64 INDEX §8.3

Void GENERAL 13 tagged_out GENERAL 13 Void_attached_to_expanded **EXCEPTIONS 44** tailVoid_call_target STRING 36 **EXCEPTIONS 44** three_way_comparison CHARACTER 20 wipe_out COMPARABLE 15 STRING 36 DOUBLE 28 writable_if_extendible INTEGER 23 FILE 41 REAL 25 STRING 35 zero to_boolean DOUBLE 28 STRING 36 INTEGER 22 to_c NUMERIC 17 ARRAY 33 REAL 25 Symbols to_double STRING 36 "@" ARRAY 32 $to_integer$ STRING 36 to_lower STRING 36 to_next_line FILE 40 STD_FILES 38 to real STRING 36 to_upper STRING 37 truncated_to_integer DOUBLE 29 REAL 26 $truncated_to_real$ DOUBLE 29 upper ARRAY 32 valid_index ARRAY 32

STRING 35

§8.3 INDEX 63

put_substring STRING 36 raise	REAL 26 Routine_failure EXCEPTIONS 44
EXCEPTIONS 44 read character	same_type
FILE 41 STD_FILES 38	GENERAL 12 set_error_default STD_FILES 38
read_double FILE 41 STD_FILES 38	set_item BOOLEAN_REF 47 CHARACTER REF 48
read_integer FILE 41 STD_FILES 38 read_line	DOUBLE_REF 49 INTEGER_REF 50 POINTER_REF 51 REAL_REF 52
FILE 41 STD_FILES 38	set_output_default STD_FILES 38
read_real FILE 41 STD_FILES 38	sign DOUBLE 28
read_stream FILE 41 STD_FILES 38	INTEGER 22 REAL 25 standard_clone
read_word FILE 41	GENERAL 13 standard_copy
REAL 25	GENERAL 13
Real_bits PLATFORM 46	standard_default STD_FILES 38
REAL_REF 52 remake	standard_equal GENERAL 12
STRING 34 remove	standard_is_equal GENERAL 13
STRING 36	STD_FILES 38
resize ARRAY 33 STRING 36	STORABLE 42 STRING 34
retrieved STORABLE 42	stripped GENERAL 12
right_adjust STRING 36	substring STRING 37
rounded DOUBLE 29	substring_index STRING 34

62 INDEX §8.3

non_negative_count	POINTER_REF 51
ARRAY 33 STRING 37	Postcondition EXCEPTIONS 44
No_more_memory EXCEPTIONS 44	Precondition EXCEPTIONS 44
NUMERIC 17	<i>prefix "not"</i> BOOLEAN 19
object_id GENERAL 12 occurrences STRING 34	prefix "-" DOUBLE 30 INTEGER 23 NUMERIC 17 REAL 27
one DOUBLE 28 INTEGER 22 NUMERIC 17 REAL 25	prefix "+" DOUBLE 29 INTEGER 23 NUMERIC 17 REAL 26
open_read FILE 40	<i>print</i> GENERAL 13
open_read_append FILE 40	<i>put</i> ARRAY 33 STRING 36
open_read_write FILE 40 open_write FILE 40	put_boolean FILE 41 STD_FILES 38
FILE 40 out BOOLEAN 19	<i>put_character</i> FILE 41 STD_FILES 38
CHARACTER 21 DOUBLE 30 GENERAL 13 INTEGER 24	put_double FILE 41 STD_FILES 38
POINTER 31 REAL 27 STRING 37	<i>put_integer</i> FILE 41 STD_FILES 38
output STD_FILES 38	put_new_line STD_FILES 38 put_real
PLATFORM 46	FILE 41 STD_FILES 38
POINTER 31 Pointer_bits PLATFORM 46	<pre>put_string FILE 41 STD_FILES 38</pre>

