 26, 1978 12:30 PM

Printed by: Weyer, Steve

```
'From Smalltalk 5.2j on 9 January 1976 at 10:49:29 pm.'⌋

CodePane asFollows⌋

As stream
append: text        "append text (a string or paragraph) to my end; don't show"
   [pared append: text]
display       "scroll to the end of the paragraph and show it"
   [pared display]
⌋

CodeWindow asFollows⌋

As stream
append: text
   [(panes∘1) append: text]
cr
   [(panes∘1) append: 015 inString]
display
   [(panes∘1) display]
next ← char
   [(panes∘1) append: char inString]
print: x
   [(panes∘1) append: x asString]
space
   [(panes∘1) append: ' ']
⌋
```

copies false?

```
ParagraphEditor asFollows⌋

Public Messages
append: text  "append text (a string or paragraph) to my end; don't show"
   [self fintype.
    [oldpara⇒ [] oldpara ← oldpara copy].
    para ← para replace: para length+1 to: para length by: text]
display  "scroll to the end of the paragraph and show it"
   [self fintype; select: para length+1; show]
⌋

ParagraphEditor asFollows⌋
```

✓ SW 6-1

```
Public Messages
selectAndScroll | l dY i
   [l ← self lineheight. self select.
   dY ← p1 y - window origin y.
    [dY≥0⇒ [dY ← (p1 y + l-1) - (window corner y) max: 0]].
   dY≠0⇒[self scrollby: (dY abs+l-1)/l*dY sign]
   ]
⌋

Point asFollows⌋

Arithmetic
+ delta
```

```
    ["Return a Point that is the sum of me and delta (which is a Point or
Number)"
    ⇑Point new x: (x + delta asPtX) y: (y + delta asPtY)
    ]
- delta
    ["Return a Point that is the difference of me and delta (which is a Point
or Number)"
    ⇑Point new x: (x - celta asPtX) y: (y - delta asPtY)
    ]
⌋
```

✓sw

✓sw

```
SystemOrganizer asFollows⌋

Mapping
classesIn: cat | c υ                  "⇑Vector of all classes in this category (-ies)"
    [cat is: String⊃
        [⇑(self category: cat) transform⦂ c to⦂ Smalltalk◦c]
    υ← Vector new: 0.
    for⦂ c from: cat do⦂
        [υ← υ+(self classesIn: c)]
    ⇑υ]
whoIn: cat sends: message | x t
    [⇑((self classesIn: cat) transform⦂ x to⦂
        [(t← x whosends: message) length=0⊃[nil]
        '
'+x title+' '+t]) notNil]
"
SystemOrganization whoIn: ↪('Forms' 'Simulator' 'Car Wash' 'Playground')
sends: ↪hide
"
⌋
```

?X

```
ClassOrganizer asFollows⌋

Access to parts
classify: selector alsounder: heading | s
    [selector is: Vector⊃
        [for⦂ s from: selector do⦂
            [self classify: s under: heading]]
    s ← commentVector find: heading.
    s>0 and⦂ (groupVector◦s has: selector)⊃[⇑self]
    [s=0⊃ [s ← self insert: heading]].
    groupVector◦s ← groupVector◦s insertSorted: selector.
    ]
⌋
```

bug →

alsounder:

✓

```
UserView asFollows⌋

Special windows
notify: errorString | notifyWindow s
    ["Create a notify window looking at the Context stack"
    Top currentPriority≠1⊃
        [notifyWindow ← self notifier: errorString stack: thisContext sender
interrupt: false.
        notifyWindow⊃
            [thisContext sender ← nil.
            self scheduleOnBottom: notifyWindow.
            Top errorReset]
        ⇑nil]
      s ← thisContext sender.
      thisContext sender ← nil.
      (user sched∘1) showError: errorString stack: s interrupt: false]
oldnotify: errorString | notifyWindow
    ["Create a notify window looking at the Context stack"
    notifyWindow ← self notifier: errorString stack: thisContext sender
interrupt: false.
    notifyWindow⇒
        [thisContext sender ← nil.
         Top currentPriority=1⇒
            [self restartup: notifyWindow]
         self scheduleOnBottom: notifyWindow.
         Top errorReset]
    ⇑nil]
⌋

Object asFollows⌋

System Primitives
showError: errorString stack: context interrupt: flag | notifyWindow
    [notifyWindow ← user notifier: errorString stack: context interrupt: flag.
     notifyWindow ⇒ [user restartup: notifyWindow]]
⌋

PriorityScheduler asFollows⌋

Top level
init11      " Top terminate: 11; init11. "
    [GRODSK ← Top
        install⦂
            [user displayoffwhile⦂
                [dp0 growSmalltalkBy: 100].
            GRODSK deepsleep]
        at: 11.
    GRODSK enable]
⌋
```

```
Class new title: 'ScrollBar'
   subclassof: Object
   fields: 'rect bitstr owner position'
   declare: 'JumpCursor DownCursor UpCursor ';
   asFollows⌋
```

I am a bar to the left of an awake window. With the cursor in me I can make that window scroll.

Initialization

```
classInit        "ScrollBar classInit"
   [UpCursor ← Cursor new fromtext: '
0000000110000000
0000001111000000
0000011111100000
0000111111110000
0001111111111000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000'.
   DownCursor ← Cursor new fromtext: '
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0000000110000000
0001111111111000
0000111111110000
0000011111100000
0000001111000000
0000000110000000'.
   JumpCursor ← Cursor new fromtext: '
0000001000000000
0110001100000000
1111111110000000
0110001100000000
0000001000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000']
on: f from: o
   [self on: f from: o at: o scrollPos]
on: frame from: o at: f | w
   [owner ← o asCitation.
   rect ← Rectangle new
      origin: frame origin-((w←24)⊙2)
      extent: w⊙(frame height+4).
   position ← Rectangle new origin: 0⊙0 extent: 8⊙6.
   self position: f]

Scheduling
close
   [owner←nil]
eachtime | p h t oldCursor
   [(rect has: user mp)≡false⇒[⇑false]
   t← rect minus: (rect inset: 0⊙(h← rect width min: rect height/4)).
   oldCursor← Cursor new frompage1.
   while¨ [rect has: (p← user mp)] do¨
      [t◦1 has: p⇒
         [DownCursor topage1. while¨ user redbug do¨
            [self reposition¨ [owner value scrollUp: ¯40]]]
      t◦2 has: p⇒
         [UpCursor topage1. while¨ user redbug do¨
            [self reposition¨ [owner value scrollUp: 40]]]
      jumpCursor topage1. while¨ user redbug do¨
            [self reposition¨ [owner value scrollTo: ((0.0 max:
               (user mp y-h-rect minY-4) asFloat/(rect height-12-(2*h)))
               min: 1.0)]]].
   oldCursor topage1]

firsttime
   [⇑rect has: user mp]
lasttime

Image
hide      "restore background"
   [bitstr≡nil⇒ [user notify: 'Attempt to hide unshown scrollbar']
   rect bitsFromString: bitstr.  bitstr← nil]
hidewhile¨ expr  | v
   [self hide. v ← expr eval. self show. ⇑v]
position: f | t
   [t← rect width min: rect height/4.
   position moveto: rect origin+
         ((rect width-position width/2)⊙(3+t+(f*((rect height-(2*t))-12))))]
reposition¨ expr
   [self reshow¨
      [expr eval. self position: owner value scrollPos]]
reshow¨ expr | r
   [r ← position inset: ¯1.  expr eval.
   r clear: white.  position outline: 1]
```

```
show | r      "Save background and show"
   [bitstr ← rect bitsIntoString.
   rect clear: black.
   (r← rect inset: 2⊙2 and: 1⊙2) clear: ltgray.
   (r inset: 0⊙(rect width min: rect height/4)) outline: 1.
   position outline: 1]

Reclamation
keepCitations
   [owner keep]
⌟
SystemOrganization classify: ↪ScrollBar under: 'Panes and Menus'.⌟
ScrollBar classInit⌟
```

```
ParagraphEditor asFollows⌋

Public Messages
again | t
    [ [self fintype⇒[Scrap← Scrap text]].
    t← para findString: Deletion startingAt: loc2.          ✓ sw 6-1
    t=0⇒[frame flash]
    loc1← t.  loc2← loc1+Deletion length.
    self paste]

Private Messages
fintype
    [typein⇒
        [   [typein<loc1⇒
                [Scrap ← para copy: typein to: loc1-1.        ✓ sw 6-1
                loc1 ← typein]].
        typein ← false]
    ⇑false]

Public Messages
paste [self fintype; replace: Scrap; selectAndScroll]        ✓ sw 6-1

Private Messages
replace: t
    [   [oldpara⇒[] oldpara ← para copy].
        [typein≡false⇒[Deletion ← self selection]].            ✓ sw 6-1
    para ← para replace: loc1 to: loc2-1 by: t.
    loc2 ← loc1 + t length.
    self show]

Public Messages
scrollby: n | l w
    [n ← n max: (frame origin y-4-window origin y)/(l← self lineheight).
    n ← n*l.
    frame moveby: 0⊙(0-n).
    w ← [n<0⇒[window inset: 0⊙0 and: 0⊙(0-n)]
            window inset: 0⊙n and: 0⊙0].
    w empty⇒[self show; select]
    w blt: w origin-(0⊙n) mode: storing.
        [n<0⇒[w corner y ← w origin y - n]
            w origin y ← w corner y - n].
    self showin: w; selectin: w]
scrollPos | t
    [t← (self ptofchar: para length+1) y - frame minY.        ✓ sw 6-1
    t=0⇒[⇑0.0]
    ⇑(window minY-frame minY) asFloat / t]
scrollTo: f
    [sզlf scrollUp:
            (f*(self lineheight + (self ptofchar: para length+1) y - frame minY))   ✓ sw 6-1
asInteger -
            (window minY+4-frame minY)]
typing | more char
    [[loc1<loc2⇒[self checklooks⇒[self show. ⇑self select]]].
    more ← Stream default.
    [typein⇒[] Deletion← self selection. typein ← loc1].          ✓ sw 6-1
    while⦂ user kbck do⦂
        [(char ← user kbd)
        =bs⇒ [more empty⇒[loc1 ← 1 max: loc1-1. typein ← typein min: loc1]
                more skip: ¯1];                              "backspace"
        =ctlw⇒ [more reset. loc1 ← 1 max: loc1-1.                    "ctl-w for backspace
word"
                while⦂ [loc1>1⇒[(para◦(loc1-1)) tokenish] false] do⦂ [loc1 ← loc1-1]
                typein ← typein min: loc1];
        =cut⇒ [⇑self cut];
        =paste⇒ [⇑self paste];
        =esc⇒ [        [more empty⇒[] self replace: more contents. loc1 ← loc2].
                self fintype. "select previous type-in"
                loc1 ← loc2-Scrap length. ⇑self select]
        more next← char].
    self replace: more contents. loc1←loc2.
    user kbck⇒[] self selectAndScroll]
⌟
```

```
ListPane asFollows⌋

Private
scrollControl⦂ expr
        | dY onlyFirst butFirst onlyLast butLast x1 x2 y1 y2 y3 y4 k
    ["Selection is highlighted.  Unhighlight it.  Invalidate my saved para if I
scroll.  Then reselect selection, or deselect if it is no longer displayed."
    self compselection. dY ← self lineheight.
    x1 ← window origin x. x2 ← window corner x.
    y1 ← window origin y+2. y4 ← window height-4 |dY + y1. y2←y1+dY.
y3←y4-dY.
    onlyFirst ← x1+2⊙y1 rect: 2000⊙y2. butFirst ← x1⊙y2 rect: x2⊙y4.
    onlyLast ← x1+2⊙y3 rect: 2000⊙y4. butLast ← x1⊙y1 rect: x2⊙y3.
    while⦂ (k←expr eval)≠0 do⦂
        [k>0⇒[self scrollBy⦂ expr eval copying: butFirst into: butLast showing:
lastShown
                in: onlyLast direction: 1]
        self scrollBy⦂ expr eval copying: butLast into: butFirst showing:
firstShown
                in: onlyFirst direction: ¯1].
    self select: selection]

As yet unclassified
scrollPos
    [firstShown≡nil or⦂ list length=0⇒[⇑0.0]
    ⇑firstShown asFloat / list length]

Private
scrollUp: n
    [self scrollControl⦂
        [user buttons=4⇒
            [n sign*[mem◦0177036 nomask: 0100⇒[2] 1]]
        0]]
⌋

ParagraphEditor asFollows⌋

Public Messages
again | t
    [ [self fintype⇒[Scrap← Scrap text]].
    t← para findString: Deletion startingAt: loc2.
    t=0⇒[frame flash]
    loc1←t.  loc2← loc1+Deletion length.
    self paste]

Private Messages
fintype
    [typein⇒
        [    [typein<loc1⇒
                [Scrap ← para copy: typein to: loc1-1.
                loc1 ← typein]].
        typein ← false]
    ⇑false]
```

```
Public Messages
paste [self fintype; replace: Scrap; selectAndScroll]

Private Messages
replace: t
    [   [oldpara⇒[] oldpara ← para copy].
        [typein=false⇒[Deletion ← self selection]].
    para ← para replace: loc1 to: loc2-1 by: t.
    loc2 ← loc1 + t length.
    self show]

Public Messages
scrollby: n | l w
    [n ← n max: (frame origin y-4-window origin y)/(l← self lineheight).
    n ← n*l.
    frame moveby: 0⌾(0-n).
    w ← [n<0⇒[window inset: 0⌾0 and: 0⌾(0-n)]
          window inset: 0⌾n and: 0⌾0].
    w empty⇒[self show; select]
    w blt: w origin-(0⌾n) mode: storing.
        [n<0⇒[w corner y ← w origin y - n]
          w origin y ← w corner y - n].
    self showin: w; selectin: w]
scrollPos | t
    [t← (self ptofchar: para length+1) y - frame minY.
    t=0⇒[⇑0.0]
    ⇑(window minY-frame minY) asFloat / t]
scrollTo: f
    [self scrollUp:
          (f*(self lineheight + (self ptofchar: para length+1) y - frame minY))
asInteger -
          (window minY+4-frame minY)]
typing | more char
    [[loc1<loc2⇒[self checklooks⇒[self show. ⇑self select]]].
    more ← Stream default.
    [typein⇒[] Deletion← self selection. typein ← loc1].
    while⦂ user kbck do⦂
        [(char ← user kbd)
        =bs⇒ [more empty⇒[loc1 ← 1 max: loc1-1. typein ← typein min: loc1]
                more skip: ¯1];                         "backspace"
        =ctlw⇒ [more reset. loc1 ← 1 max: loc1-1.              "ctl-w for backspace
word"
                while⦂ [loc1>1⇒[(para◦(loc1-1)) tokenish] false] do⦂ [loc1 ← loc1-1]
                typein ← typein min: loc1];
        =cut⇒ [⇑self cut];
        =paste⇒ [⇑self paste];
        =esc⇒ [       [more empty⇒[] self replace: more contents. loc1 ← loc2].
                self fintype. "select previous type-in"
                loc1 ← loc2-Scrap length. ⇑self select]
        more next← char].
    self replace: more contents. loc1←loc2.
    user kbck⇒[] self selectAndScroll]
⌋

Textframe asFollows⌋

Image
```

```
scrollPos | t
    [t← (self ptofchar: para length+1) y - frame minY.
    t=0⇒[⇑0.0]
    ⇑(window minY-frame minY) asFloat / t]
⌋

ListPane asFollows⌋

Private
scrollTo: f | t
    [self scrollControls
        [t← (f*list length) asInteger-firstShown.
        t<0⇒[firstShown≤0⇒[0] t]
        lastShown>list length⇒[0] t]]
⌋
```

```
Class asFollows⌋
printSubclassesOn: strm indent: n | x
    [strm cr. for⦂ x to: n do⦂ [strm tab].
    strm append: title.
    for⦂ x from: AllClassNames do⦂
        [(Smalltalk◦x) superclass≡self⇒
            [Smalltalk◦x printSubclassesOn: strm indent: n+1]]]
"
 | f. user displayoffwhile⦂
    [f← dp0 file: 'subclasses'.
    Object printSubclassesOn: f indent: 0.
    f shorten; close].
"
⌋

Dispframe asFollows⌋

Scheduler
hardcopy: p [text hardcopy: p]
⌋

ListPane asFollows⌋

Private
dummy
    [⇑'———']
fill | i        "Given firstShown, compute lastShown and show me."
    [lastShown ← firstShown-1 + (window extent y-4 /self lineheight)
        min: list length+1.
    [self locked⇒
        [i ← (selection-lastShown max: 0) + (selection-firstShown min: 0).
        i≠0⇒ [para←nil. firstShown ← firstShown + i. lastShown ← lastShown
+ i]]].
    (frame ← window inset: 2) width ← 999.
    self refresh;  show]

Pane protocol
hardcopy: p [self r:fresh. ⇑super hardcopy: p]

As yet unclassified
pressSelect: p
    [selection=0⇒[]
    ColorPrint≡false⇒[]
    p hue: 40; saturation: 200.
    p showrect: self selectionRect color: 255]

Private
refresh | i s            "If para is nil, then recompute it"
    [para≡nil⇒
        [s ← (String new: 256) asStream.
        for⦂ i from: (firstShown to: lastShown) do⦂
            [[0<i and⦂ i≤list length⇒ [(list◦i) printon: s] self du·nmy copyto: s].
            s cr].
        para ← s contents]]
```

