5.20 Class *PLATFORM*

indexing

description: *"Platform–dependent properties. This class may be used as ancestor by classes needing its facilities"*

class interface

PLATFORM

feature -- Access

Boolean_bits: *INTEGER*
-- Number of bits in a value of type *BOOLEAN*
ensure
meaningful: *Result >= 1*

Character_bits: *INTEGER*
-- Number of bits in a value of type *CHARACTER*
ensure
meaningful: *Result >= 1*
large_enough: *2 ^ Result >= Maximum_character_code*

Double_bits: *INTEGER*
-- Number of bits in a value of type *DOUBLE*
ensure
meaningful: *Result >= 1*;
meaningful: *Result >= Real_bits*

Integer_bits: *INTEGER*
-- Number of bits in a value of type *INTEGER*
ensure
meaningful: *Result >= 1*;
large_enough: *2 ^ Result >= Maximum_integer*;
large_enough_for_negative: *2 ^ Result >= – Minimum_integer*

Maximum_character_code: *INTEGER*
-- Largest supported code for *CHARACTER* values
ensure
meaningful: *Result >= 127*

Maximum_integer: *INTEGER*
-- Largest supported value of type *INTEGER*.
ensure
meaningful: *Result >= 0*

Minimum_character_code: *INTEGER*
-- Smallest supported code for *CHARACTER* values
ensure
meaningful: ~~*Result >=*~~ *Result <= 0*

Minimum_integer: *INTEGER*
-- Smallest supported value of type *INTEGER*
ensure
meaningful: *Result <= 0*

Pointer_bits: *INTEGER*
-- Number of bits in a value of type *POINTER*
ensure
meaningful: *Result >= 1*

Real_bits: *INTEGER*
-- Number of bits in a value of type *REAL*
ensure
meaningful: *Result >= 1*

end

5.19 Class *ARGUMENTS*

indexing

description: *"Access to command-line arguments. This class may be used as ancestor by classes needing its facilities."*

class interface

ARGUMENTS

feature -- Access

argument (*i*: *INTEGER*): *STRING*
-- *i*-th argument of command that started system execution
-- (the command name if *i* = 0)
require
index_large_enough: *i >= 0*;
index_small_enough: *i <= argument_count*

command_name: *STRING*
-- Name of command that started system execution
ensure
definition: *Result = argument (0)*

feature -- Measurement

argument_count: *INTEGER*
-- Number of arguments given to command that started
-- system execution (command name does not count)
~~***eusnure***~~
ensure
Result >= 0

end

5.18 Class *EXCEPTIONS*

indexing

description: *"Facilities for adapting the exception handling mechanism. This class may be used as ancestor by classes needing its facilities."*

class interface

EXCEPTIONS

feature -- Access

developer_exception_name: *STRING*

-- Name of last developer-raised exception

require

applicable: *is_developer_exception*

feature -- Status report

assertion_violation: *BOOLEAN*

-- Is last exception originally due to a violated

-- assertion or non–decreasing variant?

exception: *INTEGER*

-- Code of last exception that occurred

is_developer_exception: *BOOLEAN*

-- Is the last exception originally due to

-- a developer exception?

is_signal: *BOOLEAN*

-- Is last exception originally due to an external

-- event (operating system signal)?

feature -- Basic operations

die (*code*: *INTEGER*)

-- Terminate execution with exit status *code*,

-- without triggering an exception.

raise (*name*: *STRING*)

-- Raise a developer exception of name *name*.

end

5.17 Class *MEMORY*

indexing

description: *"Facilities for tuning up the garbage collection mechanism. This class may be used as ancestor by classes needing its facilities."*

class interface

MEMORY

feature -- Status report

collecting: *BOOLEAN*
-- Is garbage collection enabled?

feature -- Status setting

collection_off
-- Disable garbage collection.

collection_on
-- Enable garbage collection.

feature -- Removal

dispose
-- Action to be executed just before garbage collection
-- reclaims an object.
-- Default version does nothing; redefine in descendants
-- to perform specific dispose actions. Those actions
-- should only take care of freeing external resources;
-- they should not perform remote calls on other objects
-- since these may also be dead and reclaimed.

full_collect
-- Force a full collection cycle if garbage
-- collection is enabled; do nothing otherwise.

end

5.16 Class *STORABLE*

indexing

description: *"Objects that may be stored and retrieved along with all their dependents. This class may be used as ancestor by classes needing its facilities."*

class interface

STORABLE

feature -- Access

retrieved (*file*: *FILE*): *STORABLE*
-- Retrieved object structure, from external
-- representation previously stored in *file*.
-- To access resulting object under correct type,
-- use assignment attempt.
-- Will raise an exception (code *Retrieve_exception*)
-- if file content is not a *STORABLE* structure.

require

file_not_void: *file /= Void;*
file_exists: *file*.*exists;*
file_is_open_read: *file*.*is_open_read*
file_not_plain_text: ***not*** *file*.*is_plain_text*

ensure

result_exists: *Result /= Void*

feature -- Element change

basic_store (*file*: *FILE*)
-- Produce on *file* an external representation of the
-- entire object structure reachable from current object.
-- Retrievable within current system only.

require

file_not_void: *file /= Void;*
file_exists: *file*.*exists;*
file_is_open_write: *file*.*is_open_write;*
file_not_plain_text: ***not*** *file*.*is_plain_text*

general_store (*file*: *FILE*)
-- Produce on *file* an external representation of the
-- entire object structure reachable from current object.
-- Retrievable from other systems for same platform
-- (machine architecture).

require

file_not_void: *file /= Void;*
file_exists: *file*.*exists;*
file_is_open_write: *file*.*is_open_write;*
file_not_plain_text: ***not*** *file*.*is_plain_text*

independent_store (*file*: *FILE*)
-- Produce on *file* an external representation of the
-- entire object structure reachable from current object.
-- Retrievable from other systems for the same or other
-- platforms (machine architectures).

require

file_not_void: *file /= Void;*
file_exists: *file*.*exists;*
file_is_open_write: *file*.*is_open_write;*
file_not_plain_text: ***not*** *file*.*is_plain_text*

end

feature -- Removal

delete
-- Remove link with physical file; delete physical
-- file if no more link.
require
exists: *exists*

dispose
-- Ensure this medium is closed when garbage-collected.

feature -- Input

read_character
-- Read a new character.
-- Make result available in *last_character*.
require
is_readable: *file_readable*
is_readable: *readable*

read_double
-- Read the ASCII representation of a new double
-- from file. Make result available in *last_double*.
require
is_readable: *file_readable*
is_readable: *readable*

read_integer
-- Read the ASCII representation of a new integer
-- from file. Make result available in *last_integer*.
require
is_readable: *file_readable*
is_readable: *readable*

read_line
-- Read a string until new line or end of file.
-- Make result available in *laststring*.
-- New line will be consumed but not part of *last_string*.
require
is_readable: *file_readable*
is_readable: *readable*

read_real
-- Read the ASCII representation of a new real
-- from file. Make result available in *last_real*.
require
is_readable: *file_readable*
is_readable: *readable*

read_stream (*nb_char*: *INTEGER*)
-- Read a string of at most *nb_char* bound characters
-- or until end of file.
-- Make result available in *last_string*.
require
is_readable: *file_readable*
is_readable: *readable*

read_word
-- Read a new word from standard input.
-- Make result available in *last_string*.