§8.3 INDEX 61

is_writable FILE 40	make_from_array ARRAY 32
item BOOLEAN_REF 47	make_from_string STRING 34
CHARACTER_REF 48 DOUBLE_REF 49	make_open_append FILE 39
INTEGER_REF 50 POINTER_REF 51 REAL_REF 52	make_open_read FILE 39
item ARRAY 32	make_open_read_write FILE 39
STRING 34	make_open_write FILE 39
last_character FILE 40 STD_FILES 38 last_double FILE 40 STD_FILES 38 last_integer	CHARACTER 20 COMPARABLE 15 DOUBLE 28 INTEGER 22 REAL 25 STRING 35
FILE 40 STD_FILES 38	Maximum_character_code PLATFORM 46
last_real FILE 40	<i>Maximum_integer</i> PLATFORM 46
STD_FILES 38	MEMORY 43
last_string FILE 40 STD_FILES 38	min CHARACTER 20 COMPARABLE 15
left_adjust STRING 36	DOUBLE 28 INTEGER 22 REAL 25
Loop_invariant EXCEPTIONS 44	STRING 35
Loop_variant EXCEPTIONS 44	<i>Minimum_character_code</i> PLATFORM 46
lower ARRAY 32	<i>Minimum_integer</i> PLATFORM 46
make ARRAY 32	name FILE 39
FILE 39 STRING 34	name_exists FILE 41
make_create_read_write FILE 39	name_not_empty FILE 41

60 INDEX §8.3

NUMERIC 17 REAL 26	REAL 25 STRING 35
infix "*"	infix "@"
DOUBLE 29	STRING 34
INTEGER 23	infix "\"
NUMERIC 17	INTEGER 23
REAL 26	
infix "+"	infix "^"
DOUBLE 29	DOUBLE 29
INTEGER 23	INTEGER 23
NUMERIC 17	NUMERIC 17
REAL 26	REAL 26
infix "/"	input
DOUBLE 29	STD_FILES 38
INTEGER 23	insert
NUMERIC 17	STRING 36
REAL 26	insert_character
infix "//"	STRING 36
INTEGER 23	INTEGER 22
infix "<"	Integer_bits PLATFORM 46
CHARACTER 20 COMPARABLE 15	
DOUBLE 28	INTEGER_REF 50
INTEGER 22	io
REAL 25	GENERAL 13
STRING 35	is_closed
	FILE 40
infix "<=" CHARACTER 20	is_developer_exception
COMPARABLE 15	EXCEPTIONS 44
DOUBLE 28	
INTEGER 22	is_equal
REAL 25	ARRAY 32
STRING 35	COMPARABLE 15 GENERAL 12
	STRING 34
infix ">" CHARACTER 20	
COMPARABLE 15	is_open_read FILE 40
DOUBLE 28	
INTEGER 22	is_open_write
REAL 25	FILE 40
STRING 35	is_plain_text
	FILE 40
infix ">=" CHARACTER 20	is_readable
COMPARABLE 15	FILE 40
DOUBLE 28	is_signal
INTEGER 22	EXCEPTIONS 44
II I I I I I I I I I I I I I I I I I I	LACLI HOND TT

§8.3 INDEX 59

enter	
ARRAY 32	HASHABLE 16
entry	hash_code
ARRAY 32	BOOLEAN 19
equal	BOOLEAN_REF 47
GENERAL 12	CHARACTER 20 CHARACTER_REF 48
error	DOUBLE 28
STD_FILES 38	DOUBLE_REF 49
exception	HASHABLE 16
EXCEPTIONS 44	INTEGER 22
EXCEPTIONS 44	INTEGER_REF 50 POINTER 31
exists	POINTER_REF 51
FILE 40	REAL 25
exponentiable	REAL_REF 52
DOUBLE 29	STRING 34
INTEGER 23	head
NUMERIC 17	STRING 36
REAL 26	id_object
	GENERAL 12
FILE 39	Incorrect_inspect_value
fill	EXCEPTIONS 44
STRING 36	independent_store
floor	STORABLE 42
DOUBLE 29	index_of
REAL 26	STRING 34
force	infix "and then"
ARRAY 32	BOOLEAN 19
from_c	infix "and"
STRING 34	BOOLEAN 19
full_collect	<i>infix "implies"</i> BOOLEAN 19
MEMORY 43	infix "or else"
GENERAL A	BOOLEAN 19
GENERAL 12	infix "or"
general_store	BOOLEAN 19
STORABLE 42	infix "xor"
generating_type	BOOLEAN 19
GENERAL 12	infix "-"
generator	DOUBLE 29
GENERAL 12	INTEGER 23