X?

⌋

PanedWindow asFollows⌋

Window protocol

```
hardcopy | p t
    [user displayoffwhile⦂
        [p ← dp0 pressfile: (t← (self title + '.press.') asFileName).
        self hardcopy: p.
        p close.
        user quitThen:
'Empress ' + t + '
Resume small.boot.
']]
⌋
```

Paragraph asFollows⌋

Press printing

```
presson: strm in: r
    [⇑self presson: strm in: r style: DefaultTextStyle]                          ✓sw
presson: strm in: r style: style               "Output paragraph inside rectangle (page
coordinates)"
        | char s1 s2 s3 y chop
    [s3 ← ParagraphScanner new of: self to: strm style: style.
    s3 init in: r.  y ← r corner y.
    s1 ← s3 copy. s2 ← s3 copy.
    chop ← [alignment=1⇒ [0] alignment].
    while⦂ (char ← s3 scan) do⦂
        [char ≡ true ⇒              "Exceeded max width"
            [[s2 position = s1 position ⇒                   "No blanks in line"
                [y ← s3 printfrom: s1 align: 0 backup: 1. s3 backup. s3 copyto: s2]
             y ← s2 printfrom: s1 align: alignment backup: 1].
            y ⇒
                [s2 init in: (r origin rect: r corner x ⊙ y). s2 copyto: s1. s2 copyto:
s3]
            ⇑ self copy: s1 position + 1 to: text length]
        char = 040 ⇒                                                              ✓sw
            [s3 copyto: s2. s3 space]
        char = 015 ⇒
            [y ← s3 printfrom: s1 align: chop backup: 1 ⇒
                [s3 init in: (r origin rect: r corner x ⊙ y). s3 copyto: s1. s3 copyto:
s2]
            ⇑ self copy: s1 position + 1 to: text length]
        char = 011 ⇒
            [s3 tab]
        "user notify: 'unimplemented control char'"]
    "Put out trailing text if any"
    s3 width=0⇒ [⇑y]
    y ← s3 printfrom: s1 align: chop backup: 0 ⇒ [⇑y]
    ⇑ self copy: s1 position + 1 to: text length]

⌋
```

ParagraphScanner asFollows⌋

Scanning

```
scan "Scan up to a zero-width character"
        | char w maxw cd len pos
    [maxw ← rec: width asInteger.
    [textstrm end ⇒ []
     ascent ← ascent max: font ascent.
     descent ← descent min: font descent.
    ].
    while§ (char ← textstrm next) do§
        [w ← font widthof: char.
        w = 0 ⇒ [⇑char]
        width ← width + w.
        width > maxw ⇒ [⇑true]].
    while§ (len  ← runstrm next) do§
        [cd ← runstrm next.
        len = 0 ⇒ []                "Go to next run"
        pos ← textstrm position.
        textstrm of: para text from: pos+1 to: pos+len.
        font ← press codefont: cd style: style.
        ⇑self scan].
    ⇑false]
⌋
```

✓sw

PressFile asFollows⌋

Bitmap support

```
bitmap: rect bits: bits | r dlpos w w1 h len "w1 is for Swinehart"
    [boundbox ← boundbox max: (r ← self transrect: rect).
    dlpos ← file wordpos.
        [dlpos*2=file position⇒[]
        self skipchars: 1. file positioneven: 0. dlpos←dlpos+1].
    w ← rect width. h ← rect height.
    self setcodingdots: (w1 ← w+15|16) lines: h;
        setmode;
        setsizewidth: (scale * w1) height: (scale * h);
        setwindowwidth: w1 height: h;
        dotsfollow.
    [bits⇒[file append: bits]                   "bits supplied"
        rect bitsOntoStream: file].                   "else from screen"
    self setp: r origin;
        showdots: (file wordpos - dlpos)]
```

✓sw

File structure commands

```
close | p i fp
    [   [EL empty⇒[] self closePage].
    "put out font directory, part directory, document directory"
    self fontdir.
    fp ← file pages.
    parts puton: file.
    file padpage.
    p ← file pages.
    file nextwordvector ← (27183, "press password"
        (p + 1) "number of records",
        (parts position / 8) "number of parts",
        fp, (p - fp), "part dir and length"
        ¯1, ¯1, ¯1, "junk"
        1, 1, "first and last copies"
```

✓sw

```
        ('S'o1 "solid color")).
    for: i from: (10 to: 0177) do: [file nextword← ¯1].
    file append: (file title asBCPL: 52);
        append: (dp0 use name asBCPL: 32);
        append: (Date default asString+' '+Time default asString asBCPL: 40);
        padpage; shorten; close]
closePage | padding dllength st
    ["Assumes entity trailer has been put on"
    dllength ← file position - dlstart.                          ✓sw
    file nextword ← 0.
    EL puton: file; reset.
    padding ← file padpage. st ← dlstart/512.
    parts nextwordvector ← (0, st, (file position/512 - st), padding).
    dlstart ← file position]
entity: box containing: expr | fp1 fp2 EL1 v
    [fp1← file position. EL1← EL position/2.
    v← expr eval.
    EL positioneven: 255.          "word-pad EL with <Nop>"
    file positioneven: 0.          "word-pad DL with 0"
    fp2 ← file position - fp1.
    "Put a trailer into the EL"
    EL nextwordvector ← (0, "type and fontset"                   ✓sw
        0, fp1 "dlstart relative to DL location in file", 0, fp2,
        800, 1200, "paper origin"
        (box origin x), (box origin y),
        (box width), (box height)).
    EL nextword ← EL position/2 - EL1 + 1.
    ⇑v]
```

Initialization

```
of: f
    [file ← f. self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.
    fontcodes ← Vector new: 0.                                   ✓sw
    fontdefs ← Vector new: 0.
    dlstart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]
```

Bitmap support

```
screenout: rect scale: scale
    ["puts a bit map image onto the PressFile. The standard
    scaling is 32 micas per Alto dot. 22 looks better, Dover only
    works with 32"
    user displayoffwhile:                                        ✓sw
        [self somefont.
        self bitmap: rect bits: false.
        self page.
        self close]]
┘
```

Textframe asFollows┘

Printing

```
brightness [⇑255]
hardcopy | p
    [p ← dp0 pressfile: 'frame.press'.
    self hardcopy: p.
    p close]
hardcopy: p | e w
    [p entity: (p transrect: (w← window inset: ¯2)) containing⦂
        [   [ColorPrint⇒[p hue: self hue; saturation: self saturation.
                p showrect: window color: self brightness]].
        self pressSelect: p.
        for⦂ e from: (w minus: window) do⦂
            [p showrect: e color: 0].
        para asParagraph presson: p
            in: (p transrect: ((window inset: 0) width← frame width))
            style: style]]
hue [⇑120]
pressSelect: p            "ignored"
saturation [⇑191]
┘

UserView asFollows┘

Display
displayoffwhile⦂ expr | i t v
    [t ← memo◦067.
    for⦂ i from: t to: 58 by: ¯2 do⦂
        [memo◦067 ← i. 1/1/1/1/1 "slow"].
    v ← expr eval.
    for⦂ i from: 58 to: t by: 2 do⦂
        [memo◦067 ← i. 1/1/1/1/1 "slow"].
    ⇑v]
┘
```

```
UserView asFollows⌋

Window Scheduling
restore | w
    ["screenrect clear. disp outline.
    for⦂ w from: (sched length to: 1 by: ¯1) do⦂
        [(sched∘w) show]"
     (sched∘1) refresh]
⌋

Window asFollows⌋

Framing
refresh
    [self show]
⌋

~~ListPane asFollows⌋~~

~~Pane protocol~~
~~outline~~
    ~~[window outline: 2]~~
~~⌋~~

ScrollBar asFollows⌋

Image
hide        "restore background"
    [bitstr≡nil⇒ []
    rect bitsFromString: bitstr.  bitstr← nil]
⌋

~~PanedWindow asFollows⌋~~

~~Window protocol~~
~~outline~~
    ~~[frame outline: 2]~~
~~⌋~~

ScrollBar asFollows⌋

Scheduling
firsttime
    [⇑ (rect inset:¯5⊙ ¯5) has: user mp]
⌋

ListPane asFollows⌋

Private
fill | i          "Given firstShown, compute lastShown and show me."
    [lastShown ← ~~firstShown-1 + (window~~ extent y-4 /self lineheight)
        min: list length+1.
    [self locked⇒
        [i ← (selection-lastShown max: 0) + (selection-firstShown min: 0).
        i=0⇒ [para←nil. firstShown ← firstShown + i. lastShown ← lastShown
+ i]]].
    (frame ← window inset: 2) width ← 999.
    self refill; show]

Pane protocol
hardcopy: p [self refill. ⇑super hardcopy: p]
outline
    [window outline: 1]

Private
refill | i s                "If para is nil, then recompute it"
    [para≡nil⇒
        [s ← (String new: 256) asStream.
        for⦂ i from: (firstShown to: lastShown) do⦂
            [[0<i and⦂ i≤list length⇒ [(list∘i) printon: s] self dummy copyto: s].
            s cr].
        para ← s contents]]
⌋

Pane1Window asFollows⌋

Window protocol
outline
    [frame outline: 1]
⌋

ScrollBar asFollows⌋

Image
position: f | t
    [t← rect width min: rect height/4.
    position moveto: rect origin+
            ((rect width-6/2)⊙(3+t+(f*((rect height-(2*t))-12)))).
    ⇑f]
⌋

Textframe asFollows⌋

Image
outline
    [window border: 1 color: black]
⌋

WidthTable asFollows⌋

Initialization
lookup | key font file i
    [key ← name + pointsize asString + (↪('' 'I' 'B' 'BI')∘(face+1)).
    font ← WidthDict lookup: key⇒ [⇑font]
    file ← dp0 file: 'fonts.widths'.
    self fontfrom: file.
    file close.
```