feature -- Output

put_boolean (*b*: *BOOLEAN*)
-- Write ASCII value of *b* at current position.
require
extendible: *extendible*

put_character (*c*: *CHARACTER*)
-- Write *c* at current position.
require
extendible: *extendible*

put_double (*d*: *DOUBLE*)
-- Write ASCII value of *d* at current position.
require
extendible: *extendible*

put_integer (*i*: *INTEGER*)
-- Write ASCII value of *i* at current position.
require
extendible: *extendible*

put_real (*r*: *REAL*)
-- Write ASCII value of *r* at current position.
require
extendible: *extendible*

put_string (*s*: *STRING*)
-- Write *s* at current position.
require
extendible: *extendible*

invariant

name_exists: *name /= Void;*

name_not_empty: ***not*** *name*.*empty;*

writable_if_extendible: *extendible* ***implies*** *is_writable*

end

exists: *BOOLEAN*
-- Does physical file exist?

extendible: *BOOLEAN*
-- May new items be added?

is_closed: *BOOLEAN*
-- Is file closed?

is_creatable: *BOOLEAN*
-- Is file creatable in parent directory?

is_open_read: *BOOLEAN*
-- Is file open for reading?

is_open_write: *BOOLEAN*
-- Is file open for writing?

is_plain_text: *BOOLEAN*
-- Is file reserved for text (character sequences)?

is_readable: *BOOLEAN*
-- Is file readable?
require
handle_exists: *exists*

is_writable: *BOOLEAN*
-- Is file writable?
require
handle_exists: *exists*

last_character: *CHARACTER*
-- Last character read by *read_character*

last_double: *DOUBLE*
-- Last double read by *read_double*

last_integer: *INTEGER*
-- Last integer read by *read_integer*

last_real: *REAL*
-- Last real read by *read_real*

last_string: *STRING*
-- Last string read by *read_line*,
-- *read_stream*, or *read_word*

feature -- Status setting

close
-- Close file.
require
medium_is_open: ***not*** *is_closed*
ensure
is_closed: *is_closed*

open_read
-- Open file in read–only mode.
require
is_closed: *is_closed*
ensure
exists: *exists;*
open_read: *is_open_read*

open_read_append
-- Open file in read and write–at-end mode;
-- create it if it does not exist.
require
is_closed: *is_closed*
ensure
exists: *exists;*
open_read: *is_open_read;*
open_append: *is_open_append*

open_read_write
-- Open file in read and write mode.
require
is_closed: *is_closed*
ensure
exists: *exists;*
open_read: *is_open_read;*
open_write: *is_open_write*

open_write
-- Open file in write–only mode;
-- create it if it does not exist.
ensure
exists: *exists;*
open_write: *is_open_write*

feature -- Cursor movement

next_line
-- Move to next input line.
require
is_readable: *file_readable*

feature -- Element change

change_name (*new_name*: *STRING*)
-- Change file name to *new_name*
require
not_new_name_void: *new_name* /= *Void;*
file_exists: *exists*
ensure
name_changed: *name*.*is_equal* (*new_name*)

5.15 Class *FILE*

indexing

description: *"Files viewed as persistent sequences of characters"*

class interface

FILE

creation

make (*fn*: *STRING*)
-- Create file object with *fn* as file name.
require
string_exists: *fn /= Void;*
string_not_empty: ***not*** *fn*.*empty*
ensure
file_named: *name*.*is_equal* (*fn*);
file_closed: *is_closed*

make_create_read_write (*fn*: *STRING*)
-- Create file object with *fn* as file name
-- and open file for both reading and writing;
-- create it if it does not exist.
require
string_exists: *fn /= Void;*
string_not_empty: ***not*** *fn*.*empty*
ensure
exists: *exists;*
open_read: *is_open_read;*
open_write: *is_open_write*

make_open_append (*fn*: *STRING*)
-- Create file object with *fn* as file name
-- and open file in append–only mode.
require
string_exists: *fn /= Void;*
string_not_empty: ***not*** *fn*.*empty*
ensure
exists: *exists;*
open_append: *is_open_append*

make_open_read (*fn*: *STRING*)
-- Create file object with *fn* as file name
-- and open file in read mode.
require
string_exists: *fn /= Void;*
string_not_empty: ***not*** *fn*.*empty*
ensure
exists: *exists;*
open_read: *is_open_read*

make_open_read_write (*fn*: *STRING*)
-- Create file object with *fn* as file name
-- and open file for both reading and writing.
require
string_exists: *fn /= Void;*
string_not_empty: ***not*** *fn*.*empty*
ensure
exists: *exists;*
open_read: *is_open_read;*
open_write: *is_open_write*

make_open_write (*fn*: *STRING*)
-- Create file object with *fn* as file name
-- and open file for writing;
-- create it if it does not exist.
require
string_exists: *fn /= Void;*
string_not_empty: ***not*** *fn*.*empty*
ensure
exists: *exists;*
open_write: *is_open_write*

feature -- Access

name: *STRING*
-- File name

feature -- Measurement

count: *INTEGER*
-- Size in bytes (0 if no associated physical file)

feature -- Status report

empty: *BOOLEAN*
-- Is structure empty?

end_of_file: *BOOLEAN*
-- Has an EOF been detected?
require
opened: ***not*** *is_closed*

5.14 Class *STD_FILES*

indexing

description: *"Commonly used input and output mechanisms. This class may be used as either ancestor or supplier by classes needing its facilities."*

class interface

STD_FILES

feature -- Access

default_output: *FILE*
-- Default output.

error: *FILE*
-- Standard error file

input: *FILE*
-- Standard input file

output: *FILE*
-- Standard output file

standard_default: *FILE*
-- Return the *default_output* or *output*
-- if *default_output* is Void.

feature -- Status report

last_character: *CHARACTER*
-- Last character read by *read_character*

last_double: *DOUBLE*
-- Last double read by *read_double*

last_integer: *INTEGER*
-- Last integer read by *read_integer*

last_real: *REAL*
-- Last real read by *read_real*

last_string: *STRING*
-- Last string read by *read_line*,
-- *read_stream*, or *read_word*

feature -- Element change

new_line
-- Write line feed at end of default output.

put_boolean (*b*: *BOOLEAN*)
-- Write *b* at end of default output.

put_character (*c*: *CHARACTER*)
-- Write *c* at end of default output.

put_double (*d*: *DOUBLE*)
-- Write *d* at end of default output.

put_integer (*i*: *INTEGER*)
-- Write *i* at end of default output.

put_real (*r*: *REAL*)
-- Write *r* at end of default output.

put_string (*s*: *STRING*)
-- Write *s* at end of default output.
require
s /= *Void*

set_error_default
-- Use standard error as default output.

set_output_default
-- Use standard output as default output.

feature -- Input

next_line
-- Move to next input line on standard input.

read_character
-- Read a new character from standard input.
-- Make result available in *last_character*.

read_double
-- Read a new double from standard input.
-- Make result available in *last_double*.

read_integer
-- Read a new integer from standard input.
-- Make result available in *last_integer*.

read_line
-- Read a line from standard input.
-- Make result available in *last_string*.
-- New line will be consumed but not part of *last_string*.

read_real
-- Read a new real from standard input.
-- Make result available in *last_real*.

read_stream (*nb_char*: *INTEGER*)
-- Read a string of at most *nb_char* bound characters
-- from standard input.
-- Make result available in *last_string*.

read_word
-- Read a new word from standard input.
-- Make result available in *last_string*.