58 INDEX §8.3

BOOLEAN_REF 47	STRING 34
ceiling DOUBLE 29	deep_clone GENERAL 13
REAL 26 change_name	deep_equal GENERAL 12
FILE 40 CHARACTER 20	default GENERAL 13
Character_bits PLATFORM 46	default_output STD_FILES 38
CHARACTER_REF 48	default_pointer GENERAL 13
Check_instruction EXCEPTIONS 44	default_rescue GENERAL 13
Class_invariant EXCEPTIONS 44	delete FILE 40
clone GENERAL 13	developer_exception_name EXCEPTIONS 44
close FILE 40	die EXCEPTIONS 44
code CHARACTER 20	dispose FILE 40
collecting MEMORY 43	MEMORY 43
collection_off MEMORY 43	<i>divisible</i> DOUBLE 29 INTEGER 23
collection_on MEMORY 43	NUMERIC 17 REAL 26
command_name ARGUMENTS 45	DOUBLE 28 Double_bits
COMPARABLE 15	PLATFORM 46
conforms_to	DOUBLE_REF 49
GENERAL 12 consistent_size ARRAY 33	do_nothing GENERAL 13
copy ARRAY 33 GENERAL 13	empty FILE 39 STRING 35
STRING 37 count	empty_definition STRING 37
ARRAY 32 FILE 39	end_of_file FILE 39

§8.1 INDEX 57

8 INDEX

[This Index is not part of the Standard.]

8.1

This Index indicates the page of definition of every class and feature appearing in the Required Flatshort Forms of section 5

8.2

Following the standard Eiffel conventions, feature names appear in lower-case italics and class names, when making up index entries, in UPPER-CASE ITALICS. Operator functions appear under *prefix* and *infix*; for example division appears under *infix* "/". This also applies to boolean operators, which appear under *infix* "and", infix "and then" and so on.

8.3

In a class entry, the class appears in *UPPER-CASE ITALICS*. Each reference to a feature name is followed by the name of the class or classes in which it is available, each with the corresponding page. To avoid any confusion with occurrences of the class name in its other role – as an index entry pointing to the beginning of the class specification – the class name in this case appears in UPPER-CASE ROMAN.

abs

DOUBLE 29
INTEGER 23
REAL 26

ANY 14

append_boolean
STRING 35

append_character
STRING 35

append_double
STRING 35

append_integer
STRING 35

append_integer
STRING 35

append_real
STRING 35

append_string
STRING 35

argument
ARGUMENTS 45

ARGUMENTS 45

argument_count
ARGUMENTS 45

ARRAY 32

assertion_violation
EXCEPTIONS 44

basic_store STORABLE 42 BOOLEAN 19 Boolean_bits

PLATFORM 46

	read*character, read*double, read*integer, read*line, read*real, read*stream, read*word, to_next*line. (See 5.14, page 38, and 5.15, page 39.)
7.17	• Addition to <i>EXCEPTIONS</i> of a procedure <i>die</i> that terminates the execution cleanly with a given exit status, without triggering an exception. (See page 44.)
7.18	• In class <i>STRING</i> , replacement of the <i>adapt</i> function by a more convenient procedure <i>make_from_string</i> which descendants of the class can use to initialize a string-like object from a manifest string, as in !! <i>t.make_from_string</i> ("THIS STRING"), where the type of <i>t</i> is a descendant of <i>STRING</i> . (See page 34.)
7.19	• Similarly, addition to <i>ARRAY</i> of a procedure <i>make_from_array</i> allowing initialization from a manifest array, as in !! <i>a.make_from_array</i> (<< <i>a</i> , <i>b</i> , <i>c</i> , <i>d</i> >>).
7.20	• Removal from <i>STRING</i> of a number of features which some committee members judged too specialized: <i>mirror</i> , <i>mirrored</i> , <i>share</i> , <i>shared_with</i> , <i>item_code</i> , <i>has</i> , <i>prepend</i> , <i>set</i> , <i>prune</i> , <i>prune_all</i> . Renaming of <i>replace_substring</i> to <i>put_substring</i> . Removal of <i>fill_blanks</i> , replaced by <i>fill</i> (applying to an arbitrary character). Change of the result type of <i>out</i> to <i>STRING</i> (rather than <i>like Current</i>).