✓SW

```
    for⦂ i from: ↪(011 015 040) do⦂
        [i≥min and: i≤max ⇒ [widths∘(i-min+1) ← 0]].
    WidthDict insert: key with: self.
    user show: 'OK.'; cr.
    ⇑self]
⌋

LargeInteger asFollows⌋

Conversion
asInteger | t i
    [bytes length>3⇒[⇑self]
    t ← 0.
    for⦂ i from: bytes length to: 1 by: ¯1 do⦂
        [t ← t*0200+(bytes∘i)].
    neg⇒[⇑0-t]
    ⇑t]
⌋

ListPane asFollows⌋

Private
fill | i          "Given firstShown, compute lastShown and show me."
    [lastShown ← firstShown-1 + (window extent y-4/self lineheight)
        min: list length+1.
    [self locked⇒
        [i ← (selection-lastShown max: 0) + (selection-firstShown min: 0).
        i≠0⇒ [para←nil. firstShown ← firstShown + i. lastShown ← lastShown
+ i]]].
    (frame ← window inset: 2) width ← 999.
    [para≡nil⇒[self refill]].
    self show]

Pane protocol
hardcopy: p
    [(frame ← window inset: 2) width ← 999.
    self refill. ⇑super hardcopy: p]

Private
refill | i s          "Recompute para from list"
    [s ← (String new: 256) asStream.
    for⦂ i from: (firstShown to: lastShown) do⦂
        [   [0<i and⦂ i≤list length⇒ [(list∘i) printon: s]
                self dummy copyto: s].
        s cr].
    para ← s contents asParagraph]
⌋

Paragraph asFollows⌋

Normal access
+ c        "Concatenates two paragraphs"
    [c← c asParagraph.
    ⇑Paragraph new
        text: text + c text
```

√sw

```
      runs: (self runcat: self runs to: c runs)
      alignment: alignment]

Manipulation of format runs
allBold [self allStyle: 1]
allStyle: val
   [runs← self makerun: text length val: val]
⌋

ParagraphEditor asFollows⌋

Public Messages
append: text  "append text (a string or paragraph) to my end; don't show"
   [self fintype.
      [oldpara⇒ [] oldpara ← oldpara copy].
    para ← para + text]
⌋

ParagraphScanner asFollows⌋

Printing
printfrom: ps align: align backup: n "Returns false if goes below bottom"
      | ybot a b p xs sp rs c len tpos ts
   [ybot ← rect corner y - (style lineheight*32).
   ybot < rect origin y ⇒ [⇑false]                    "Won't fit"
   a ← ps position + 1. b ← textstrm position - n.
   a > b ⇒ [⇑ybot]              "No text"
   p ← rect origin x ⊙ (rect corner y - (style maxascent*32)).
   xs ← rect width - width.
   sp ← font space.               "Kludge"
   ts ← tabpos contents asStream.
   [align
      = 2 ⇒ [press setp: xs/2⊙0+p; setspacex: sp];
      = 4 ⇒ [press setp: xs⊙0+p; setspacex: sp].
      press setp: p; setspacex: sp].
   press append: para text∘(a to: b).
   rs ← (para run: a to: b) asStream.
   tpos ← ts next.
   while⦂ (len ← rs next) do⦂
      [c ← rs next land: 0363.                    "Remove underline and strikeout"
      len = 0 ⇒ []
      press selectfont: (press fontindex: c style: style) - 1.
      b ← a+len.
      while⦂ (tpos and⦂ tpos<b) do⦂                     "Put out tabs"
         [press showchars: tpos-a;
            skipchars: 1;
            setx: p x + ts next.
         a ← tpos+1. tpos ← ts next].
      [align=1 and⦂ tpos=false ⇒                    "Reset space width"
         [press setspacex: xs/spaces+sp.
         align ← 0]].
      press showchars: b-a.
      a ← b].
   ⇑ybot]
⌋
```

✓ SW

```
PressFile asFollows⌋

Font handling stuff
fontindex: code style: style | ix font n
    ["return index if in font dictionary"
    code ← code land: 0363.                        "Remove underline and strikeout"
        [style=prevstyle⇒
            [ix ← fontcodes find: code.
            ix > 0 ⇒ [⇑ix]]
        fontcodes all← nil. "invalid across style change"
        prevstyle← style].
    n ← code / 16 + 1.
    font ← (WidthTable new
        named: (style fontfamily: n)
        pointsize: (style fontsize: n)
        face: (self pressface: code))
     lookup.
    (ix← fontdefs find: font)>0⇒
        [fontcodes◦ix← code. ⇑ix]
    "add entry to font dictionary"
    fontdefs length=16⇒[user notify: 'too many fonts'. ⇑1]
    fontcodes ← fontcodes, code.
    fontdefs ← fontdefs, font.
    ⇑fontcodes length]
```

✓SW

```
Initialization
of: f
    [file ← f.  self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.
    fontcodes ← Vector new: 0.
    fontdefs ← Vector new: 0.
    prevstyle← nil.
    dlstart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]
⌋
```

✓SW

```
Textframe asFollows⌋

Printing
hardcopy: p | t w
    [p box: window hue: self hue sat: self saturation
        bright: self brightness containing§
        [self pressSelect: p.
        p brightness: 0.
        w← (window intersect: frame) width← frame width.
        t← self charnearpt: w origin+1.
        ([t>1⇒[para copy: t to: para length] para])
            asParagraph presson: p
            in: (p transrect: w)
            style: style]]

Image
inset: d            "put some leading inside window"
   [frame← window inset: d]
```

```
Initialization
style← st [style← st]
"
    | each.
    MenuStyle setfont: 0 name: 'CREAM10'.
    Menu allInstances notNil transform⦂ each to⦂
        (each rescan).
    MenuPaneStyle setfont: 0 name: 'CREAM10'.
    MenuPane allInstances notNil transform⦂ each to⦂
        (each style← MenuPaneStyle).
    DefaultTextStyle setfont: 0 name: 'CREAM12'.
"
⌋

TextStyle asFollows⌋

Access
lineheight
    [⇑self maxascent + self maxdescent]
maxascent
    [maxascent=nil⇒[⇑(fonts∘1) word: 6] ⇑maxascent]
maxdescent
    [maxdescent=nil⇒[⇑(fonts∘1) word: 7] ⇑maxdescent]
⌋
```

✓ SW

✓

✓

```
WidthTable asFollows⌋

Initialization
lockup | key font file i
    [key ← name + pointsize asString + (↪('' 'I' 'B' 'BI')∘(face+1)).
    font ← WidthDict lookup: key⇒ [⇑font]
    file ← dp0 file: 'fonts.widths'.
    self fontfrom: file.
    file close.
    for⦂ i from: ↪(011 015 040) do⦂
        [i≥min and: i≤max ⇒ [widths∘(i-min+1) ← 0]].
    WidthDict insert: key with: self.
    ⇑self]
⌋
```

✓SW

```
SystemPane asFollows⌋

Window protocol
enter       "be sure I am up to date"
    [mySysOrgVersion=user classNames⇒ [super enter]
    window outline. self update.
    super enter]
⌋
```

✓SW,AB

```
Class asFollows⌋
printSubclassesOn: strm indent: n | x
    [strm cr. for⦂ x to: n do⦂ [strm tab].
    strm append: title.
    for⦂ x from: AllClassNames do⦂
        [(Smalltalk◦x) superclass≡self⇒
            [Smalltalk◦x printSubclassesOn: strm indent: n+1]]]
"
 | f. user displayoffwhile⦂
    [f← dp0 file: 'subclasses'.
    Object printSubclassesOn: f indent: 0.
    f shorten; close].
"
⌋

Dispframe asFollows⌋

Scheduler
hardcopy: p [text hardcopy: p]
⌋

ListPane asFollows⌋

Private
dummy
    [⇑'———']
fill | i          "Given firstShown, compute lastShown and show me."
    [lastShown ← firstShown-1 + (window extent y-4/self lineheight)
        min: list length+1.
    [self locked⇒
        [i ← (selection-lastShown max: 0) + (selection-firstShown min: 0).
        i≠0⇒ [para←nil. firstShown ← firstShown + i. lastShown ← lastShown
+ i]]].
    (frame ← window inset: 2) width ← 999.
    self refresh;  show]

Pane protocol
hardcopy: p [self refresh. ⇑super hardcopy: p]

As yet unclassified
pressSelect: p
    [selection=0⇒[]
    ColorPrint≡false⇒[]
    p hue: 40; saturation: 200.
    p showrect: self selectionRect color: 255] .

Private
refresh | i s          "If para is nil, then recompute it"
    [para≡nil⇒
        [s ← (String new: 256) asStream.
        for⦂ i from: (firstShown to: lastShown) do⦂
            [[0<i and⦂ i≤list length⇒ [(list◦i) printon: s] self dummy copyto: s].
            s cr].
        para ← s contents]]
```

```
⌟

PanedWindow asFollows⌟

Window protocol
hardcopy | p t
   [user displayoffwhile⦂
      [p ← dp0 pressfile: (t← (self title + '.press.') asFileName).
      self hardcopy: p.
      p close.
      user quitThen:
'Empress ' + t + '
Resume small.boot.
']]
⌟

Paragraph asFollows⌟

Press printing
presson: strm in: r
   [⇑self presson: strm in: r style: DefaultTextStyle]
presson: strm in: r style: style                    "Output paragraph inside rectangle (page
coordinates)"
      | char s1 s2 s3 y chop
   [s3 ← ParagraphScanner new of: self to: strm style: style.
   s3 init in: r.  y ← r corner y.
   s1 ← s3 copy. s2 ← s3 copy.
   chop ← [alignment=1⇒ [0] alignment].
   while⦂ (char ← s3 scan) do⦂
      [char ≡ true ⇒                 "Exceeded max width"
         [[s2 position = s1 position ⇒                  "No blanks in line"
            [y ← s3 printfrom: s1 align: 0 backup: 1. s3 backup. s3 copyto: s2]
          y ← s2 printfrom: s1 align: alignment backup: 1].
         y ⇒
            [s2 init in: (r origin rect: r corner x ⊙ y). s2 copyto: s1. s2 copyto:
s3]
         ⇑ self copy: s1 position + 1 to: text length]
      char = 040 ⇒
         [s3 copyto: s2. s3 space]
      char = 015 ⇒
         [y ← s3 printfrom: s1 align: chop backup: 1 ⇒
            [s3 init in: (r origin rect: r corner x ⊙ y). s3 copyto: s1. s3 copyto:
s2]
         ⇑ self copy: s1 position + 1 to: text length]
      char = 011 ⇒
         [s3 tab]
      "user notify: 'unimplemented control char'"]
   "Put out trailing text if any"
   s3 width=0⇒ [⇑y]
   y ← s3 printfrom: s1 align: chop backup: 0 ⇒ [⇑y]
   ⇑ self copy: s1 position + 1 to: text length]

⌟

ParagraphScanner asFollows⌟
```

JSW

JSW

Scanning

```
scan "Scan up to a zero-width character"
        | char w maxw cd len pos
    [maxw ← rect width asInteger.
    [textstrm end ⇒ []
     ascent ← ascent max: font ascent.
     descent ← descent min: font descent.
    ].
    while⦂ (char ← textstrm next) do⦂
        [w ← font widthof: char.
        w = 0 ⇒ [⇑char]
        width ← width + w.
        width > maxw ⇒ [⇑true]].
    while⦂ (len  ← runstrm next) do⦂
        [cd ← runstrm next.
        len = 0 ⇒ []                "Go to next run"
        pos ← textstrm position.
        textstrm of: para text from: pos+1 to: pos+len.
        font ← press codefont: cd style: style.
        ⇑self scan].
    ⇑false]
⌋
```

✓SW

PressFile asFollows⌋

Bitmap support

```
bitmap: rect bits: bits | r dlpos w w1 h len "w1 is for Swinehart"
    [boundbox ← boundbox max: (r ← self transrect: rect).
    dlpos ← file wordpos.
        [dlpos*2=file position⇒[]
        self skipchars: 1. file positioneven: 0. dlpos←dlpos+1].
    w ← rect width. h ← rect height.
    self setcodingdots: (w1 ← w+15|16) lines: h;
        setmode;
        setsizewidth: (scale * w1) height: (scale * h);
        setwindowwidth: w1 height: h;
        dotsfollow.
    [bits⇒[file append: bits]                    "bits supplied"
        rect bitsOntoStream: file].                    "else from screen"
    self setp: r origin;
        showdots: (file wordpos - dlpos)]
```