end

prune_all (*c*: *CHARACTER*)
-- Remove all occurrences of *c*.
ensure
changed_count: *count* = (***old*** *count*) – (***old*** *occurrences* (*c*))
-- *removed*: For every *i* in 1..*count*, *item* (*i*) /= *c*
no_more_occurrences: ***not*** *has* (*c*)

remove (*i*: *INTEGER*)
-- Remove *i*–th character.
require
index_small_enough: *i* <= *count;*
index_large_enough: *i* > *0*
ensure
new_count: *count* = ***old*** *count – 1*

wipe_out
-- Remove all characters.
ensure
empty_string: *count = 0;*
wiped_out: *empty*

feature -- Resizing

resize (*newsize*: *INTEGER*)
-- Rearrange string so that it can accommodate
-- at least *newsize* characters.
-- Do not lose any previously entered character.
require
new_size_non_negative: *newsize* >= *0*

feature -- Conversion

mirror
-- Reverse the order of characters.
-- "Hello world" –> "dlrow olleH".
ensure
same_count: *count* = ***old*** *count*
-- *reversed*: For every *i* in 1..*count*,
-- *item* (*i*) = ***old*** *item* (*count*+1–*i*)

mirrored: ***like*** *Current*
-- Mirror image of string;
-- result for "Hello world" is "dlrow olleH".
ensure
same_count: *Result*.*count* = *count*
-- *reversed*: For every *i* in 1..*count*,
-- *Result*.*item* (*i*) = *item* (*count*+1–*i*)

to_boolean: *BOOLEAN*
-- Boolean value;
-- "true" yields *true*, "false" yields *false*
-- (case–insensitive)

to_pointer: *POINTER*
-- A pointer to a C form of current string.
-- Useful only for interfacing with C software.

to_double: *DOUBLE*
-- "Double" value;
-- for example, when applied to "123.0", will yield 123.0 (double)

to_integer: *INTEGER*
-- Integer value;
-- for example, when applied to "123", will yield 123

to_lower
-- Convert to lower case.

to_real: *REAL*
-- Real value;
-- for example, when applied to "123.0", will yield 123.0

to_upper
-- Convert to upper case.

feature -- Duplication

copy (*other*: ***like*** *Current*)
-- Reinitialize by copying the characters of *other*.
-- (This is also used by *clone*.)
-- (From *GENERAL*.)
ensure
new_result_count: *count* = *other*.*count*
-- *same_characters*: For every *i* in 1..*count*,
-- *item* (*i*) = *other*.*item* (*i*)

substring (*n1*, *n2*: *INTEGER*): ***like*** *Current*
-- Copy of substring containing all characters at indices
-- between *n1* and *n2*
require
meaningful_origin: *1* <= *n1;*
meaningful_interval: *n1* <= *n2;*
meaningful_end: *n2* <= *count*
ensure
new_result_count: *Result*.*count* = *n2 – n1 + 1*
-- *original_characters*: For every *i* in 1..*n2–n1*,
-- *Result*.*item* (*i*) = *item* (*n1*+*i*–1)

feature -- Output

out: ***like*** *Current*
-- Printable representation
-- (From *GENERAL*.)

invariant

irreflexive_comparison: ***not*** (*Current* < *Current*);

empty_definition: *empty* = (*count* = *0*);

non_negative_count: *count* >= *0*

end

append_string (*s*: *STRING*)
-- Append a copy of *s*, if not void, at end.
ensure
new_count: *count* = ***old*** *count* + *s*.*count*
-- *appended*: For every *i* in 1..*s*.*count*,
-- *item* (***old*** *count* + *i*) = *s*.*item* (*i*)

fill_blank
-- Fill with blanks.
ensure
-- *allblank*: For every *i* in 1..*count*, *item* (*i*) = *Blank*

head (*n*: *INTEGER*)
-- Remove all characters except for the first *n*;
-- do nothing if *n* >= *count*.
require
non_negative_argument: *n* >= *0*
ensure
new_count: *count* = *n*.*min* (***old*** *count*)
-- *first_kept*: For every *i* in 1..*n*, *item* (*i*) = ***old*** *item* (*i*)

insert (*s*: ***like*** *Current; i*: *INTEGER*)
-- Add *s* to the left of position *i* in current string.
require
string_exists: *s* /= *Void;*
index_small_enough: *i* <= *count;*
index_large_enough: *i* > *0*
ensure
new_count: *count* = ***old*** *count* + *s*.*count*

left_adjust
-- Remove leading white space.
ensure
new_count: (*count* /= *0*) ***implies*** (*item* (*1*) /= ' ')

precede (*c*: *CHARACTER*)
-- Add *c* at front.
ensure
new_count: *count* = ***old*** *count* + *1*

prepend (*s*: *STRING*)
-- Prepend a copy of *s* at front.
require
argument_not_void: *s* /= *Void*
ensure
new_count: *count* = ***old*** *count* + *s*.*count*

put (*c*: *CHARACTER; i*: *INTEGER*)
-- Replace character at position *i* by *c*.
require
good_key: *valid_index* (*i*)
ensure
insertion_done: *item* (*i*) = *c*

replace_substring (*s*: ***like*** *Current; start_pos*, *end_pos*: *INTEGER*)
-- Copy the characters of *s* to positions
-- *start_pos* .. *end_pos*.
require
string_exists: *s* /= *Void;*
index_small_enough: *end_pos* <= *count;*
order_respected: *start_pos* <= *end_pos;*
index_large_enough: *start_pos* > *0*
ensure
new_count: *count* = ***old*** *count* + *s*.*count* – *end_pos* + *start_pos* – *1*

right_adjust
-- Remove trailing white space.
ensure
new_count: (*count* /= *0*) ***implies*** (*item* (*count*) /= ' ')

set (*t*: ***like*** *Current; n1*, *n2*: *INTEGER*)
-- Set current string to substring of *t* from indices *n1*
-- to *n2*, or to empty string if no such substring.
require
argument_not_void: *t* /= *Void*
ensure
is_substring: *is_equal* (*t*.*substring* (*n1*, *n2*))

share (*other*: ***like*** *Current*)
-- Make current string share the text of *other*.
-- Subsequent changes to the characters of current string
-- will also affect *other*, and conversely.
require
argument_not_void: *other* /= *Void*
ensure
shared_count: *other*.*count* = *count*
-- *sharing*: For every *i* in 1..*count*,
-- *Result*.*item* (*i*) = *item* (*i*)

tail (*n*: *INTEGER*)
-- Remove all characters except for the last *n*;
-- do nothing if *n* >= *count*.
require
non_negative_argument: *n* >= *0*
ensure
new_count: *count* = *n*.*min* (***old*** *count*)

feature -- Removal

prune (*c*: *CHARACTER*)
-- Remove first occurrence of *c*, if any.

feature -- Comparison

is_equal (*other*: ***like*** *Current*): *BOOLEAN*
-- Is string made of same character sequence as *other*?
-- (Redefined from *GENERAL*.)
require
other_exists: *other /= Void*

infix *"<"* (*other*: *STRING*): *BOOLEAN*
-- Is string lexicographically lower than *other*?
-- (False if *other* is void)
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
asymmetric: *Result* ***implies not*** (*other < Current*)

infix *"<="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object less than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*Current < other*) ***or*** *is_equal* (*other*);

infix *">="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other <= Current*)

infix *">"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other < Current*)

max (*other*: ***like*** *Current*): ***like*** *Current*)
-- The greater of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_smaller: (*Current >= other*) ***implies*** (*Result = Current*)
other_if_smaller: (*Current < other*) ***implies*** (*Result = other*)

min (*other*: ***like*** *Current*): ***like*** *Current*)
-- The smaller of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_greater: (*Current <= other*) ***implies*** (*Result = Current*)
other_if_greater: (*Current > other*) ***implies*** (*Result = other*)

three_way_comparison (*other*: ***like*** *Current*): *INTEGER*)
-- If current object equal to *other*, 0; if smaller,
-- –1; if greater, 1.
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
equal_zero: (*Result = 0*) = *is_equal* (*other*);
smaller: (*Result = –1*) = *Current < other*;
greater_positive: (*Result = 1*) = *Current > other*

feature -- Status report

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

empty: *BOOLEAN*
-- Is string empty?

valid_index (*i*: *INTEGER*): *BOOLEAN*
-- Is *i* within the bounds of the string?

feature -- Element change

append_boolean (*b*: *BOOLEAN*)
-- Append the string representation of *b* at end.

append_character (*c*: *CHARACTER*)
-- Append *c* at end.
ensure
item_inserted: *item* (*count*) = *c*
one_more_occurrence: *occurrences* (*c*) = ***old*** (*occurrences* (*c*)) + *1*
item_inserted: *has* (*c*)

append_double (*d*: *DOUBLE*)
-- Append the string representation of *d* at end.

append_integer (*i*: *INTEGER*)
-- Append the string representation of *i* at end.

append_real (*r*: *REAL*)
-- Append the string representation of *r* at end.