7.4	 Addition of a number of assertion clauses to the features of GENERAL, in particular to specify more precisely the semantics of equality, copying, cloning and conformance.
7.5	• Addition to <i>GENERAL</i> of a function <i>stripped</i> such that <i>stripped</i> (a) is a clone of the current object limited to the fields that apply to a's dynamic type. As a result, the old semantics of copying and equality mentioned in 7.2 may now be achieved through calls such as a.copy (b.stripped (a)) and equal (a, b.stripped (a)). (See page 12.)
7.6	 Addition to GENERAL of object_id and id_object to allow unique identification of objects. (See page 12.)
7.7	• In class <i>PLATFORM</i> , removal of the assumption that <i>Character_bits</i> , <i>Integer_bits</i> , <i>Real_bits</i> and <i>Double_bits</i> are constants. This does not introduce any incompatibility with earlier uses except if they relied on the specific numerical values. (See page 46.)
7.8	• Removal of <i>PLATFORM</i> from the universal inheritance hierarchy; <i>PLATFORM</i> is no longer a parent of <i>ANY</i> and hence an ancestor of every class, and has no particular language-defined role; classes that need its facilities must name it explicitly among their proper ancestors. This is actually a language change. (See section 4, page 10.)
7.9	 Addition to PLATFORM of features Maximum_integer, Minimum_integer, Maximum_ character_code and Minimum_character_code. (See page 46.)
7.10	• Addition to <i>COMPARABLE</i> of <i>min</i> and <i>max</i> functions and of a three-way comparison function, <i>three_way_comparison</i> , which returns 0, –1 or 1. (See page 15.)
7.11	 Addition to the arithmetic basic classes of functions abs and sign (the latter defined in terms of three_way_comparison). Addition to REAL and DOUBLE of floor, ceiling, rounded and integer_part. Addition to DOUBLE of real_part. (See page 23 and following.)
7.12	 Addition of inheritance links making all basic classes (<i>INTEGER</i> and so on) heirs of <i>HASHABLE</i>, so that it is now possible to hash any object. (See section 4, page 10.) Removal of function <i>is_hashable</i> and the corresponding preconditions.
7.13	 Addition to ARRAY of features enter and entry as redefinable synonyms to put and item (or infix "@"), the latter becoming frozen. (See page 32.)
7.14	 Addition to STORABLE of a procedure independent_store which produces machine-independent representations of object structures. (See page 42.)
7.15	 Addition of a few features to class FILE describing file opening and opening modes (such as read-only or read-write). In earlier presentations the corresponding class was UNIX_FILE. The new class is very similar but removes any Unix-specific aspect. (See 5.15, page 39.)
7.16	• Changes of names in class STD_FILES and FILE: for consistency with the usual Eiffel naming style, underscores were added and abbrevations were expanded. In the following list (which uses the order of appearance of the features in STD_FILES), the added underscores appear as * and the added letters appear in bold italics: last*character, last*double, last*real, last*integer, last*string, put*boolean, put*character, put*double, put*integer, put_new*line, put*real, put*string,

7.1

7.2

7.3

6.5 Intermediate corrections

During the time when a vintage is in effect, minor corrections may prove necessary, due for example to typographical errors in the current version of this Standard or to inconsistencies discovered by users or implementors of Eiffel. In such a case the chairman of the Library Committee of NICE may, at his discretion, submit a motion covering one or more revisions. To be approved, such motions shall require a unanimous vote of the Library Committee, with the possible exception of any member who has notified the chairman of an absence of more than one month. If approved, such a revision shall receive a revision level and shall give rise to a modified Kernel Library Standard, identified as "Vintage year Level revision_level". The modifications shall be integrated into the following year's vintage.

6.6 Eiffel Kernel Supplier requirements

Any provider of an Eiffel environment must make the following information available to any NICE member:

- Vintage and revision level currently supported.
- Any features not supported. (It is not permitted to have a non-supported class.)
- List of classes needed by kernel classes, but not in the kernel, hereafter referred to as para-kernel classes.
- Full inheritance hierarchy of kernel and para-kernel classes.
- List of names of features (immediate or inherited) that appear in the provider's kernel classes but not in this Standard.