✓SW

File structure commands

```
close | p i fp
    [   [EL empty⇒[] self closePage].
    "put out font directory, part directory, document directory"
    self fontdir.
    fp ← file pages.
    parts puton: file.
    file padpage.
    p ← file pages.
    file nextwordvector ← (27183, "press password"
        (p + 1) "number of records",
        (parts position / 8) "number of parts",
        fp, (p - fp), "part dir and length"
        ¯1, ¯1, ¯1, "junk"
        1, 1, "first and last copies"
```

✓SW

```
        ('S'◦1 "solid color")).
    for: i from: (10 to: 0177) do: [file nextword← ¯1].
    file append: (file title asBCPL: 52);
        append: (dp0 username asBCPL: 32);
        append: (Date default asString+' '+Time default asString asBCPL: 40);
        padpage; shorten; close]
closePage | padding dllength st
    ["Assumes entity trailer has been put on"
    dllength ← file position - dlstart.
    file nextword ← 0.
    EL puton: file; reset.                                              ✓sw
    padding ← file padpage. st ← dlstart/512.
    parts nextwordvector ← (0, st, (file position/512 - st), padding).
    dlstart ← file position]
entity: box containing: expr | fp1 fp2 EL1 v
    [fp1← file position. EL1← EL position/2.
    v← expr eval.
    EL positioneven: 255.               "word-pad EL with <Nop>"
    file positioneven: 0.               "word-pad DL with 0"
    fp2 ← file position - fp1.                                          ✓sw
    "Put a trailer into the EL"
    EL nextwordvector ← (0, "type and fontset"
        0, fp1 "dlstart relative to DL location in file", 0, fp2,
        800, 1200, "paper origin"
        (box origin x), (box origin y),
        (box width), (box height)).
    EL nextword ← EL position/2 - EL1 + 1.
    ⇑v]
```

Initialization

```
of: f
    [file ← f.  self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.                                   ✓sw
    fontcodes ← Vector new: 0.
    fontdefs ← Vector new: 0.
    dlstart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]
```

Bitmap support

```
screenout: rect scale: scale
    ["puts a bit map image onto the PressFile.  The standard
    scaling is 32 micas per Alto dot.  22 looks better, Dover only
    works with 32"                                                      ✓sw
    user displayoffwhile:
        [self somefont.
        self bitmap: rect bits: false.
        self page.
        self close]]
⌋
```

Textframe asFollows⌋

Printing

```
brightness [⇑255]
hardcopy | p
    [p ← dp0 pressfile: 'frame.press'.
    self hardcopy: p.
    p close]
hardcopy: p | e w
    [p entity: (p transrect: (w← window inset: ¯2)) containing⦂
        [   [ColorPrint⇒[p hue: self hue; saturation: self saturation.
                p showrect: window color: self brightness]].
        self pressSelect: p.
        for⦂ e from: (w minus: window) do⦂
            [p showrect: e color: 0].
        para asParagraph presson: p
            in: (p transrect: ((window inset: 0) width← frame width))
            style: style]]
hue [⇑120]
pressSelect: p          "ignored"
saturation [⇑191]
⌋

UserView asFollows⌋

Display
displayoffwhile⦂ expr | i t v
    [t ← mem◦067.
    for⦂ i from: t to: 58 by: ¯2 do⦂
        [mem◦067 ← i. 1/1/1/1/1 "slow"].
    v ← expr eval.
    for⦂ i from: 58 to: t by: 2 do⦂
        [mem◦067 ← i. 1/1/1/1/1 "slow"].
    ⇑v]
⌋
```

```
PanedWindow asFollows⌋

Window protocol
hardcopy: p | pane
    [self showtitle.
    titleframe hardcopy: p.
    for⦂ pane from: panes do⦂ [pane hardcopy: p]]
⌋

PressFile asFollows⌋

Bitmap support
bitmap: rect bits: bits | r dlpos w w1 h len "w1 is for Swinehart"
    [boundbox ← boundbox max: (r ← self transrect: rect).
    dlpos ← file wordpos.
        [dlpos*2=file position⇒[]
        self skipchars: 1. file positioneven: 0. dlpos←dlpos+1].
    w ← rect width. h ← rect height.
    self setcodingdots: (w1 ← w-15|16) lines: h;
        setmode;
        setsizewidth: (scale * w1) height: (scale * h);
        setwindowwidth: w1 height: h;
        dotsfollow.
    [bits⇒[file append: bits]                    "bits supplied"
        rect bitsOntoStream: file].                      "else from screen"
    self setp: r origin;
        showdots: (file wordpos - dlpos)]

File structure commands
box: rect hue: hue sat: sat bright: bright containing⦂ expr | w r
    [self entity: (self transrect: (w← rect inset: ¯2)) containing⦂
        [for⦂ r from: (w minus: rect) do⦂
            [self showrect: r color: 0].
        [ColorPrint⇒
            [self hue: hue; saturation: sat;
                showrect: rect color: bright; brightness: 0]].
        expr eval]]
⌋

Textframe asFollows⌋

Printing
hardcopy: p | e w
    [p box: window hue: self hue sat: self saturation
        bright: self brightness containing⦂
        [self pressSelect: p.
        p brightness: 0.
        para asParagraph presson: p
            in: (p transrect:
                (((window intersect: frame) inset: 0⊙¯1) width← frame width))
            style: style]]
⌋

Window asFollows⌋
```

✓sw

✓sw

```
Framing
show
   [self outline. growing⇒[]
   self showtitle]
showtitle
   [titleframe put:
      (Paragraph new text: self title runs: titlerun alignment: 0)
      at: frame origin+titleloc.
   titleframe outline]
⌋
```

```
Generator asFollows┘

Services
compile: sourceStream in: class under: category notifying: requestor |
selector
    [user displayoffwhile⦂
        [selector ← self compileIn: class⇒
            [class organization classify: selector under: category]].
    ↑selector]
┘

LargeInteger asFollows┘

Conversion
asInteger | t i
    [bytes length>3⇒[↑self]
    t ← 0.
    for⦂ i from: bytes length to: 1 by: ¯1 do⦂
        [t ← t*0200+(bytes◦i)].
    neg⇒[↑0-t]
    ↑t]
┘
```

✓sw

```
ListPane asFollows┘

Private
fill | i          "Given firstShown, compute lastShown and show me."
    [lastShown ← firstShown-1 + (window extent y-4/self lineheight)
        min: list length+1.
    [self locked⇒
        [i ← (selection-lastShown max: 0) + (selection-firstShown min: 0).
        i≠0⇒ [para←nil. firstShown ← firstShown + i. lastShown ← lastShown
+ i]]].
    (frame ← window inset: 2) width ← 999.
    [para≡nil⇒[self refill]].
    self show]

Pane protocol
hardcopy: p
    [(frame ← window inset: 2) width ← 999.
    self refill. ↑super hardcopy: p]

Private
refill | i s            "Recompute para from list"
    [s ← (String new: 256) asStream.
    for⦂ i from: (firstShown to: lastShown) do⦂
        [   [0<i and⦂ i≤list length⇒ [(list◦i) printon: s]
                self dummy copyto: s].
        s cr].
    para ← s contents asParagraph]
┘

Paragraph asFollows┘
```

Normal access

```
+ c         "Concatenates two paragraphs"
   [c← c asParagraph.
   ⇑Paragraph new
      text: text + c text
      runs: (self runcat: self runs to: c runs)
      alignment: alignment]
```

Manipulation of format runs

```
allBold [self allStyle: 1]
allStyle: val
   [runs← self makerun: text length val: val]
⌟
```

ParagraphEditor asFollows⌟

Public Messages

```
append: text  "append text (a string or paragraph) to my end; don't show"
   [self fintype.
      [oldpara⇒ [] oldpara ← oldpara copy].
    para ← para + text]
⌟
```

ParagraphScanner asFollows⌟

Printing

```
printfrom: ps align: align backup: n "Returns false if goes below bottom"
      | ybot a b p xs sp rs c len tpos ts
   [ybot ← rect corner y - (style lineheight*32).
   ybot < rect origin y ⇒ [⇑false]                     "Won't fit"
   a ← ps position + 1. b ← textstrm position - n.
   a > b ⇒ [⇑ybot]            "No text"
   p ← rect origin x ⊙ (rect corner y - (style maxascent*32)).
   xs ← rect width - width.
   sp ← font space.           "Kludge"
   ts ← tabpos contents asStream.
   [align
      = 2 ⇒ [press setp: xs/2⊙0+p; setspacex: sp];
      = 4 ⇒ [press setp: xs⊙0+p; setspacex: sp].
      press setp: p; setspacex: sp].
   press append: para text∘(a to: b).
   rs ← (para run: a to: b) asStream.
   tpos ← ts next.
   while⦂ (len ← rs next) do⦂
      [c ← rs next land: 0363.                 "Remove underline and strikeout"
      len = 0 ⇒ []
      press selectfont: (press fontindex: c style: style) - 1.
      b ← a+len.
      while⦂ tpos and⦂ tpos<b) do⦂                     "Put out tabs"
         [press showchars: tpos-a;
            skipchars: 1;
            setx: p x + ts next.
         a ← tpos+1. tpos ← ts next].
      [align=1 and⦂ tpos≡false ⇒                  "Reset space width"
         [press setspacex: xs/spaces+sp.
         align ← 0]].
```

✓sw

```
        press showchars: b-a.
        a ← b].
    ⇑ybot]
⌋

PressFile asFollows⌋

Font handling stuff
fontindex: code style: style | ix font n
    ["return index if in font dictionary"
    code ← code land: 0363.                    "Remove underline and strikeout"
        [style=prevstyle⇒
            [ix ← fontcodes find: code.
            ix > 0 ⇒ [⇑ix]]
        fontcodes all← nil. "invalid across style change"
        prevstyle← style].
    n ← code / 16 + 1.
    font ← (WidthTable new
        named: (style fontfamily: n)
        pointsize: (style fontsize: n)
        face: (self pressface: code))
     lookup.
    (ix← fontdefs find: font)>0⇒
        [fontcodes∘ix← code. ⇑ix]
    "add entry to font dictionary"
    fontdefs length=16⇒[user notify: 'too many fonts'. ⇑1]
    fontcodes ← fontcodes, code.
    fontdefs ← fontdefs, font.
    ⇑fontcodes length]
```

✓sw

```
Initialization
of: f
    [file ← f.  self scale: 32.
    EL ← (String new: 1) asStream.
    parts ← (String new: 1) asStream.
    fontcodes ← Vector new: 0.
    fontdefs ← Vector new: 0.
    prevstyle← nil.
    distart ← 0.
    boundbox ← 0⊙0 rect: 0⊙0]
⌋
```

✓sw

```
Textframe asFollows⌋

Printing
hardcopy: p | t w
    [p box: window hue: self hue sat: self saturation
        bright: self brightness containing:
        [self pressSelect: p.
        p brightness: 0.
        w← (window intersect: frame) width← frame width.
        t← self charnearpt: w origin+1.
        ([t>1⇒[para copy: t to: para length] para])
            asParagraph presson: p
            in: (p transrect: w)
            style: style]]
```

```
Image
inset: d          "put some leading inside window"
   [frame← window inset: d]

Initialization
style← st [style← st]
"
   | each.
   MenuStyle setfont: 0 name: 'CREAM10'.
   Menu allInstances notNil transform⦂ each to⦂
      (each rescan).
   MenuPaneStyle setfont: 0 name: 'CREAM10'.
   MenuPane allInstances notNil transform⦂ each to⦂
      (each style← MenuPaneStyle).
   DefaultTextStyle setfont: 0 name: 'CREAM12'.
"
⌋

TextStyle asFollows⌋

Access
lineheight
   [⇑self maxascent + self maxdescent]
maxascent
   [maxascent≡nil⇒[⇑(fonts◦1) word: 6] ⇑maxascent]
maxdescent
   [maxdescent≡nil⇒[⇑(fonts◦1) word: 7] ⇑maxdescent]
⌋
```

✓✓ ~~✓sw~~ no

```
WidthTable asFollows⌋

Initialization
lookup | key font file i
   [key ← name + pointsize asString + (↪('' 'I' 'B' 'BI')◦(face+1)).
   font ← WidthDict lookup: key⇒ [⇑font]
   file ← dp0 file: 'fonts.widths'.
   self fontfrom: file.
   file close.
   for⦂ i from: ↪(011 015 040) do⦂
      [i≥min and: i≤max ⇒ [widths◦(i-min+1) ← 0]].
   WidthDict insert: key with: self.
   ⇑self]
⌋
```