5.13 Class *STRING*

indexing

description: *"Sequences of characters, accessible through integer indices in a contiguous range."*

class interface

STRING

creation

frozen *make* (*n*: *INTEGER*)
-- Allocate space for at least *n* characters.
require
non_negative_size: *n >= 0*
ensure
empty_string: *count = 0*

make_from_string (*s*: *STRING*)
-- Initialize from the characters of *s*.
-- (Useful in proper descendants of class *STRING*,
-- to initialize a string-like object from a manifest string.)
require
string_exists: s/= Void

feature -- Initialization

from_c (*c_string*: *POINTER*)
-- Reset contents of string from contents of *c_string*,
-- a string created by some external C function.
require
C_string_exists: c_string /= Void

frozen *remake* (*n*: *INTEGER*)
-- Allocate space for at least *n* characters.
require
non_negative_size: *n >= 0*
ensure
empty_string: *count = 0*

make_from_string (*s*: *STRING*)
-- Initialize from the characters of *s*.
-- (Useful in proper descendants of class *STRING*,
-- to initialize a string-like object from a manifest string.)
require
string_exists: s/= Void

feature -- Access

has (*c*: *CHARACTER*): *BOOLEAN*
-- Does string include *c*?
ensure
not_found_in_empty: *Result* ***implies not*** *empty*

hash_code: *INTEGER*
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

index_of (*c*: *CHARACTER; start*: *INTEGER*): *INTEGER*
-- Position of first occurrence of *c* at or after *start*;
-- 0 if none.
require
start_large_enough: *start >= 1*;
start_small_enough: *start <= count*
ensure
non_negative_result: *Result >= 0*;
at_this_position: *Result > 0* ***implies*** *item* (*Result*) = *c*;
-- *none_before*: For every *i* in *start..Result*, *item* (*i*) /= *c*
-- *zero_iff_absent*:
-- (*Result = 0*) = For every *i* in 1..*count*, *item* (*i*) /= *c*

item (*i*: *INTEGER*): *CHARACTER*
-- Character at position *i*
require
good_key: *valid_index* (*i*)

item_code (*i*: *INTEGER*): *INTEGER*
-- Numeric code of character at position *i*
require
index_small_enough: *i <= count;*
index_large_enough: *i > 0*

shared_with (*other*: ***like*** *Current*): *BOOLEAN*
-- Does string share the text of *other*?

substring_index (*other*: *STRING; start*: *INTEGER*): *INTEGER*
-- Position of first occurrence of *other* at or after *start*;
-- 0 if none.

infix *"@"* (*i*: *INTEGER*): *CHARACTER*
-- Character at position *i*
require
good_key: *valid_index* (*i*)

feature -- Measurement

count: *INTEGER*
-- Actual number of characters making up the string

occurrences (*c*: *CHARACTER*): *INTEGER*
-- Number of times *c* appears in the string
ensure
non_negative_occurrences: *Result >= 0*

frozen *put* (*v*: ***like*** *item; i*: *INTEGER*)
-- Replace *i*–th entry, if in index interval, by *v*.
require
good_key: *valid_index* (*i*)
ensure
inserted: *item* (*i*) = *v*

feature -- Resizing

resize (*minindex*, *maxindex*: *INTEGER*)
-- Rearrange array so that it can accommodate
-- indices down to *minindex* and up to *maxindex*.
-- Do not lose any previously entered item.
require
good_indices: *minindex* <= *maxindex*
ensure
no_low_lost: *lower* = *minindex*.*min* (***old*** *lower*);
no_high_lost: *upper* = *maxindex*.*max* (***old*** *upper*)

feature -- Conversion

to_c: *POINTER*
-- Address of actual sequence of values,
-- for passing to external (non–Eiffel) routines.

feature -- Duplication

copy (*other*: ***like*** *Current*)
-- Reinitialize by copying all the items of *other*.
-- (This is also used by *clone*.)
-- (From *GENERAL*.)

invariant

consistent_size: *count* = *upper* – *lower* + *1*;

non_negative_count: *count* >= *0*

end

5.12 Class *ARRAY*

indexing

description: *"Sequences of values, all of the same type or of a conforming one, accessible through integer indices in a contiguous interval"*

class interface

ARRAY [*G*]

creation

make (*minindex*, *maxindex*: *INTEGER*)
-- Allocate array; set index interval to
-- *minindex* .. *maxindex*; set all values to default.
-- (Make array empty if *minindex* > *maxindex*.)
ensure
no_count: (*minindex* > *maxindex*) ***implies*** (*count* = *0*);
count_constraint: (*minindex* <= *maxindex*) ***implies*** (*count* = *maxindex* – *minindex* + *1*)

make_from_array (*a*: *ARRAY* [*G*])
-- Initialize from the items of *a*.
-- (Useful in proper descendants of class *ARRAY,*
-- to initialize an array-like object from a manifest array.)

feature -- Initialization

make (*minindex*, *maxindex*: *INTEGER*)
-- Set index interval to *minindex* .. *maxindex*;
-- reallocate if necessary; set all values to default.
-- (Make array empty if *minindex* > *maxindex*.)
ensure
no_count: (*minindex* > *maxindex*) ***implies*** (*count* = *0*);
count_constraint: (*minindex* <= *maxindex*) ***implies*** (*count* = *maxindex* – *minindex* + *1*)

make_from_array (*a*: *ARRAY* [*G*])
-- Initialize from the items of *a*; reallocate if
-- necessary. (Useful in proper descendants of
-- class *ARRAY,* to initialize an array-like object
-- from a manifest array.)

feature -- Access

entry (*i*: *INTEGER*): *G*
-- Entry at index *i*, if in index interval
-- (Redefinable synonym for *item* and ***infix*** "@".)
require
good_key: *valid_index* (*i*)

frozen *item* (*i*: *INTEGER*): *G*
-- Entry at index *i*, if in index interval
require
good_key: *valid_index* (*i*)

frozen infix *"@"* (*i*: *INTEGER*): *G*
-- Entry at index *i*, if in index interval
require
good_key: *valid_index* (*i*)

feature -- Measurement

count: *INTEGER*
-- Number of available indices

lower: *INTEGER*
-- Minimum index

upper: *INTEGER*
-- Maximum index

feature -- Comparison

is_equal (*other*: ***like*** *Current*): *BOOLEAN*
-- Is array made of the same items as *other*?
-- (Redefined from *GENERAL*.)

feature -- Status report

valid_index (*i*: *INTEGER*): *BOOLEAN*
-- Is *i* within the bounds of the array?

feature -- Element change

enter (*v*: *G; i*: *INTEGER*)
-- Replace *i*–th entry, if in index interval, by *v*.
-- (Redefinable synonym for *put*.)
require
good_key: *valid_index* (*i*)
ensure
inserted: *item* (*i*) = *v*

force (*v*: ***like*** *item; i*: *INTEGER*)
-- Assign item *v* to *i*–th entry.
-- Always applicable: resize the array if *i* falls out of
-- currently defined bounds; preserve existing items.
ensure
inserted: *item* (*i*) = *v;*
higher_count: *count* >= ***old*** *count*