7 APPENDIX B: DIFFERENCES

[This Appendix is not part of the Standard.]

The following differences exist between this Standard and earlier presentations of the Kernel Library:

- Addition to *GENERAL* of a query *default* which returns the default value of the type of the current object. This also addresses the need to obtain the default value for type *POINTER*; for convenience, since *POINTER* has no manifest constant, a query *default_pointer* has also been included. (See page 13.)
- Adaptation of the semantics of *copy* and equality features (*equal*, *is_equal* and their *standard_* versions) so that the result is true if and only if the objects are truly identical, and in particular have the same type. This implies a language change too; the previous definition was non-symmetric so that *a.copy* (*b*) and *equal* (*a, b*) only applied to the fields corresponding to the attributes of *a*'s type. The earlier effect can still be achieved through function *stripped*, as explained next in 7.5. (See page 12.)
- Addition to *GENERAL* of a frozen feature *same_type* which specifies conformance both ways. Addition of the requirement that *conforms_to* is frozen too. (See page 12.)

6 APPENDIX A: THE KERNEL STANDARDIZATION PROCESS

[This Appendix is not part of the Standard.]

6.1 Why plan a process?

The Eiffel Kernel Library cannot be specified for eternity. Ideas willcome up for new classes and features; ways will be found to do thingsbetter. The evolution process must be fast enough to enable Eiffel users to benefit from this flow of ideas and avoid technical obsolescence, but orderly enough to protect their existing investments and modes of operation.

6.2 Cycle time

A revision every ten to fifteen years, as has occurred for programming language standards (Fortran, C and Ada are examples) is not appropriate for the Eiffel Kernel Library. It would foster complacency most of the time, and major upheavals when a revision is finally brought into effect. A yearly process, similar to the upgrading of wines, car models and stable software products, provides the right pace of change.

6.3 Vintages

Each revision of this Standard describes a **vintage** of the Eiffel Library Kernel Standard. The present version is vintage 1995.

6.4 Yearly schedule

The following deadlines apply to year *year*:

- 6.4.1 1 January: Vintage *year* takes effect.
- 1 April: first permitted date for starting discussions on Vintage *year+1* in NICE's Library Committee. (1 January to 31 March is acooling-off period.)
- 1 May: first permitted date for submitting formal proposals to the Library Committee for Vintage *year* + 1.
- 1 July: last permitted date for submitting initial proposals for Vintage *year* + 1.
- 1 September: last permitted date for submitting final proposals (which may result from merging of several proposals) for Vintage *year* + 1.
- 1 October: last date by which the Committee may have defined Vintage year +1.

This schedule is applicable starting with vintage 96. For the present vintage (95), the first, the date of applicability is 1 July 1995.

5.26 Class *REAL_REF*

```
indexing
   description: "Reference class for REAL"
class interface
   REAL_REF
feature -- Access
   item: REAL
         -- Real value
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set\_item\ (r: REAL)
          -- Make r the associated real value.
          item\_set: item = r
end
```

5.25 Class POINTER_REF

```
indexing
   description: "Reference class for POINTER"
class interface
   POINTER_REF
feature -- Access
   item: POINTER
         -- Pointer value
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set_item (p: POINTER)
          -- Make p the associated pointer value.
         item\_set: item = p
end
```

5.24 Class INTEGER_REF

```
indexing
   description: "Reference class for INTEGER"
class interface
   INTEGER_REF
feature -- Access
   item: INTEGER
         -- Integer value
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set_item (i: INTEGER)
          -- Make i the associated integer value.
         item\_set: item = i
end
```

5.23 Class *DOUBLE_REF*

```
indexing
   description: "Reference class for DOUBLE"
class interface
   DOUBLE_REF
feature -- Access
   item: DOUBLE
         -- Double value
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
         good_hash_value: Result >= 0
feature -- Element change
   set_item (d: DOUBLE)
         -- Make d the associated double value.
         item\_set: item = d
end
```