✓ sw

```
ListPane asFollows⌋

Private
scrollBy⦂ expr copying: src into: dest showing: item in: frame direction: n
        | strm final stop pt delay chars locked t
    [strm ← Stream new. chars ← 2*frame width/self lineheight. para ←
String new: chars.
    pt ← dest origin. final ← [n<0⇒ [0] list length+1].
    stop ← [locked←self locked⇒ [0 max: (list length+1 min: (lastShown -
firstShown * n sign + selection))] final].
    while⦂ item≠stop do⦂
        [firstShown ← firstShown + n. lastShown ← lastShown + n. item ←
item + n.
        strm of: para from: 1 to: chars.
        [item≠final⇒ [(list◦item) printon: strm] self dummy copyto: strm].
        strm cr. src blt: pt mode: storing. self show.
        (t← expr eval) abs ≤1⇒ [for⦂ delay to: chars/4 do⦂ [strm myend]. para
← nil. ⇑false]
            t*n<0⇒[⇑false]].
    para ← nil. locked and: stop≠final⇒ [locked flash. ⇑false]]
scrollControl⦂ expr
        | dY onlyFirst butFirst onlyLast butLast x1 x2 y1 y2 y3 y4 k
    ["Selection is highlighted.  Unhighlight it.  Invalidate my saved para if I
scroll.  Then reselect selection, or deselect if it is no longer displayed."
    self compselection. dY ← self lineheight.
    x1 ← window origin x. x2 ← window corner x.
    y1 ← window origin y+2. y4 ← window height-4 |dY + y1. y2←y1+dY.
y3←y4-dY.
    onlyFirst ← x1+2⊙y1 rect: 2000⊙y2. butFirst ← x1⊙y2 rect: x2⊙y4.
    onlyLast ← x1+2⊙y3 rect: 2000⊙y4. butLast ← x1⊙y1 rect: x2⊙y3.
    while⦂ (k←expr eval)≠0 do⦂
        [k>0⇒[self scrollBy⦂ expr eval copying: butFirst into: butLast showing:
lastShown
                in: onlyLast direction: 1⇒[] ⇑self select: selection]
        self scrollBy⦂ expr eval copying: butLast into: butFirst showing:
firstShown
                in: onlyFirst direction: ¯1⇒[] ⇑self select: selection].
    self select: selection]
scrollPos
    [list≡nil⇒[⇑0.0]
    list length=0⇒[⇑0.0]
    ⇑firstShown asFloat/list length]
⌋

Menu asFollows⌋

Initialization
string: str |  i pt tpara
    [[str last≠13⇒[str←str+'
']].        "make sure str ends with CR"
    text ← Textframe new para: (tpara ← str asParagraph)
                frame: (Rectangle new origin: (pt ←  0 ⊙ 0)
                                            corner: 1000 ⊙ 1000)
                style: MenuStyle.
    pt ← text maxx: str length+1.
    text frame growto: pt + (4 ⊙ 0).
    tpara center.
    frame ← text frame inset: ¯2 ⊙ ¯2.
    thisline ← Rectangle new origin: text frame origin
            corner: text frame corner x ⊙ text lineheight]
⌋

ScrollBar asFollows⌋

Scheduling
eachtime | p h t oldCursor
    [(rect has: user mp)=false⇒[⇑false]
    t← rect minus: (rect inset: 0⊙(h← rect width min: rect height/4)).
    oldCursor← Cursor new frompage1.
    while⦂ [rect has: (p← user mp)] do⦂
        [t◦1 has: p⇒
            [DownCursor topage1. while⦂ user redbug do⦂
                [self reposition⦂ [owner value scrollUp: ¯42]]]
        t◦2 has: p⇒
            [UpCursor topage1. while⦂ user redbug do⦂
                [self reposition⦂ [owner value scrollUp: 42]]]
        JumpCursor topage1. while⦂ user redbug do⦂
                [self reshow⦂ [owner value scrollTo: (self position: ((0.0 max:
                    (user mp y-h-rect minY-4) asFloat/(rect height-12-(2*h)))
                    min: 1.0))]]].
    oldCursor topage1]


Image
position: f | t
    [t← rect width min: rect height/4.
    position moveto: rect origin+
            (12⊙(3+t+(f*((rect height-(2*t))-12)))).
    ⇑f]
⌋

Textframe asFollows⌋

Initialization
style← st [style← st]
"
    | each. Smalltalk define: ↱MenuStyle as: DefaultTextStyle copy.
    Smalltalk define: ↱MenuPaneStyle as: DefaultTextStyle copy.
    Menu allInstances notNil transform⦂ each to⦂ (each rescan).
    MenuPane allInstances notNil transform⦂ each to⦂ (each style←
MenuPaneStyle).
    MenuPane allInstances notNil transform⦂ each to⦂ (each style←
MenuPaneStyle).
    DefaultTextStyle setfont: 0 name: 'CREAM12'.
"
⌋

TextStyle asFollows⌋

Initialization
copy | t
    [⇑super copy fonts: fonts copy fontnames: fontnames copy]
```

✓sw

```
fonts: fonts fontnames: fontnames                    ✓sw
⌋

ParagraphEditor asFollows⌋

Private Messages
checklooks | t val mask runvals
    [t ← ↪(248 232 219 233   312 296 283 297
        217 241 226 225 210 209 211 213 262 231
        281 305 290 289 274 273 322 308) find: user kbck.
    t=0⇒[⇑false]
    user kbd.
    t=25⇒[self toBravo]; "ctl-T"
?    =26⇒[self fromBravo]. "ctl-F"
        [oldpara⇒[] oldpara ← para recopy].
    runvals ← ↪(1 2 4 256   ¯1 ¯2 ¯4 256  "ctl-b i - x   B I ¯ X"
        0 16 32 48 64 80 96 112 128 144  "ctl-0 1 ... 9"
        160 176 192 208 224 240).  "ctl-shift-0 1 ... 5"
    [(val← runvals∘t)=256⇒[mask← 0377.  val← 0]                          "reset all"
        val<0⇒[mask← 0-val.  val← 0]                  "reset emphasis"
        val>0 and⦂ val<16⇒[mask← val]                 "set emphasis"
        mask← 0360].               "set font"
    para maskrun: loc1 to: loc2-1 under: mask to: val]

" Set up kbmap for unmapped ctl chars:
    | x. ↪(248 232 219 233   312 296 283 297
        217 241 226 225 210 209 211 213 262 231
        281 305 290 289 274 273 0502 0464) transform⦂ x to⦂ (kbMap∘x← x).
    ParagraphEditor classvars delete: ↪(ctlchars runvals).
"
classInit
?   [bs ← 010. ctlw ← 013. esc ← 020.
    cut ← 127. paste ← 2]
⌋

Rectangle asFollows⌋

Conversion
bitsOntoStream: strm | len rec s
    [rec ← origin rect: origin + (self width ⊙ (16 min: self height)).
    s← rec bitsIntoString all← 0.  len← 0.
    while⦂ rec maxY ≤ corner y do⦂                                    ✓sw
        [rec bitsIntoString: s.
        strm append: s.
        len ← len + (s length / 2).
        rec moveby: 0⊙16].
    rec minY<corner y⇒
        [s← (rec origin rect: corner) bitsIntoString.
?       strm append: s.  ⇑len+s length]
    ⇑len]
⌋

TextStyle asFollows⌋

Fonts
fontfamily: n | s char
```

```
    ["return the family name taken out of fontnames"
    s ← Stream default.
    for⦂ char from: fontnames ◦ n do⦂
        [char isletter or⦂ char=('-'◦1)⇒
            [s next ← char]
        ⇑s contents]]
```

✓sw

```
fontsize: n | s c size
    ["return size from fontname"
    s ← (fontnames ◦ n) asStream.
    until⦂ (c ← s next) isdigit do⦂ [].
    size← c-060.
    while⦂ [c ← s next] do⦂
        [size ← size*10 + (c - 060). ].
    ⇑size]
⌋
```

✓sw

```
PanedWindow asFollows⌋

Window protocol
hardcopy: p | pane w
    [self showtitle.
    for⦂ w from: (titleframe window, titleframe frame) do⦂
        [w width← w width*11/10].
    titleframe hardcopy: p.
    for⦂ pane from: panes do⦂ [panz hardcopy: p]]
⌋

PressFile asFollows⌋

Scaling and Transformations
transpt: p
    [⇑ (p x * scale) asInteger ⊙ (24500 - (p y * scale)) asInteger]
⌋
```

✓sw

XEROX XEROX

EARS

Filename: office.crd.st

Creation Date: April 25, 1978 2:54 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 17 April 1978 at 11:24:52 am.'

Class new title: 'CRD'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows

This class has not yet been commented

Models
reportSchedule
   ["Specify when a full report should
    be made. Return 0 if no reports
    should be scheduled.
    For example, the default is to get
    a report every 24 hours"
    ⇑8*60 "8 hour days"]

Parts
arrivalSchedule  | i v n
   ["State the job, input schedule,
    and initia. assignments (the
    names of stations the job will
    visit). For example"

   "number of offices"
   n ← 1.

   "offices to send jobs to"
   v ← Vector new: n.
   for⦂ i to: n do⦂ [
      v◦i ← 'Office' + i asString].

   "how to compute reasonable parameters for this simulation.
   e.g. 1 Xerox 9200 handling
   6 jobs (jobs/office * # of offices)/ hour
      (each with 10 originals and
      80 copies/original (run length))
   with 70% utilization

   ((15.0 +
      (1.0 * originals) +
      (0.5 * originals * runlengths))
   * jobs per hour)/36.0 =  ?% utilization

   ((15.0 +
      (1.0 * 10) +
      (0.5 * 10 * 80))
   * 6)/36.0 =  70.833333% utilization

   then set parameters in
   PaperJob setFeature,
   X9200 serviceTime:
   CRD arrivalSchedule."
```

```
    "create 2 jobs per office per hour and randomly distribute these among
all offices"
    self use: PaperJob new
        startAt: 0
        schedule: (Inputschedule new
            constant: 60.0/(2.0*n))
        assignments: (Inputschedule new
            data: v)]
layout | var workers i r cor boxsize csize in out room org mailclerk n
    ["Create the stations with workers."

    "number of offices"
    n ← 1.

    "width of corridor"
    cor ← 9.
    "left and top edges"
    org ← cor⊙cor.

    "station naming in
        CRD arrivalSchedule,
            layout
        OfficeCopyTask atExit:
        MailClerk taskSchedule.
    naming conventions:
        MailRoom, Copier,
        CopierInbox, CopierOutBox,
        Inbox1, Office1, Outbox1, ..."
    in ← 'Inbox'.
    out ← 'Outbox'.

    "Provide space for the station."
    var ← Station init
        name: 'MailRoom'
        of: (r ← Rectangle new
            origin: org
            corner: (67+ org x)⊙390).

    "Get the workers."
    mailclerk ← MailClerk init id: 1.
    var with: mailclerk.

    "Now construct the station."
    self constructStation: var.

    "tell the mail clerk where to go"
    mailclerk setTaskSchedule.

    "size of copy stations"
    csize ← 133 ⊙ 120.

    for⦂ i from: 0 to: n do⦂ [
        [i=0⇒ [
            room ← 'Copier'.
            boxsize ← csize]
        room ←  i asString.
        var← 390 - (2*cor+ csize y).
    boxsize ←
        ((csize x-cor)/2)⊙[
        n≤ 2⇒ [var]
        "up to 4 offices" (var-cor)/2]].

        "in box"
        var ← MailBox init
            name: [
                i=0⇒ [room + in]
                in + room]
            of: (r ← Rectangle new
                origin: (r corner x + cor) ⊙
                    r origin y
                extent: boxsize).
        "tell box to expect mail clerk"
        var use: mailclerk.
        self constructStation: var.

        "copier or office"
        var ← ([i=0⇒ [CopierRoom]
            OfficeCopyTask]) init.
        var name: [
                i=0⇒ [room]
                'Office' + room]
            of: (r ← Rectangle new
                origin: (r corner x + cor) ⊙
                    r origin y
                extent: boxsize).
        var with: [
            i=0⇒ [X9200 init id: 1]
            OfficeWorker init id: 1].
        self constructStation: var.

        "out box"
        var ← MailBox init
            name: [
                i=0⇒ [room + out]
                out + room]
            of: (r ← Rectangle new
                origin: (r corner x + cor) ⊙
                    r origin y
                extent: boxsize).
        var use: mailclerk.
        self constructStation: var.

        i\2=0⇒ [
            "for row wraparound"
            r ← Rectangle new
                origin: (r corner x - (([i=0⇒ [3] 6]) * (boxsize x + cor))) ⊙ (r
corner y + cor)
                extent: 0⊙0]
]]
⌋
SystemOrganization classify: ↪CRD under: 'Office'.⌋
```

```
Class new title: 'CopierRoom'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⅃
```

This class has not yet been commented

Models

Parts

```
atExit: job |
   ["Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'. No code
    means that the job is completed"


   "change job so that office knows it's done"
   job feature◦3 ← 'copied']
⅃
SystemOrganization classify: ↪CcpierRoom under: 'Office'.⅃
```

```
Class new title: 'MailBox'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

As yet unclassified
⌋
SystemOrganization classify: ↪MailBox under: 'Office'.⌋
```

```
Class new title: 'OfficeCopyTask'
    subclassof: Station
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Models

Parts

```
atExit: job | in out copier mailroom room
    ["Modification of the list of stations
     that the job is to visit. Example:
     job addTask: 'Station'.  No code
     means that the job is completed"

    (job feature∘3) = 'copied'⇒ [
        "job finished"]

    out ← 'Outbox'.
    in ← 'Inbox'.
    copier ← 'Copier'.
    mailroom ← 'MailRoom'.
    room ← self name last inString.

    "route the job"
    job appendTasks:
        ( out + room ),
        ( mailroom ),
        ( copier + in ),
        ( copier ),
        ( copier + out ),
        ( mailroom ),
        ( in + room ),
        ( self name )]
⌋
SystemOrganization classify: ↪OfficeCopyTask under: 'Office'.⌋
```

```
Class new title: 'MailClerk'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Models

```
picture
    ["Name of the picture representing
     the worker. The default is a small
     rectangular shape."
     ⇑ 'man']
serviceTime: job
    ["The time the worker spends
     giving service to the job. It is
     possible that job is a set of jobs.
     Time might a function of a
     job property."