5.11 Class *POINTER*

indexing

description: *"References to objects meant to be exchanged with non-Eiffel software."*

deferred class interface

POINTER

feature -- Access

hash_code: *INTEGER*
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

feature -- Status report

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

feature -- Output

out: *STRING*
-- Printable representation of pointer value
-- (From *GENERAL*.)

end

```
    infix "^" (other: like Current): like Current
            -- Current double to the power other
            -- (From NUMERIC.)
        require
            other_exists: other /= Void;
            good_exponent: exponentiable (other)
        ensure
            result_exists: Result /= Void

    prefix "+": like Current
            -- Unary plus
            -- (From NUMERIC.)
        ensure
            result_exists: Result /= Void

    prefix "–": like Current
            -- Unary minus
            -- (From NUMERIC.)
        ensure
            result_exists: Result /= Void

feature -- Output

    out: STRING
            -- Printable representation of double value
            -- (From GENERAL.)

invariant

    irreflexive_comparison: not (Current < Current);

    neutral_addition: equal (Current + zero, Current);

    self_subtraction: equal (Current – Current, zero);

    neutral_multiplication: equal (Current * one, Current);

    self_division: divisible (Current) implies equal (Current /
      Current, one)

    sign_times_abs: equal (sign*abs, Current)

end
```

three_way_comparison (*other*: ***like*** *Current*): *INTEGER*
-- If current object equal to *other*, 0; if smaller,
-- –1; if greater, 1.
require
other_exists: *other /= Void*
-- (From *COMPARABLE*.)
ensure
equal_zero: (*Result = 0*) = *is_equal* (*other*);
smaller: (*Result = –1*) = *Current < other*;
greater_positive: (*Result = 1*) = *Current > other*

feature -- Status report

divisible (*other*: ***like*** *Current*): *BOOLEAN*
-- May current object be divided by *other*?
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
not_exact_zero: *Result* ***implies*** (*other /= 0.0*)

exponentiable (*other*: *NUMERIC*): *BOOLEAN*
-- May current object be elevated to the power *other*?
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
safe_values: (*other.conforms_to* (*0*) ***or***
(*other.conforms_to* (*Current*) ***and*** (*Current >= 0.0*)))
implies *Result*

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

feature -- Conversion

ceiling: *INTEGER*
-- Smallest integral value no smaller than current object
ensure
result_no_smaller: *Result >= Current*;
close_enough: *Result – Current < one*

floor: *INTEGER*
-- Greatest integral value no greater than current object
ensure
result_no_greater: *Result <= Current*;
close_enough: *Current – Result < one*

rounded: *INTEGER*
-- Rounded integral value
ensure
definition: *Result = sign * ((abs + 0.5).floor*)

truncated_to_integer: *INTEGER*
-- Integer part (same sign, largest absolute
-- value no greater than current object's)

truncated_to_real: *REAL*
-- Real part (same sign, largest absolute
-- value no greater than current object's)

feature -- Basic operations

abs: ***like*** *Current*
-- Absolute value
ensure
non_negative: *Result >= 0*;
same_absolute_value: (*Result = Current*) ***or*** (*Result = –Current*)

infix *"*"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Product by *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
result_exists: *Result /= Void*

infix *"+"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Sum with *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
result_exists: *Result /= Void*;
commutative: *equal* (*Result*, *other + Current*)

infix *"–"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Result of subtracting *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
result_exists: *Result /= Void*

infix *"/"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Division by *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*;
good_divisor: *divisible* (*other*)
ensure
result_exists: *Result /= Void*

5.10 Class *DOUBLE*

indexing

description: *"Real values, double precision"*

expanded class interface

DOUBLE

feature -- Access

hash_code: *INTEGER*
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

one: ***like*** *Current*
-- Neutral element for "*" and "/"
-- (From *NUMERIC*.)
ensure
Result_exists: *Result /= Void*;
value: *Result = 1.0*

sign: *INTEGER*
-- Sign value (0, –-1 or 1)
ensure
three_way: *Result = three_way_comparison* (*zero*)

zero: ***like*** *Current*
-- Neutral element for "+" and "–"
-- (From *NUMERIC*.)
ensure
Result_exists: *Result /= Void*;
value: *Result = 0.0*

feature -- Comparison

infix *"<"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* greater than current double?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
asymmetric: *Result* ***implies not*** (*other < Current*)

infix *"<="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object less than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*Current < other*) ***or*** *is_equal* (*other*);

infix *">="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other <= Current*)

infix *">"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other < Current*)

max (*other*: ***like*** *Current*): ***like*** *Current*
-- The greater of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_smaller: (*Current >= other*) ***implies*** (*Result = Current*)
other_if_smaller: (*Current < other*) ***implies*** (*Result = other*)

min (*other*: ***like*** *Current*): ***like*** *Current*
-- The smaller of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_greater: (*Current <= other*) ***implies*** (*Result = Current*)
other_if_greater: (*Current > other*) ***implies*** (*Result = other*)

infix *"^"* (*other*: *NUMERIC*): *DOUBLE*
-- Current real to the power *other*
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*;
good_exponent: *exponentiable* (*other*)
ensure
result_exists: *Result* /= *Void*

prefix *"+"*: ***like*** *Current*
-- Unary plus
-- (From *NUMERIC*.)
ensure
result_exists: *Result* /= *Void*

prefix *"–"*: ***like*** *Current*
-- Unary minus
-- (From *NUMERIC*.)
ensure
result_exists: *Result* /= *Void*

feature -- Output

out: *STRING*
-- Printable representation of real value
-- (From *GENERAL*.)

invariant

irreflexive_comparison: ***not*** (*Current* < *Current*);

neutral_addition: *equal* (*Current* + *zero*, *Current*);

self_subtraction: *equal* (*Current* – *Current*, *zero*);

neutral_multiplication: *equal* (*Current* * *one*, *Current*);

self_division: *divisible* (*Current*) ***implies*** *equal* (*Current* / *Current*, *one*)

sign_times_abs: *equal* (*sign*abs*, *Current)*

end

three_way_comparison (*other*: ***like*** *Current*): *INTEGER*
-- If current object equal to *other*, 0; if smaller,
-- –1; if greater, 1.
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
equal_zero: (*Result = 0*) = *is_equal* (*other*);
smaller: (*Result = –1*) = *Current < other*;
greater_positive: (*Result = 1*) = *Current > other*

feature -- Status report

divisible (*other*: ***like*** *Current*): *BOOLEAN*
-- May current object be divided by *other*?
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
not_exact_zero: *Result* ***implies*** (*other /= 0.0*)

exponentiable (*other*: *NUMERIC*): *BOOLEAN*
-- May current object be elevated to the power *other*?
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
safe_values: (*other.conforms_to* (*0*) ***or***
(*other.conforms_to* (*Current*) ***and*** (*Current >= 0.0*)))
implies *Result*

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

feature -- Conversion

ceiling: *INTEGER*
-- Smallest integral value no smaller than current object
ensure
result_no_smaller: *Result >= Current*;
close_enough: *Result – Current < one*

floor: *INTEGER*
-- Greatest integral value no greater than current object
ensure
result_no_greater: *Result <= Current*;
close_enough: *Current – Result < one*

rounded: *INTEGER*
-- Rounded integral value
ensure
definition: *Result = sign * ((abs + 0.5).floor*)

truncated_to_integer: *INTEGER*
-- Integer part (same sign, largest absolute
-- value no greater than current object's)

feature -- Basic operations

abs: ***like*** *Current*
-- Absolute value
ensure
non_negative: *Result >= 0*;
same_absolute_value: (*Result = Current*) ***or*** (*Result = –Current*)

infix *"*"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Product by *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
result_exists: *Result /= Void*

infix *"+"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Sum with *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
result_exists: *Result /= Void*;
commutative: *equal* (*Result*, *other + Current*)

infix *"–"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Result of subtracting *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*
ensure
result_exists: *Result /= Void*

infix *"/"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Division by *other*
-- (From *NUMERIC*.)
require
other_exists: *other /= Void*;
good_divisor: *divisible* (*other*)
ensure
result_exists: *Result /= Void*