5.22 Class CHARACTER_REF

```
indexing
   description: "Reference class for CHARACTER"
class interface
   CHARACTER_REF
feature -- Access
   item: CHARACTER
         -- Character value
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
          good\_hash\_value : Result >= 0
feature -- Element change
   set_item (c: CHARACTER)
          -- Make c the associated character value.
         item\_set: item = c
end
```

5.21 Class BOOLEAN_REF

```
indexing
   description: "Reference class for BOOLEAN"
class interface
   BOOLEAN_REF
feature -- Access
   item: BOOLEAN\\
         -- Boolean value
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set_item (b: BOOLEAN)
          -- Make b the associated boolean value.
         item\_set: item = b
end
```

5.20 Class *PLATFORM*

```
indexing
                                                                     Minimum_integer: INTEGER
                                                                            -- Smallest supported value of type INTEGER
   description: "Platform-dependent properties. This class may be
     used as ancestor by classes needing its facilities"
                                                                            meaningful \colon Result <= 0
class interface
                                                                     Pointer_bits: INTEGER
   PLATFORM
                                                                            -- Number of bits in a value of type POINTER
feature -- Access
                                                                            meaningful: Result >= 1
   Boolean_bits: INTEGER
                                                                     Real_bits: INTEGER
          -- Number of bits in a value of type BOOLEAN
                                                                            -- Number of bits in a value of type REAL
      ensure
          meaningful: Result >= 1
                                                                            meaningful: Result >= 1
   Character_bits: INTEGER
                                                                  end
          -- Number of bits in a value of type CHARACTER
          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 <= 0
```

§5.19 CLASS ARGUMENTS 45

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 \le argument\_count
   command_name: STRING
          -- Name of command that started system execution
          definition: Result = argument(0)
feature -- Measurement
   argument\_count: INTEGER
         -- Number of arguments given to command that started
          -- system execution (command name does not count)
      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

reauire

applicable: is_developer_exception

feature -- Access

Check_instruction: INTEGER

-- Exception code for violated check

Class_invariant: INTEGER

-- Exception code for violated class invariant

Incorrect_inspect_value: INTEGER

-- Exception code for inspect value which is not one

-- of the inspect constants, if there is no Else_part

Loop_invariant: INTEGER

-- Exception code for violated loop invariant

Loop_variant: INTEGER

-- Exception code for non-decreased loop variant

No_more_memory: INTEGER

-- Exception code for failed memory allocation

Postcondition: INTEGER

-- Exception code for violated postcondition

Precondition: INTEGER

-- Exception code for violated precondition

 $Routine_failure: INTEGER$

-- Exception code for failed routine

 $Void_attached_to_expanded$: INTEGER

-- Exception code for attachment of void value

-- to expanded entity

Void_call_target: INTEGER

-- Exception code for violated check

feature -- Status report

 $assertion_violation \colon BOOLEAN$

- -- 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?

 $is_signal \colon BOOLEAN$

-- 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.

§5.17 CLASS *MEMORY* 43

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.

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
           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.
          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