    ⇑0.1]
```

Parts

```
setTaskSchedule | m n
    ["Initialize the worker's travelling
      schedule"

    "number of offices"
    n ← 1.

    "number of minutes for a
    complete mail pickup/distribution,
    e.g. 4 times/8 hr day = 120 min"
    m ← 30.

    (station layout)
        schedule: self
        todo: ↪taskSchedule
        after: m asFloat/
            (2 * (n + 1) + 1)]
speed
    ["Number of display bits per travel
     time"
     ⇑24]
taskSchedule | task v in out room stops n
    ["It is possible that the worker can
     service more than one station. At
     that time, the worker should be sent
     the message setTaskSchedule in order
     to initialize the schedule. Example:
     worker changes stations each hour.
     Ordered list of stations is given here!"

    "number of offices"
```

```
        n ← 1.

        [tasksToDo empty⇒ [
            stops ← 2 "in + out" * (n + 1 "copier") +
                1 "mail room".

            v ← Stream new of: (Vector new: stops).
            "construct mail clerk itinerary"
            in ← 'Inbox'.
            out ← 'Outbox'.
            room ← 'Copier'.
            v next ← room + in.
            v next ← room + out.
            for⦂ task to: n do⦂ [
                room ← task asString.
                v next ← in + room.
                v next ← out + room].
            v next ← 'MailRoom'.
            self setTasks: v asArray]].

        [tasksToDo end⇒ [tasksToDo reset]].

        task ← tasksToDo next.
        (station layout)
            schedule: self
            todo: ↪travelto:, task
            after: 0.
        self setTaskSchedule ]
    travelTime
        "The amount of time it takes the
         worker to travel one display bit if its
         travel speed=1."

        [ ⇑ 0.05]
    ⌋
    SystemOrganization classify: ↪MailClerk under: 'Office'.⌋
```

```
Class new title: 'OfficeWorker'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

```
Models
picture
   ["Name of the picture representing
    the worker. The default is a small
    rectangular shape."
   ⇑(Inputschedule new
      data: 'man', 'lady') next]
serviceTime: job
   ["The time the worker spends
    giving service to the job. It is
    possible that job is a set of jobs.
    Time might a function of a
    job property."
    ⇑0.1]
⌟
SystemOrganization classify: ↪OfficeWorker under: 'Office'.⌟
```

```
Class new title: 'X9200'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows┘
```

This class has not yet been commented

Parts

```
serviceTime: job
"The time the copier spends on a single job. Job-specific information is
combined with known setup time (0.3 min. for entire job, plus 0.13 min for
each page of original) and machine performance rate (0.0083 min per sheet
of output)."
   [⇑0.3 +
   (0.13 * (job feature◦1)) +
   (0.0083 * (job feature◦1) *
      (job feature◦2))]
┘
SystemOrganization classify: ↪X9200 under: 'Office'.┘
```

```
Class new title: 'PaperJob'
   subclassof: Job
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

```
Models
picture
   ["The name of the picture for the
    job. The d:fault is a square
    shape."
    ⇑'papers']
speed
   ["The number of display bits per
    travel time"
    ⇑1000]
travelTime
   ["The amount of time it takes the
    job to travel one display bit if it
    travel speed=1."
    ⇑0.01]

Parts
setFeature
   ["Returns a descriptive feature of the job"

    "number of originals and
   run length (number of pages each)"

   ⇑(Inputschedule new
      constant: 10) next,
   (Inputschedule new
      constant: 80) next,
   'copy']
⌟
SystemOrganization classify: ↪PaperJob under: 'Office'.⌟
```

XEROX XEROX

EARS

Filename: filin-factory.st,

Creation Date: April 25, 1978 3:14 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.2n on 29 March 1978 at 12:49:02 pm.'⌋

(dp0 file: 'factory.st') filin.
(dp0 file: 'PCBoard.st') filin.

SystemOrganization classify: ↪Setup alsounder: 'Worker'.⌋
SystemOrganization classify: ↪Stuff alsounder: 'Worker'.⌋
SystemOrganization classify: ↪Inspect alsounder: 'Worker'.⌋
SystemOrganization classify: ↪Solder alsounder: 'Worker'.⌋
SystemOrganization classify: ↪Test alsounder: 'Worker'.⌋
SystemOrganization classify: ↪Pack alsounder: 'Worker'.⌋

SystemOrganization classify: ↪SettingUp alsounder: 'Station'.⌋
SystemOrganization classify: ↪Stuffing alsounder: 'Station'.⌋
SystemOrganization classify: ↪Inspecting alsounder: 'Station'.⌋
SystemOrganization classify: ↪Soldering alsounder: 'Station'.⌋
SystemOrganization classify: ↪Testing alsounder: 'Station'.⌋
SystemOrganization classify: ↪Packing alsounder: 'Station'.⌋

SystemOrganization classify: ↪PCBoard alsounder: 'Job'.⌋
SystemOrganization classify: ↪Versatec alsounder: 'Simulation'.⌋
```

XEROX XEROX

EARS

Filename: pcboard.st.

Creation Date: April 25, 1978 3:13 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.2n on 30 March 1978 at 4:06:05 pm.'⌋


PictureOrganization insert: 'PCBoard' with: (Traveller new image: ↱(255
255 255 255 128 0 63 255 128 3 191 255 128 2 191 255 128 3 191 255 128 1 63
255 128 1 63 255 128 1 63 255 128 1 63 255 159 255 63 255 144 4 63 255 144 4
63 255 185 196 63 255 169 124 63 255 185 192 63 255 128 0 63 255 255 255 255
255 ) hull: ↱(255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 ) bitsPerLine: 19).⌋
```

XEROX XEROX

EARS

Filename: factory.st,

Creation Date: April 25, 1978 3:13 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 7 April 1978 at 1:53:13 pm.'⌋

Class new title: 'Inspecting'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
atExit: job | p
"Modification of the list of stations that the job is to visit. Example: job
addTask: 'Station'.  No code means that the job is completed."
   [job addTask: 'Soldering'.
   "move job into the corridor"
   job moveby: 0⌾25.]

⌋
SystemOrganization classify: ↪Inspecting under: 'Factory'.⌋
```

```
Class new title: 'Inspect'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌟

This class has not yet been commented

Parts
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."
     [⇑ 'man']
serviceTime: job
    "For assembly line workers,
     the service time depends on
     the type of job, and is
     proportional to the number of
     components per board.  This is
     given by the job's feature."
     [⇑0.036*job feature]
⌟
SystemOrganization classify: ↪Inspect under: 'Factory'.⌟
```

```
Class new title: 'Packing'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

```
Parts
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"

   [ ]
⌟
SystemOrganization classify: ↪Packing under: 'Factory'.⌟
```

```
Class new title: 'Pack'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
   [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
   [⇑0.006*job feature]
⌋
SystemOrganization classify: ↪Pack under: 'Factory'.⌋
```

```
Class new title: 'PCBoard'
   subclassof: Job
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "The name of the picture for the
    job. The default is a square
    shape."
    [⇑'PCBoard']
speed
   "The number of display bits per
    travel time"
    [⇑8]
setFeature
   "return the value for my feature"
    [⇑33]
travelTime
   "The amount of time it takes the
    job to travel one display bit if it
    travel speed=1."
    [⇑0.0]
⌋
SystemOrganization classify: ☞PCBoard under: 'Factory'.⌋
```

```
Class new title: 'SettingUp'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"
    [job addTask: 'Stuffing'.]
⌋
SystemOrganization classify: ⤴SettingUp under: 'Factory'.⌋
```

```
Class new title: 'Setup'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
    [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportion il to the number of
    components per board.  This is
    given by the job's feature."
    [⇑0.086*job feature]
⌋
SystemOrganization classify: ⇗Setup under: 'Factory'.⌋
```

```
Class new title: 'Soldering'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

```
Parts
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"
   [job addTask: 'Testing'.]
⌋
SystemOrganization classify: ↪Soldering under: 'Factory'.⌋
```

```
Class new title: 'Solder'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

Parts

```
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
   [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
   [⇑0.176*job feature]
⌟
SystemOrganization classify: ⤻Solder under: 'Factory'.⌟
```

```
Class new title: 'Stuffing'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

```
Parts
atExit: job |
    "Modification of the list of stations
     that the job is to visit. Example:
     job addTask: 'Station'.  No code
     means that the job is completed"
    [job addTask: 'Inspecting'.]
⌋
SystemOrganization classify: ↪Stuffing under: 'Factory'.⌋
```

```
Class new title: 'Stuff'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."
     [⇑ 'man']
serviceTime: job
    [⇑0.336*job feature]
⌋
SystemOrganization classify: ↪Stuff under: 'Factory'.⌋
```

```
Class new title: 'Testing'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

Parts

```
atExit: job |
"Modification of the list of stations that the job is to visit. Example: job
addTask: 'Station'.  No code means that the job is completed."
   [job addTask: 'Packing']


⌟
SystemOrganization classify: ↪Testing under: 'Factory'.⌟
```

```
Class new title: 'Test'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
    [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
    [⇑0.086*job feature]
⌋
SystemOrganization classify: ↪Test under: 'Factory'.⌋
```

```
Class new title: 'Versatec'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
arrivalSchedule
   "Give the input schedule for new jobs."
   [self use: PCBoard new
      startAt: 0
      schedule: (Inputschedule new constant: 4)
      assignments: 'SettingUp'.]
layout | station1 station2 station3 station4 station5 station6 station7
   ["Create the stations"
    station1 ← SettingUp init
       of: (30⊙5 rect: 150⊙145).
    station2 ←  Stuffing init
       of: (180⊙5 rect: 320⊙145).
    station3 ← Inspecting init
       of: (350⊙5 rect: 470⊙145).
    station4 ← Soldering init
       of: (30⊙180 rect: 170⊙320).
    station5 ← Testing init
       of: (200⊙180 rect: 320⊙320).
    station6 ← Packing init
       of: (350⊙180 rect: 470⊙320).
    "Get the workers."
    station1 with:
       (Setup init id: 1),
       (Setup init id: 2).
    station2 with:
       (Stuff init id: 1),
       (Stuff init id: 2),
       (Stuff init id: 3),
       (Stuff init id: 4),
       (Stuff init id: 5),
       (Stuff init id: 6).
    station3 with:
       (Inspect init id: 1).
    station4 with:
       (Solder init id: 1),
       (Solder init id: 2),
       (Solder init id: 3),
       (Solder init id: 4).
    station5 with:
       (Test init id: 1),
       (Test init id: 2).
    station6 with:
       (Pack init id: 1).
    "Now construct the stations"
    self constructStation: station1.
    self constructStation: station2.
    self constructStation: station3.
```

```
        self constructStation: station4.
        self constructStation: station5.
        self constructStation: station6]
⌋
SystemOrganization classify: ⤴Versatec under: 'Factory'.⌋
```

XEROX XEROX

EARS

Filename: factory-final.st

Creation Date: April 25, 1978 3:16 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 7 April 1978 at 1:53:13 pm.'⌋
```

"Inspecting"

```
Class new title: 'Inspecting'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
atExit: job | p
"Modification of the list of stations that the job is to visit. Example: job
addTask: 'Station'.  No code means that the job is completed.

At the inspect station, there is a 11% chance that the board is bad.  If so,
go back to the Stuffing station.  Otherwise go on to the Soldering station."
   [(Bernoulli new p: 0.11) random=1⇒
      [job addTask: 'Stuffing']
   job addTask: 'Soldering'.
   "move job into the corridor"
   job moveby: 0⊙25.]

⌋
SystemOrganization classify: ↪Inspecting under: 'Factory'.⌋
```

```
Class new title: 'Inspect'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌟

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
   [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
   [⇑0.036*job feature]
⌟
SystemOrganization classify: ↪Inspect under: 'Factory'.⌟
```

```
Class new title: 'Packing'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

Parts

```
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"

   [ ]
⌟
SystemOrganization classify: ↱Packing under: 'Factory'.⌟
```

```
Class new title: 'Pack'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
    [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
    [⇑0.006*job feature]
⌋
SystemOrganization classify: ↱Pack under: 'Factory'.⌋
```