5.9 Class *REAL*

indexing

description: *"Real values, single precision"*

expanded class interface

REAL

feature -- Access

hash_code: *INTEGER*
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

one: ***like*** *Current*
-- Neutral element for "*" and "/"
-- (From *NUMERIC*.)
ensure
Result_exists: *Result /= Void*;
value: *Result = 1.0*

sign: *INTEGER*
-- Sign value (0, –-1 or 1)
ensure
three_way: *Result = three_way_comparison* (*zero*)

zero: ***like*** *Current*
-- Neutral element for "+" and "–"
-- (From *NUMERIC*.)
ensure
Result_exists: *Result /= Void*;
value: *Result = 0.0*

feature -- Comparison

infix *"<"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* greater than current real?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
asymmetric: *Result* ***implies not*** (*other < Current*)

infix *"<="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object less than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*Current < other*) ***or*** *is_equal* (*other*);

infix *">="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other <= Current*)

infix *">"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other < Current*)

max (*other*: ***like*** *Current*): ***like*** *Current*
-- The greater of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_smaller: (*Current >= other*) ***implies*** (*Result = Current*)
other_if_smaller: (*Current < other*) ***implies*** (*Result = other*)

min (*other*: ***like*** *Current*): ***like*** *Current*
-- The smaller of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_greater: (*Current <= other*) ***implies*** (*Result = Current*)
other_if_greater: (*Current > other*) ***implies*** (*Result = other*)

```
    prefix "+": like Current
            -- Unary plus
            -- (From NUMERIC.)
        ensure
            result_exists: Result /= Void
    prefix "–": like Current
            -- Unary minus
            -- (From NUMERIC.)
        ensure
            result_exists: Result /= Void
feature -- Output
    out: STRING
            -- Printable representation of current object
            -- (From GENERAL.)
invariant
    irreflexive_comparison: not (Current < Current);
    neutral_addition: equal (Current + zero, Current);
    self_subtraction: equal (Current – Current, zero);
    neutral_multiplication: equal (Current * one, Current);
    self_division: divisible (Current) implies equal (Current /
      Current, one)
    sign_times_abs: equal (sign*abs, Current)
end
```

three_way_comparison (*other*: ***like*** *Current*): *INTEGER*
-- If current object equal to *other*, 0; if smaller,
-- –1; if greater, 1.
-- (From *COMPARABLE*.)
require
other_exists: *other* /= *Void*
ensure
equal_zero: (*Result* = *0*) = *is_equal* (*other*);
smaller: (*Result* = *1*) = *Current* < *other*;
greater_positive: (*Result* = *–1*) = *Current* > *other*

feature -- Status report

divisible (*other*: ***like*** *Current*): *BOOLEAN*
-- May current object be divided by *other*?
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*
ensure
value: *Result* = (*other* /= *0*)

exponentiable (*other*: *NUMERIC*): *BOOLEAN*
-- May current object be elevated to the power *other*?
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*
ensure
safe_values: (*other*.*conforms_to* (*Current*) ***or***
(*other*.*conforms_to* (*0*.*0*) ***and*** (*Current* >= *0*)))
implies *Result*

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current* /= *default*)

feature -- Basic operations

abs: ***like*** *Current*
-- Absolute value
ensure
non_negative: *Result* >= *0*;
same_absolute_value: (*Result* = *Current*) ***or*** (*Result* = *–Current*)

infix *"*"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Product by *other*
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*

infix *"+"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Sum with *other*
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*
ensure
result_exists: *Result* /= *Void*;
commutative: *equal* (*Result*, *other* + *Current*)

infix *"–"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Result of subtracting *other*
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*
ensure
result_exists: *Result* /= *Void*

infix *"/"* (*other*: ***like*** *Current*): *DOUBLE*
-- Division by *other*
require
other_exists: *other* /= *Void*;
good_divisor: *divisible* (*other*)
ensure
result_exists: *Result* /= *Void*

infix *"//"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Integer division of Current by *other*
-- (~~From *"//"*~~ From *"/"* in *NUMERIC*.)
require
other_exists: *other* /= *Void*;
good_divisor: *divisible* (*other*)
ensure
result_exists: *divisible* (*other*)

infix *"\\"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Remainder of the integer division of Current by *other*
require
other_exists: *other* /= *Void*;
good_divisor: *divisible* (*other*)
ensure
result_exists: *Result* /= *Void*

infix *"^"* (*other*: *NUMERIC*): *DOUBLE*
-- Integer power of Current by *other*
-- (From *NUMERIC*.)
require
other_exists: *other* /= *Void*;
good_exponent: *exponentiable* (*other*)
ensure
result_exists: *Result* /= *Void*

5.8 Class *INTEGER*

indexing

description: *"Integer values"*

expanded class interface

INTEGER

feature -- Access

hash_code: *INTEGER* ***is***
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

one: ***like*** *Current*
-- Neutral element for "*" and "/"
-- (From *NUMERIC*.)
ensure
Result_exists: *Result /= Void*;
value: *Result = 1*

sign: *INTEGER*
-- Sign value (0, –-1 or 1)
ensure
three_way: *Result = three_way_comparison* (*zero*)

zero: ***like*** *Current*
-- Neutral element for "+" and "–"
-- (From *NUMERIC*.)
ensure
Result_exists: *Result /= Void*;
value: *Result = 0*

feature -- Comparison

infix *"<"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* greater than current integer?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
asymmetric: *Result* ***implies not*** (*other < Current*)

infix *"<="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object less than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*Current < other*) ***or*** *is_equal* (*other*);

infix *">="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other <= Current*)

infix *">"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other < Current*)

max (*other*: ***like*** *Current*): ***like*** *Current*
-- The greater of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_smaller: (*Current >= other*) ***implies*** (*Result = Current*)
other_if_smaller: (*Current < other*) ***implies*** (*Result = other*)

min (*other*: ***like*** *Current*): ***like*** *Current*
-- The smaller of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_greater: (*Current <= other*) ***implies*** (*Result = Current*)
other_if_greater: (*Current > other*) ***implies*** (*Result = other*)

feature -- Status report

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)

ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

feature -- Output

out: *STRING*
-- Printable representation of character
-- (From *GENERAL*.)

invariant

irreflexive_comparison: ***not*** (*Current* < *Current*)

end

5.7 Class *CHARACTER*

indexing

description: *"Characters, with comparison operations and an ASCII code"*

expanded class interface

CHARACTER

feature -- Access

code: *INTEGER*
-- Associated integer value

hash_code: *INTEGER*
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

feature -- Comparison

infix *"<"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* greater than current character?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
asymmetric: *Result* ***implies not*** (*other < Current*)

infix *"<="* (*other*:***like*** *Current*): *BOOLEAN*
-- Is current character less than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*Current < other*) ***or*** *is_equal* (*other*);

infix *">="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than or equal to *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other <= Current*)

infix *">"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than *other*?
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (*other < Current*)

max (*other*: ***like*** *Current*): ***like*** *Current*
-- The greater of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_smaller: (*Current >= other*) ***implies*** (*Result = Current*)
other_if_smaller: (*Current < other*) ***implies*** (*Result = other*)

min (*other*: ***like*** *Current*): ***like*** *Current*
-- The smaller of current object and *other*
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
current_if_not_greater: (*Current <= other*) ***implies*** (*Result = Current*)
other_if_greater: (*Current > other*) ***implies*** (*Result = other*)

three_way_comparison (*other*: ***like*** *Current*): *INTEGER*
-- If current object equal to *other*, 0; if smaller,
-- –1; if greater, 1.
-- (From *COMPARABLE*.)
require
other_exists: *other /= Void*
ensure
equal_zero: (*Result = 0*) = *is_equal* (*other*);
smaller: (*Result = –1*) = *Current < other*;
greater_positive: (*Result = 1*) = *Current > other*