§5.15 CLASS FILE 41

```
feature -- Input
                                                                           put_double (d: DOUBLE)
                                                                                   -- Write ASCII value of d at current position.
   read_character
          -- Read a new character.
                                                                                   extendible: extendible
          -- Make result available in last_character.
                                                                           put_integer (i: INTEGER)
       require
                                                                                   -- Write ASCII value of i at current position.
          readable: is_readable
                                                                               require
   read_double
                                                                                   extendible: extendible
          -- Read the ASCII representation of a new double
                                                                           put_real (r: REAL)
          -- from file. Make result available in last_double.
                                                                                   -- Write ASCII value of r at current position.
      require
                                                                               require
          readable: is_readable
                                                                                   extendible: extendible
   read_integer
                                                                           put_string (s: STRING)
          -- Read the ASCII representation of a new integer
                                                                                  -- Write s at current position.
          -- from file. Make result available in last_integer.
                                                                               require
                                                                                   extendible: extendible
          readable: is_readable
                                                                        invariant
   read_line
                                                                           name_exists: name /= Void;
          -- Read a string until new line or end of file.
          -- Make result available in laststring.
                                                                           name_not_empty: not name.empty;
          -- New line will be consumed but not part of last_string.
                                                                           writable_if_extendible: extendible implies is_writable
       require
          readable: is_readable
   read_real
          -- Read the ASCII representation of a new real
          -- from file. Make result available in last_real.
      require
          readable: is_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
          readable: is_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
```

```
exists: BOOLEAN
                                                                          open_read_append
          -- Does physical file exist?
                                                                                 -- Open file in read and write-at-end mode;
                                                                                 -- create it if it does not exist.
   is_closed: BOOLEAN
                                                                             require
          -- Is file closed?
                                                                                 is_closed: is_closed
   is_open_read: BOOLEAN
                                                                              ensure
          -- Is file open for reading?
                                                                                 exists: exists;
   is_open_write: BOOLEAN
                                                                                 open_read: is_open_read;
          -- Is file open for writing?
                                                                                 open_append: is_open_append
   is_plain_text: BOOLEAN
                                                                          open_read_write
          -- Is file reserved for text (character sequences)?
                                                                                 -- Open file in read and write mode.
   is readable: BOOLEAN
                                                                             require
          -- Is file readable?
                                                                                 is\_closed: is\_closed
      require
                                                                              ensure
          handle\_exists: exists
                                                                                 exists: exists;
   is_writable: BOOLEAN
                                                                                 open_read: is_open_read;
          -- Is file writable?
                                                                                 open_write: is_open_write
      require
          handle_exists: exists
                                                                                 -- Open file in write-only mode;
                                                                                 -- create it if it does not exist.
   last_character: CHARACTER
          -- Last character read by read_character
                                                                              ensure
                                                                                 exists: exists;
   last_double: DOUBLE
                                                                                 open_write: is_open_write
          -- Last double read by read_double
                                                                      feature -- Cursor movement
   last_integer: INTEGER
          -- Last integer read by read_integer
                                                                          to_next_line
                                                                                 -- Move to next input line.
   last_real: REAL
                                                                             require
          -- Last real read by read_real
                                                                                 readable: is_readable
   last_string: STRING
                                                                      feature -- Element change
          -- Last string read by read_line,
          -- read_stream, or read_word
                                                                          change_name (new_name: STRING)
feature -- Status setting
                                                                                 -- Change file name to new_name
                                                                             require
   close
                                                                                 not_new_name_void: new_name /= Void;
          -- Close file.
                                                                                 file_exists: exists
      require
          medium_is_open: not is_closed
                                                                                 name_changed: name • is_equal (new_name)
       ensure
          is_closed: is_closed
                                                                       feature -- Removal
   open_read
                                                                          delete
                                                                                 -- Remove link with physical file; delete physical
          -- Open file in read-only mode.
                                                                                 -- file if no more link.
      require
                                                                              require
          is_closed: is_closed
                                                                                 exists: exists
       ensure
          exists: exists;
                                                                          dispose
          open_read: is_open_read
                                                                                 -- Ensure this medium is closed when garbage-collected.
```