```
Class new title: 'PCBoard'
   subclassof: Job
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "The name of the picture for the
    job. The default is a square
    shape."
    [⇑'PCBoard']
speed
   "The number of display bits per
    travel time"
    [⇑8]
setFeature
   "The feature for a printed circuit
    board gives the number of
    components on the board.  Choose
    this from a sample space of
    typical values."
    [⇑(Inputschedule new data:
      ⤻(25 25 33 33 33 33 33 50 66 66 66))
      next]
travelTime
   "The amount of time it takes the
    job to travel one display bit if it
    travel speed=1."
    [⇑0.0]
⌋
SystemOrganization classify: ⤻PCBoard under: 'Factory'.⌋
```

```
Class new title: 'SettingUp'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"
    [job addTask: 'Stuffing'.]
⌋
SystemOrganization classify: ☞SettingUp under: 'Factory'.⌋
```

```
Class new title: 'Setup'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
    [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
    [⇑0.086*job feature]
⌋
SystemOrganization classify: ↪Setup under: 'Factory'.⌋
```

```
Class new title: 'Reworking'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

Parts

```
atExit: job |
"Modification of the list of stations that the job is to visit. Example: job
addTask: 'Station'.  No code means that the job is completed."
   [job addTask: 'Testing']

⌟
SystemOrganization classify: ↱Reworking under: 'Factory'.⌟
```

```
Class new title: 'Rework'
   subclassof: Worker
   fields: ”
   declare: ”;
   asFollows⌟

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
    [⇑ 'man']
serviceTime: job
   "It takes between 5 and 15 minutes to rework a board"
    [⇑(Inputschedule new uniform: 5 to: 15) next]
⌟
SystemOrganization classify: ↪Rework under: 'Factory'.⌟
```

```
Class new title: 'Soldering'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

Parts

```
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example: .
    job addTask: 'Station'.  No code
    means that the job is completed"
    [job addTask: 'Testing'.]
⌟
SystemOrganization classify: ↪Soldering under: 'Factory'.⌟
```

```
Class new title: 'Solder'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
    [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
    [⇑0.176*job feature]
⌋
SystemOrganization classify: ↪Solder under: 'Factory'.⌋
```

```
Class new title: 'Stuffing'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
   "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"
   [job addTask: 'Inspecting'.]
⌋
SystemOrganization classify: ↪Stuffing under: 'Factory'.⌋
```

```
Class new title: 'Stuff'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Parts

```
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."
     [⇑ 'man']
serviceTime: job
"If the job has only been to the SettingUp and Stuffing stations, then
compute the service time by multiplying the total number of components
by 0.336.  Otherise, the job is returning from the Inspecting station because
of mistakes.  In this case, there will on the average be only one or two bad
components, so choose the service time accordingly."
   [job completedTasks=
              'SettingUp Stuffing '⇒
      [⇑0.336*job feature]
    ⇑0.336* (Inputschedule new uniform: 1 to: 2) next]
⌋
SystemOrganization classify: ↪Stuff under: 'Factory'.⌋
```

```
Class new title: 'Testing'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows┘
```

This class has not yet been commented

Parts

```
atExit: job |
"Modification of the list of stations that the job is to visit. Example: job
addTask: 'Station'.  No code means that the job is completed.

There is a 16% chance that a board will fail testing.  If so, it goes to the
Reworking station.  Otherwise, it goes on to the Packing sation."
   [(Bernoulli new p: 0.16) random=1⇒
      [job addTask: 'Reworking']
   job addTask: 'Packing']


┘
SystemOrganization classify: ↪Testing under: 'Factory'.┘
```

```
Class new title: 'Test'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."
   [⇑ 'man']
serviceTime: job
   "For assembly line workers,
    the service time depends on
    the type of job, and is
    proportional to the number of
    components per board.  This is
    given by the job's feature."
   [⇑0.086*job feature]
⌋
SystemOrganization classify: ↪Test under: 'Factory'.⌋
```

```
Class new title: 'Versatec'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows_|
```

This class has not yet been commented

Parts

```
arrivalSchedule
   "Give the input schedule for new jobs."
   [self use: PCBoard new
      startAt: 0
      schedule: (Inputschedule new constant: 4)
      assignments: 'SettingUp'.]
layout | station1 station2 station3 station4 station5 station6 station7
   ["Create the stations"
    station1 ← SettingUp init
       of: (30⊙5 rect: 150⊙145).
    station2 ←  Stuffing init
       of: (180⊙5 rect: 320⊙145).
    station3 ← Inspecting init
       of: (350⊙5 rect: 470⊙145).
    station4 ← Soldering init
       of: (30⊙180 rect: 170⊙320).
    station5 ← Testing init
       of: (200⊙180 rect: 320⊙320).
    station6 ← Packing init
       of: (350⊙180 rect: 470⊙320).
    station7 ← Reworking init
       of: (103⊙339 rect: 369⊙432).
    "Get the workers."
    station1 with:
       (Setup init id: 1),
       (Setup init id: 2).
    station2 with:
       (Stuff init id: 1),
       (Stuff init id: 2),
       (Stuff init id: 3),
       (Stuff init id: 4),
       (Stuff init id: 5),
       (Stuff init id: 6).
    station3 with:
       (Inspect init id: 1).
    station4 with:
       (Solder init id: 1),
       (Solder init id: 2),
       (Solder init id: 3),
       (Solder init id: 4).
    station5 with:
       (Test init id: 1),
       (Test init id: 2).
    station6 with:
       (Pack init id: 1).
    station7 with:
       (Rework init id: 1).
```

```
        "Now construct the stations"
        self constructStation: station1.
        self constructStation: station2.
        self constructStation: station3.
        self constructStation: station4.
        self constructStation: station5.
        self constructStation: station6.
        self constructStation: station7]
⌋
SystemOrganization classify: ↪Versatec under: 'Factory'.⌋
```

XEROX XEROX

EARS

Filename: dover.org,

Creation Date: April 25, 1978 3:27 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'fix SystemOrganization'.
SystemOrganization classify: o(Printing Dover DoverRoom Memo)
    under: 'Office'.

SystemOrganization classify: oPrinting
    alsounder: 'Simulation'.
SystemOrganization classify: oDover
    alsounder: 'Worker'.
SystemOrganization classify: oDoverRoom
    alsounder: 'Station'.
SystemOrganization classify: oMemo
    alsounder: 'Job'.~
```

XEROX XEROX

EARS

Filename: dover.st,

Creation Date: April 25, 1978 3:26 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 7 April 1978 at 10:29:01 am.'⌋

Class new title: 'Dover'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋

This class has not yet been commented

Parts
serviceTime: job
    "The time the worker spends
    giving service to the job.  It is
    possible that job is a set of jobs.
    Time might a function of the
    job feature."

    [ ⇑ 2.5 * job feature]
speed
    "Number of display bits per travel
    time"

    [ ⇑ 100]
travelTime
    "The amount of time it takes the
    worker to travel one display bit if its
    travel speed=1."

    [ ⇑ 0.0]
⌋
SystemOrganization classify: ↪Dover under: 'Office'.⌋
```

```
Class new title: 'DoverRoom'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Model

Parts

```
atEntrance: job  | newjob num
   ["The time the job waits at the
    door before getting attention.
    Must return a number."
    [self name = 'DoverCity' ⇒
       [newjob ← Memo init
       id: job name + '*' in: layout.
       newjob feature: job feature.
       num ← job name asInteger \3 +1.
       newjob place: (241⊙104), (239⊙229), (242⊙373) ◦num.
       newjob startInMiddle: 'Clover', 'Grover', 'Rover' ◦num]].
   ⇑0]
⌋
SystemOrganization classify: ↪DoverRoom under: 'Office'.⌋
```

```
Class new title: 'Memo'
    subclassof: Job
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

Parts

```
picture
    "The name of the picture for the
    job. The default is a square
    shape."

    [ ⇑ 'papers']
setFeature
    "Returns a descriptive property
    of the job"

    [ ⇑ (Inputschedule new uniform: 1 to: 20) next]
speed
    "The number of display bits per
    travel time"

    [ ⇑ 100]
travelTime
    "The amount of time it takes the
    job to travel one display bit if it
    travel speed=1."

    [ ⇑ 0.0]
⌋
SystemOrganization classify: ↪Memo under: 'Office'.⌋
```

```
Class new title: 'Printing'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Model

Parts

```
arrivalSchedule
   ["State the job, input schedule,
   and initial assignments (the
   names of stations the job will
   visit). For example"
      self use: Memo new
            startAt: 0
            schedule:
              (Inputschedule new
                          geometric: 0.07)
            assignments: 'DoverCity'.]

layout | var workers
   ["Create the stations with workers."

     "Provide space for the station."
     var ←
        DoverRoom init name: 'DoverCity'
         of: (49⊙25 rect: 247⊙201).
     "Get the workers."
     var with:
            (Dover init id: 1),
            (Dover init id: 2),
            (Dover init id: 3).
     "Now construct the station."
     self constructStation: var.
     var ←
        DoverRoom init name: 'Clover'
         of: (282⊙9 rect: 497⊙122).
     "Get the workers."
     var with:
            (Dover init id: 4).
     "Now construct the station."
     self constructStation: var.
     var ←
        DoverRoom init name: 'Grover'
         of: (281⊙130 rect: 497⊙250).
     "Get the workers."
     var with:
            (Dover init id: 5).
     "Now construct the station."
     self constructStation: var.
     var ←
        DoverRoom init name: 'Rover'
```

```
		of: (280⊙258 rect: 498⊙388).
	"Get the workers."
	var with:
			(Dover init id: 6).
	"Now construct the station."
	self constructStation: var.]

┘
SystemOrganization classify: ↪Printing under: 'Office'.┘
```

EARS

Filename: tcp.org,

Creation Date: April 25, 1978 3:55 PM

Printed by: Weyer, Steve

```
'From Smalltalk 5.3b/xm on 9 April 1978 at 11:02:48 am.'┘

SystemOrganization classify: ↪Accounting alsounder: 'Station'.┘
SystemOrganization classify: ↪AccountsPayable alsounder: 'Station'.┘
SystemOrganization classify: ↪AClerk alsounder: 'Worker'.┘
SystemOrganization classify: ↪Boss alsounder: 'Worker'.┘
SystemOrganization classify: ↪CPA alsounder: 'Worker'.┘
SystemOrganization classify: ↪DClerk alsounder: 'Worker'.┘
SystemOrganization classify: ↪Disbursements alsounder: 'Station'.┘
SystemOrganization classify: ↪Ledger alsounder: 'Job'.┘
SystemOrganization classify: ↪PayDay alsounder: 'Simulation'.┘
SystemOrganization classify: ↪PaySlips alsounder: 'Job'.┘
SystemOrganization classify: ↪Projection alsounder: 'Job'.┘
SystemOrganization classify: ↪RClerk alsounder: 'Worker'.┘
SystemOrganization classify: ↪Records alsounder: 'Station'.┘
SystemOrganization classify: ↪Sorter alsounder: 'Worker'.┘
SystemOrganization classify: ↪SupervisorOffice alsounder: 'Station'.┘
SystemOrganization classify: ↪TimeCards alsounder: 'Job'.┘
SystemOrganization classify: ↪WorkArea alsounder: 'Station'.┘
```

XEROX XEROX

EARS

Filename: tcp.st.

Creation Date: April 25, 1978 3:54 PM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 9 April 1978 at 11:02:48 am.'⌋

Class new title: 'Accounting'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
atExit: job
    "Modification of the list of stations"

   [(Projection init
       id: job name in: layout)
     place: stationRect corner;
     startInMiddle: 'SupervisorOffice']
⌋
SystemOrganization classify: ↪Accounting under: 'Office'.⌋
```

```
Class new title: 'AccountsPayable'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
   "Modification of the list of stations
   that the job is to visit. Example:
   job addTask: 'Station'.  No code
   means that the job is completed"

   [job moveby: 0⊙16;
      addTask: 'Disbursements']
⌋
SystemOrganization classify: ↱AcccuntsPayable under: 'Office'.⌋
```

```
Class new title: 'AClerk'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."

   [ ⇑ 'man']
serviceTime: job
    "The time the worker spends
     giving service to the job.  It is
     possible that job is a set of jobs.
     Time might a function of the
     job feature."

   [ ⇑ 1.5 ]
⌋
SystemOrganization classify: ↪AClerk under: 'Office'.⌋
```

```
Class new title: 'Boss'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

```
Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."

   [ ⇑ 'lady']
serviceTime: job
   "The time the worker spends
    giving service to the job.  It is
    possible that job is a set of jobs.
    Time might a function of the
    job feature."

   [ job is: Projection⊃[ ⇑ 2.5 ]
     ⇑ 1.5]
⌟
SystemOrganization classify: ↪Boss under: 'Office'.⌟
```

```
Class new title: 'CPA'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌟
```

This class has not yet been commented

```
Parts
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."

    [ ⇑ 'man']
serviceTime: job
    "The time the worker spends
     giving service to the job.  It is
     possible that job is a set of jobs.
     Time might a function of the
     job feature."

    [ ⇑ 4.0]
⌟
SystemOrganization classify: ↪CPA under: 'Office'.⌟
```

```
Class new title: 'DClerk'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."

   [ ⇑ 'lady']
serviceTime: job
   "The time the worker spends
   giving service to the job.  It is
   possible that job is a set of jobs.
    Time might a function of the
    job feature."