5.6 Class *BOOLEAN*

indexing

description: *"Truth values, with the boolean operations"*

expanded class interface

BOOLEAN

feature -- Access

hash_code: *INTEGER*
-- Hash code value
-- (From *HASHABLE*.)
require
hashable: *is_hashable*
ensure
good_hash_value: ~~*Result >*~~ *Result >= 0*

feature -- Status report

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
-- (From *HASHABLE*.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

feature -- Basic operations

infix *"**and**"* (*other*: *BOOLEAN*): *BOOLEAN*
-- Boolean conjunction with *other*
require
other_exists: *other /= Void*
ensure
Result_exists: *Result /= Void*;
de_morgan: *Result* = ***not*** (***not*** *Current* ***or*** (***not*** *other*));
commutative: *Result* = (*other* ***and*** *Current*);
consistent_with_semi_strict: *Result* ***implies*** (*Current* ***and then*** *other*)

infix *"**and then**"* (*other*: *BOOLEAN*): *BOOLEAN*
-- Boolean semi–strict conjunction with *other*
require
other_exists: *other /= Void*
ensure
Result_exists: *Result /= Void*;
de_morgan: *Result* = ***not*** (***not*** *Current* ***or else*** (***not*** *other*));

infix *"**implies**"* (*other*: *BOOLEAN*): *BOOLEAN*
-- Boolean implication of *other*
-- (semi–strict)
require
other_exists: *other /= Void*
ensure
definition: *Result* = (***not*** *Current* ***or else*** *other*)

prefix *"**not**"*: *BOOLEAN*
-- Negation.

infix *"**or**"* (*other*: *BOOLEAN*): *BOOLEAN*
-- Boolean disjunction with *other*
require
other_exists: *other /= Void*
ensure
Result_exists: *Result /= Void*;
de_morgan: *Result* = ***not*** (***not*** *Current* ***and*** (***not*** *other*));
commutative: *Result* = (*other* ***or*** *Current*);
consistent_with_semi_strict: *Result* ***implies*** (*Current* ***or else*** *other*)

infix *"**or else**"* (*other*: *BOOLEAN*): *BOOLEAN*
-- Boolean semi–strict disjunction with *other*
require
other_exists: *other /= Void*
ensure
Result_exists: *Result /= Void*;
de_morgan: *Result* = ***not*** (***not*** *Current* ***and then*** (***not*** *other*));

infix *"**xor**"* (*other*: *BOOLEAN*): *BOOLEAN*
-- Boolean exclusive or with *other*
require
other_exists: *other /= Void*
ensure
definition: *Result* = ((*Current* ***or*** *other*) ***and not*** (*Current* ***and*** *other*))

feature -- Output

out: *STRING*
-- Printable representation of boolean

invariant

involutive_negation: *is_equal* (***not*** (***not*** *Current*));
non_contradiction: ***not*** (*Current* ***and*** (***not*** *Current*));
completeness: *Current* ***or*** (***not*** *Current*)

end

invariant

neutral_addition: *equal* (*Current* + *zero*, *Current*);
self_subtraction: *equal* (*Current* – *Current*, *zero*);

neutral_multiplication: *equal* (*Current* * *one*, *Current*);

self_division: *divisible* (*Current*) ***implies*** *equal* (*Current* / *Current*, *one*)

end

5.5 Class *NUMERIC*

indexing

description: *"Objects to which numerical operations are applicable"*;

note: *"The model is that of a commutative ring."*

deferred class interface

NUMERIC

feature -- Access

one: ***like*** *Current*
-- Neutral element for "*" and "/"
deferred
ensure
Result_exists: *Result /= Void*

zero: ***like*** *Current*
-- Neutral element for "+" and "–"
deferred
ensure
Result_exists: *Result /= Void*

feature -- Status report

divisible (*other*: ***like*** *Current*): *BOOLEAN*
-- May current object be divided by *other*?
require
other_exists: *other /= Void*
deferred

exponentiable (*other*: *NUMERIC*): *BOOLEAN*
-- May current object be elevated to the power *other*?
require
other_exists: *other /= Void*
deferred

feature -- Basic operations

infix *"+"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Sum with *other* (commutative).
require
other_exists: *other /= Void*
deferred
ensure
result_exists: *Result /= Void*;
commutative: *equal* (*Result*, *other* + *Current*)

infix *"–"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Result of subtracting *other*
require
other_exists: *other /= Void*
deferred
ensure
result_exists: *Result /= Void*

infix *"*"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Product by *other*
require
other_exists: *other /= Void*
deferred
ensure
result_exists: *Result /= Void*

infix *"/"* (*other*: ***like*** *Current*): ***like*** *Current*
-- Division by *other*
require
other_exists: *other /= Void*;
good_divisor: *divisible* (*other*)
deferred
ensure
result_exists: *Result /= Void*

infix *"^"* (*other*: *NUMERIC*): *NUMERIC*
-- Current object to the power *other*
require
other_exists: *other /= Void*;
good_exponent: *exponentiable* (*other*)
deferred
ensure
result_exists: *Result /= Void*

prefix *"+"*: ***like*** *Current*
-- Unary plus
deferred
ensure
result_exists: *Result /= Void*

prefix *"–"*: ***like*** *Current*
-- Unary minus
deferred
ensure
result_exists: *Result /= Void*

5.4 Class *HASHABLE*

indexing

description: *"Values that may be hashed into an integer index, for use as keys in hash tables."*

note: *"In general, the default value of a type (e.g. 0 for integers, void for references) may not be hashed."*

deferred class interface

HASHABLE

feature -- Access

hash_code: *INTEGER*
-- Hash code value
require
hashable: *is_hashable*
deferred
ensure
good_hash_value: ~~*Result >*~~ *Result >=* *0*

feature -- Status report

is_hashable: *BOOLEAN*
-- May current object be hashed?
-- (True if it is not its type's default.)
ensure
ok_if_not_default: *Result* ***implies*** (*Current /= default*)

end

5.3 Class *COMPARABLE*

indexing

description: *"Objects that may be compared according to a total order relation"*;

note: *"The basic operation is `<' (less than); others are defined in terms of this operation and `is_equal'."*

deferred class interface

COMPARABLE

feature -- Comparison

infix *"<"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object less than *other*?
require
other_exists: *other* /= *Void*
deferred
ensure
asymmetric: *Result* ***implies not*** (*other* < *Current*)

infix *"<="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object less than or equal to *other*?
require
other_exists: *other* /= *Void*
ensure
definition: *Result* = (*Current* < *other*) ***or*** *is_equal* (*other*);

infix *">="* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than or equal to *other*?
require
other_exists: *other* /= *Void*
ensure
definition: *Result* = (*other* <= *Current*)

infix *">"* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is current object greater than *other*?
require
other_exists: *other* /= *Void*
ensure
definition: *Result* = (*other* < *Current*)

is_equal (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* attached to an object ~~of the same type~~considered equal
~~-- as current object and identical to it?~~
-- to current object?
-- (Redefined from *GENERAL*.)
require
other_not_void: *other* /= *Void*
ensure
symmetric: *Result* ***implies*** *other*.*is_equal* (*Current*);
consistent: *standard_is_equal* (*other*) ***implies*** *Result*;
trichotomy: *Result* = (***not*** (*Current* < *other*) ***and not*** (*other* < *Current*))

max (*other*: ***like*** *Current*): ***like*** *Current*
-- The greater of current object and *other*
require
other_exists: *other* /= *Void*
ensure
current_if_not_smaller: (*Current* >= *other*) ***implies*** (*Result* = *Current*)
other_if_smaller: (*Current* < *other*) ***implies*** (*Result* = *other*)

min (*other*: ***like*** *Current*): ***like*** *Current*
-- The smaller of current object and *other*
require
other_exists: *other* /= *Void*
ensure
current_if_not_greater: (*Current* <= *other*) ***implies*** (*Result* = *Current*)
other_if_greater: (*Current* > *other*) ***implies*** (*Result* = *other*)

three_way_comparison (*other*: ***like*** *Current*): *INTEGER*)
-- If current object equal to *other*, 0; if smaller,
-- –1; if greater, 1.
require
other_exists: *other* /= *Void*
ensure
equal_zero: (*Result* = *0*) = *is_equal* (*other*);
smaller_negative: (*Result* = *–1*) = (*Current* < *other*);
greater_positive: (*Result* = *1*) = (*Current* > *other*)

invariant

irreflexive_comparison: ***not*** (*Current* < *Current*)

end

5.2 Class *ANY*

indexing

description: *"Project–wide universal properties. This class is an ancestor to all developer-written classes. ANY inherits from GENERAL and may be customized for individual projects or teams."*

class interface

ANY

end

frozen *standard_equal* (*some*: *GENERAL; other*: ***like*** *some*): *BOOLEAN*
-- Are *some* and *other* either both void or attached to
-- field–by-field identical objects of the same type?
-- Always uses the default object comparison criterion.
ensure
definition: *Result* = (*some* = *Void* ***and*** *other* = *Void*) ***or else*** ((*some* /= *Void* ***and*** *other* /= *Void*) ***and then*** *some*.*standard_is_equal* (*other*))

frozen *standard_is_equal* (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* attached to an object of the same type as
-- current object, and field–by-field identical to it?
require
other_not_void: *other* /= *Void*
ensure
~~*only_if_same_type*:~~ *same_type*: *Result* ***implies*** *same_type* (*other*);
symmetric: *Result* ***implies*** *other*.*standard_is_equal* (*Current*)

feature -- Duplication

frozen *clone* (*other*: *GENERAL*): ***like*** *other*
-- Void if *other* is void; otherwise new object
-- ~~identical~~ equal to *other*.
ensure
equal: *equal* (*Result*, *other*)

copy (*other*: ***like*** *Current*)
~~-- Copy the structure representing by *other* onto~~
~~-- that represented by current object.~~
-- Update current object using fields of object attached
-- to *other*, so as to yield equal objects.
require
other_not_void: *other* /= *Void;*
type_identity: *same_type* (*other*)
ensure
is_equal: *is_equal* (*other*)

frozen *deep_clone* (*other*: *GENERAL*): ***like*** *other*
-- Void if *other* is void: otherwise, new object structure
-- recursively duplicated from the one attached to *other*
ensure
deep_equal: *deep_equal* (*other*, *Result*)

frozen *standard_clone* (*other*: *GENERAL*): ***like*** *other*
-- Void if *other* is void; otherwise new object
-- field–by-field identical to *other*.
-- Always uses the default copying semantics.
ensure
equal: *standard_equal* (*Result*, *other*)

frozen *standard_copy* (*other*: ***like*** *Current*)
-- Copy every field of *other* onto corresponding field
-- of current object.
require
other_not_void: *other* /= *Void;*
type_identity: *same_type* (*other*)
ensure
is_standard_equal: *standard_is_equal* (*other*)

feature -- Basic operations

frozen *default*: ***like*** *Current*
-- Default value of current type

frozen *default_pointer*: *POINTER*
-- Default value of type *POINTER*
-- (Avoid the need to write *p*.*default* for some *p*
-- of type *POINTER*.)
ensure
Result = *Result*.*default*

default_rescue
-- Handle exception if no Rescue clause.
-- (Default: do nothing.)

frozen *do_nothing*
-- Execute a null action.

frozen *Void*: *NONE*
-- Void reference

feature -- Output

io: *STD_FILES*
-- Handle to standard file setup

out: *STRING*
-- New string containing terse printable representation
-- of current object

print (*some*: *GENERAL*)
-- Write terse external representation of *some* on
-- standard output.

frozen *tagged_out*: *STRING*
-- New string containing printable representation of
-- current object, each field preceded by its attribute
-- name, a colon and a space.

invariant

reflexive_equality: *standard_is_equal* (*Current*);

~~*reflexive_equality*: *is_standard_equal* (~~*reflexive_conformance*: *conforms_to* (*Current*);

involutive_object_id: *id_object* (*object_id*) = *Current*

end

5 SHORT FORMS OF REQUIRED CLASSES

5.1 Class *GENERAL*

indexing

description: *"Platform–independent universal properties. This class is an ancestor to all developer-written classes."*

class interface

GENERAL

feature -- Access

generating_type: *STRING*
-- Name of current object's generating type
-- (type of which it is a direct instance)

generator: *STRING*
-- Name of current object's generating class
-- (base class of the type of which it is a direct instance)

id_object (*id*: *INTEGER*): *ANY*
-- Object for which *object_id* has returned *id*;
-- void if none.

object_id: *INTEGER*
-- Value identifying current object uniquely;
-- meaningful only for reference types.

stripped (*other*: *GENERAL*): ***like*** *other*
-- New object with fields copied from current object,
-- but limited to attributes of type of *other*.
require
conformance: *conforms_to* (*other*)
ensure
stripped_to_other: *Result*.*same_type* (*other*)

feature -- Status report

frozen conforms_to (*other*: *GENERAL*): *BOOLEAN*
-- Does type of current object conform to type
-- of *other* (as per *Eiffel*: *The Language*, chapter13)?
require
other_not_void: *other /= Void*
~~***ensure***~~
~~*reflexive*: *is_equal* (*other*) ***implies*** *Result*~~

frozen same_type (*other*: *GENERAL*): *BOOLEAN*
-- Is type of current object identical to type of *other*?
require
other_not_void: *other /= Void*
ensure
definition: *Result* = (*conforms_to* (*other*) ***and*** *other*.*conforms_to* (*Current*))

feature -- Comparison

frozen *deep_equal* (*some*: *GENERAL*; *other*: ***like*** ~~*Current*~~): *some*): *BOOLEAN*
-- Are *some* and *other* either both void
-- or attached to isomorphic object structures?
ensure
shallow_implies_deep: *standard_equal* (*some*, *other*) ***implies*** *Result;*
same_type: *Result* ***implies*** *some*.*same_type* (*other*);
symmetric: *Result* ***implies*** *deep_equal* (*other*, *some*)

frozen *equal* (*some*: *GENERAL*; *other*: ***like*** ~~*Current*~~): *some*): *BOOLEAN*
-- Are *some* and *other* either both void or attached
-- to ~~identical~~ objects ~~of the same type?~~considered equal?
ensure
definition: *Result* = (*some* = *Void* ***and*** *other* = *Void*) ***or else*** ((*some /= Void* ***and*** *other /= Void*) ***and then*** *some*.*is_equal* (*other*));

is_equal (*other*: ***like*** *Current*): *BOOLEAN*
-- Is *other* attached to an object ~~of the same type~~considered equal
~~-- as current object and identical to it?~~
-- to current object?
require
other_not_void: *other /= Void*
ensure
consistent: *standard_is_equal* (*other*) ***implies*** *Result;*
same_type: *Result implies same_type* (*other*);
symmetric: *Result* ***implies*** *other*.*is_equal* (~~*Current*~~);*Current*)
~~*consistent*: *standard_is_equal* (*other*) ***implies*** *Result*~~