   [ ⇑ 2.0]
⌋
SystemOrganization classify: ↪DClerk under: 'Office'.⌋
```

```
Class new title: 'Disbursements'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job | newJob y
  [job moveby: 0⊙32;
      addTask: 'Records'.
   newJob ← PaySlips init.
   newJob id: job name in: layout.
   newJob sta⁻tInMiddle: 'WorkArea'.
   newJob place: stationRect corner-
                 newJob extent.
   y ← 16.
   until⦂
      (newJob stepby: (20⊙y)
         wrt: layout corridor)
      do⦂ [y←y+32].]
⌋
SystemOrganization classify: ↪Disbursements under: 'Office'.⌋
```

```
Class new title: 'Ledger'
   subclassof: Job
   fields: ''
   declare: '';
   asFollows┘

This class has not yet been commented

Parts
picture
    "The name of the picture for the
    job. The default is a square
    shape."

   [ ⇑ 'ledger']
speed
    "The number of display bits per
    travel time"

   [ ⇑ 30 ]
┘
SystemOrganization classify: ↪Ledger under: 'Office'.┘
```

```
Class new title: 'PayDay'
   subclassof: Simulation
   fields: ''
   declare: '';
   asFollows⌟
```

This class has not yet been commented

Parts

```
arrivalSchedule
    "State the job, input schedule,
    and initial assignments (the
    names of stations the job will
    visit). For example"

   [ self use: TimeCards new
         startAt: 0
         schedule:
           (Inputschedule new
                    data: 1,1,1,2,3)
         assignments: 'WorkArea'.]
layout | worker
   [self constructStation:
      ((WorkArea init
         of: (35⊙8 rect: 140⊙125))
      with: (Sorter init id: 1)).
    self constructStation:
      ((SupervisorOffice init
         of: (185⊙8 rect: 290⊙125))
      with: (Boss init id: 1)).
    self constructStation:
      ((Accounting init
         of: (335⊙8 rect: 440⊙125))
      with: (CPA init id: 1)).
    self constructStation:
      ((Disbursements init
         of: (35⊙165 rect:
              140⊙282))
      with: (DClerk init id: 1)).
    self constructStation:
      ((AccountsPayable init
         of: (185⊙165 rect:
              290⊙282))
      with: (AClerk init id: 1)).
    worker ← RClerk init id: 1.
    self constructStation:
      ((Records init
         of: (335⊙165 rect:
              440⊙282))
      with: worker).
    worker setTaskSchedule]
⌟
SystemOrganization classify: ↱PayDay under: 'Office'.⌟
```

```
Class new title: 'PaySlips'
    subclassof: Job
    fields: ''
    declare: '';
    asFollows⌋

This class has not yet been commented

Parts
picture
    "The name of the picture for the
    job. The default is a square
    shape."

    [ ⇑ 'payslips']
speed
    "The number of display bits per
    travel time"

    [ ⇑ 40 ]
⌋
SystemOrganization classify: ↪PaySlips under: 'Office'.⌋
```

```
Class new title: 'Projection'
   subclassof: Job
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
picture
    "The name of the picture for the
    job. The default is a square
    shape."

   [ ⇑ 'projection' ]
speed
    "The number of display bits per
    travel time"

   [ ⇑ 20 ]
⌋
SystemOrganization classify: ↪Projection under: 'Office'.⌋
```

```
Class new title: 'RClerk'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."

   [ ⇑ 'lady']
serviceTime: job
   "The time the worker spends
   giving service to the job.  It is
   possible that job is a set of jobs.
   Time might a function of the
   job feature."

   [ ⇑ 0.5 ]
setTaskSchedule
   "Initialize the worker's travelling
    schedule. This example has the
    worker move every hour. "

   [ (station layout) schedule: self
                      todo: ↪taskSchedule
                      after: 8. ]
taskSchedule | task

   [(Ledger init
      id: (station time+4 /8) asInteger
      in: station layout)
    place: station exit+(0⊙16);
    startInMiddle: 'Accounting'.
    self setTaskSchedule. ]
⌋
SystemOrganization classify: ↪RClerk under: 'Office'.⌋
```

```
Class new title: 'Records'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

As yet unclassified

```
⌋
SystemOrganization classify: ☞Records under: 'Office'.⌋
```

```
Class new title: 'Sorter'
   subclassof: Worker
   fields: ''
   declare: '';
   asFollows⌋

This class has not yet been commented

Parts
picture
   "Name of the picture representing
    the worker. The default is a small
    rectangular shape."

   [ ⇑ 'man']
serviceTime: job
   "The time the worker spends
    giving service to the job.  It is
    possible that job is a set of jobs.
    Time might a function of the
    job feature."

   [ job is: PaySlips⇒[ ⇑ 1.0 ]
     ⇑ 0.2 ]
⌋
SystemOrganization classify: ↪Sorter under: 'Office'.⌋
```

```
Class new title: 'SupervisorOffice'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
    "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"

   [job is: TimeCards⇒
      [job moveby: 0⊙16;
       addTask: 'AccountsPayable']]
⌋
SystemOrganization classify: ↪SupervisorOffice under: 'Office'.⌋
```

```
Class new title: 'TimeCards'
    subclassof: Job
    fields: ''
    declare: '';
    asFollows┘

This class has not yet been commented

Parts
picture
    "The name of the picture for the
    job. The default is a square
    shape."

   [ ⇑ 'timecards']
speed
    "The number of display bits per
    travel time"

   [ ⇑ 40 ]
┘
SystemOrganization classify: ↪TimeCards under: 'Office'.┘
```

```
Class new title: 'WorkArea'
   subclassof: Station
   fields: ''
   declare: '';
   asFollows⌋
```

This class has not yet been commented

Parts

```
atExit: job |
    "Modification of the list of stations
    that the job is to visit. Example:
    job addTask: 'Station'.  No code
    means that the job is completed"

   [job is: TimeCards⇒
      [job addTask: 'SupervisorOffice'
      ]
   ]
⌋
SystemOrganization classify: ↪WorkArea under: 'Office'.⌋
```

Alto II/Orbit/Dover Press file printer

Spruce version 7.0

File: tcp.pics

Creation date: 25-APR-78 15:56:08

Name: Weyer, Steve

2 total sheets = 1 page, 1 copy.

```
Problems encountered:
Character code 36b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 36b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 17b not found in Font set 0, font 0.
Character code 36b not found in Font set 0, font 0.
... more problems not listed ...
```

```
'From Smalltalk 5.3b/xm on 17 April 1978 at 6:12:11 pm.'
**


PictureOrganization insert: 'ledger' with: (Traveller new image: (255 255 128 1 128 1 128 125 128 1 128
** 125 128 1 128 125 128 1 128 125 128 1 190 125 128 1 190 125 128 1 255 255 ) hull: (255 255 255 255 2
**55 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 25
**5 255 255 ) bitsPerLine: 16).
**

PictureOrganization insert: 'projection' with: (Traveller new image: (255 255 128 1 128 1 128 1 128 1 1
**28 1 128 1 128 3 128 5 128 9 128 17 136 161 149 65 162 1 192 1 255 255 ) hull: (255 255 255 255 255 2
**55 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 25
**5 255 ) bitsPerLine: 16).
**

PictureOrganization insert: 'payslips' with: (Traveller new image: (255 248 63 255 128 14 63 255 128 11
** 191 255 128 10 191 255 128 74 191 255 128 234 191 255 129 90 191 255 129 74 191 255 128 234 191 255
**128 90 191 255 129 90 191 255 128 234 191 255 128 74 191 255 255 250 191 255 63 254 191 255 15 255 19
**1 255 ) hull: (255 248 63 255 255 254 63 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255
**191 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 255 255 255 191 2
**55 255 255 191 255 255 255 191 255 63 255 191 255 15 255 191 255 ) bitsPerLine: 18).
**

PictureOrganization insert: 'timecards' with: (Traveller new image: (255 252 127 255 128 34 127 255 128
** 17 127 255 128 15 255 255 128 10 255 255 128 10 255 255 128 10 255 255 128 10 255 255 255 255 255 25
**5 ) hull: (255 252 127 255 255 254 127 255 255 255 127 255 255 255 255 255 255 255 255 255 255 255 25
**5 255 255 255 255 255 255 255 255 255 255 255 255 255 ) bitsPerLine: 17).
**
```

XEROX XEROX

EARS

Filename: fieldservice.st

Creation Date: April 26, 1978 10:59 AM

Printed by: Weyer, Steve

XEROX XEROX

```
'From Smalltalk 5.3b/xm on 6 April 1978 at 12:42:35 pm.'⌋
                                                                    "Location"
Class new title: 'Location'
    subclassof: Station
    fields: ''
    declare: '';
    asFollows⌋

This class has not yet been commented

As yet unclassified
⌋
SystemOrganization classify: ↱Location under: 'FieldService'.⌋
SystemOrganization classify: ↱Location alsounder: 'Station'.⌋
```

```
Class new title: 'Machine'
    subclassof: Job
    fields: ''
    declare: '';
    asFollows⌋
```

This class has not yet been commented

As yet unclassified

```
⌋
SystemOrganization classify: ↪Machine under: 'FieldService'.⌋
SystemOrganization classify: ↪Machine alsounder: 'Job'.⌋
```

```
Class new title: 'Repair1'
    subclassof: Simulation
    fields: ''
    declare: '';
    asFollows┘

This class has not yet been commented

Parts
arrivalSchedule
     "State the job, input schedule,
     and initia assignments (the
     names of stations the job will
     visi:). For example"

    [ self use: Machine new
             startAt: 0
             schedule:
                (Inputschedule new
                          constant: 40)
             assignments: (Inputschedule new data: 'Loc1', 'Loc2', 'Loc3',
'Loc4', 'Loc5', 'Loc6') .]
layout | var workers i stns
    ["Create the stations with workers."

     "Provide space for the station."
    stns←Set new default.
    for⦂ i to: 3 do⦂
        [ var ← Location init
        name: ('Loc'+i asString)
         of: ((i*100-80)⊙20
             rect: (i*100)⊙140).
        var use: (TechRep init id: i in: var).
        stns next←var].
    for⦂ i to: 3 do⦂
        [ var ← Location init
        name: ('Loc'+(i+3) asString)
         of: ((i*100-80)⊙160
             rect: (i*100)⊙280).
        var use: (TechRep init id: (i+3) in: var).
        stns next←var].
     "Set up the workers."
    "workers←Set new default.
    for⦂ i to: 3 do⦂
        [workers next←TechRep init
        id: i in: (stns asStream◦i)]."
     "Now construct the stations, giving each a list of all TechReps from
whom service may be sought."
    for⦂ var from: stns do⦂
        [var threshold: 1.
        self constructStation: var.]]
┘
SystemOrganization classify: ↪Repair1 under: 'FieldService'.┘
SystemOrganization classify: ↪Repair1 alsounder: 'Simulation'.┘
```

```
Class new title: 'Repair2'
    subclassof: Simulation
    fields: ''
    declare: '';
    asFollows⌋

This class has not yet been commented

Parts
arrivalSchedule
      "State the job, input schedule,
      and initial assignments (the
      names of stations the job will
      visit). For example"

    [ self use: Machine new
                startAt: 0
                schedule:
                   (Inputschedule new
                                  constant: 40)
                assignments: (Inputschedule new data: 'Loc1', 'Loc2', 'Loc3',
'Loc4', 'Loc5', 'Loc6') .]
layout | var workers i stns
    ["Create the stations with workers."

     "Provide space for the station."
    stns←Set new default.
    for⦂ i to: 3 do⦂
        [ var ← Location init
        name: ('Loc'+i asString)
         of: ((i*100-80)⊙20
            rect: (i*100)⊙140).
        stns next←var].
    for⦂ i to: 3 do⦂
        [ var ← Location init
        name: ('Loc'+(i+3) asString)
         of: ((i*100-80)⊙160
            rect: (i*100)⊙280).
        stns next←var].
     "Set up the workers."
    workers←Set new default.
    for⦂ i to: 3 do⦂
        [workers next←TechRep init
        id: i in: (stns asStream∘i)].
     "Now construct the stations, giving each a list of all TechReps from
whom service may be sought."
    for⦂ var from: stns do⦂
        [var use: workers.
        var threshold: 1.
        self constructStation: var.]]
⌋
SystemOrganization classify: ↪Repair2 under: 'FieldService'.⌋
SystemOrganization classify: ↪Repair2 alsounder: 'Simulation'.⌋
```

```
Class new title: 'TechRep'
    subclassof: Worker
    fields: ''
    declare: '';
    asFollows⌋

This class has not yet been commented

Parts
picture
    "Name of the picture representing
     the worker. The default is a small
     rectangular shape."

    [ ⇑ 'dryer']
serviceTime: job
    "The time the worker spends
     giving service to the job.  It is
     possible that job is a set of jobs.
     Time might a function of the
     job feature."

    [ ⇑ 120]
speed
    "Number of display bits per travel
     time"

    [ ⇑ 16]
travelTime
    "The amount of time it takes the
     worker to travel one display bit if its
     travel speed=1."

    [ ⇑ 1.0]
⌋
SystemOrganization classify: ↪TechRep under: 'FieldService'.⌋
SystemOrganization classify: ↪TechRep alsounder: 'Worker'.⌋
```