§5.15 CLASS FILE 39

5.15 Class FILE

```
indexing
                                                                           make_open_read (fn: STRING)
                                                                                   -- Create file object with fn as file name
   description: "Files viewed as persistent sequences of
                                                                                   -- and open file in read mode.
     characters"
                                                                               require
class interface
                                                                                   string_exists: fn /= Void;
                                                                                   string_not_empty: not fn.empty
   FILE
                                                                               ensure
                                                                                   exists: exists;
creation
                                                                                   open_read: is_open_read
   make (fn: STRING)
                                                                            make_open_read_write (fn: STRING)
           -- Create file object with fn as file name.
                                                                                   -- Create file object with fn as file name
                                                                                   -- and open file for both reading and writing.
       require
                                                                               require
          string_exists: fn /= Void;
                                                                                   string_exists: fn /= Void;
          string_not_empty: not fn • empty
                                                                                   string\_not\_empty \colon \boldsymbol{not} \ fn \bullet empty
          file_named: name • is_equal (fn);
                                                                                   exists: exists;
                                                                                   open_read: is_open_read;
          file_closed: is_closed
                                                                                   open_write: is_open_write
   make_create_read_write (fn: STRING)
                                                                            make_open_write (fn: STRING)
          -- Create file object with fn as file name
                                                                                   -- Create file object with fn as file name
          -- and open file for both reading and writing;
                                                                                   -- and open file for writing;
                                                                                   -- create it if it does not exist.
          -- create it if it does not exist.
                                                                               require
       require
                                                                                  string_exists: fn /= Void;
          string_exists: fn /= Void;
                                                                                   string_not_empty: not fn.empty
          string_not_empty: not fn• empty
                                                                                   exists: exists;
       ensure
                                                                                   open_write: is_open_write
           exists: exists;
                                                                       feature -- Access
          open_read: is_open_read;
                                                                           name: STRING
          open_write: is_open_write
                                                                                   -- File name
   make_open_append (fn: STRING)
                                                                        feature -- Measurement
          -- Create file object with fn as file name
                                                                            count: INTEGER
          -- and open file in append-only mode.
                                                                                   -- Size in bytes (0 if no associated physical file)
                                                                        feature -- Status report
       require
                                                                           empty: BOOLEAN
          string_exists: fn /= Void;
                                                                                   -- Is structure empty?
          string_not_empty: not fn.empty
                                                                            end_of_file: BOOLEAN
       ensure
                                                                                   -- Has an EOF been detected?
          exists: exists;
                                                                               require
                                                                                   opened: not is_closed
          open_append: is_open_append
```

indexing

5.14 Class STD_FILES

classes needing its facilities."

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

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 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_new_line
          -- Write line feed 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
   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 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.
   to_next_line
          -- Move to next input line on standard input.
```

§5.13 CLASS *STRING* 37

```
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 \cdot count
          -- same_characters: For every i in 1..count,
               item(i) = other \cdot item(i)
   substring (n1, n2: INTEGER): like Current
          -- Copy of substring containing all characters at indices
          -- between n1 and n2
       require
          meaningful\_origin: 1 \le n1;
          meaningful\_interval: n1 \le n2;
          meaningful\_end: n2 \le 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: STRING
          -- Printable representation
          -- (From GENERAL.)
          result_not_void: Result /= Void
invariant
   irreflexive_comparison: not (Current < Current);</pre>
   empty\_definition: empty = (count = 0);
   non\_negative\_count: count >= 0
end
```

```
fill (c: CHARACTER)
                                                                          right_adjust
                                                                                 -- Remove trailing white space.
       -- Replace every character with c.
   ensure
                                                                                 new_count: (count /= 0) implies (item (count) /= ' ')
       -- allblank: For every i in 1..count, item (i) = Blank
                                                                          tail (n: INTEGER)
head (n: INTEGER)
                                                                                 -- Remove all characters except for the last n;
       -- Remove all characters except for the first n;
                                                                                 -- do nothing if n >= count.
       -- do nothing if n >= count.
                                                                             require
   require
                                                                                 non\_negative\_argument: n >= 0
       non\_negative\_argument: n >= 0
   ensure
                                                                                new\_count: count = n \cdot min (old count)
       new\_count: count = n \cdot min (old count)
                                                                      feature -- Removal
       -- first\_kept: For every i in 1..n, item(i) = old item(i)
                                                                          remove (i: INTEGER)
insert (s: like Current; i: INTEGER)
                                                                                 -- Remove i-th character.
       -- Add s to the left of position i.
                                                                             require
                                                                                 index\_small\_enough: i \le count;
       string_exists: s /= Void;
                                                                                 index\_large\_enough: i > 0
       index\_small\_enough: i \le count;
       index\_large\_enough: i > 0
                                                                                new\_count: count = old count - 1
   ensure
                                                                          wipe_out
       new\_count: count = old count + s \cdot count
                                                                                 -- Remove all characters.
insert_character (c: CHARACTER; i: INTEGER)
       -- Add c to the left of position i.
                                                                                 empty_string: count = 0;
   ensure
                                                                                 wiped_out: empty
       new\_count: count = old count + 1
                                                                     feature -- Resizing
left_adjust
                                                                          resize (newsize: INTEGER)
       -- Remove leading white space.
                                                                                 -- Rearrange string so that it can accommodate
                                                                                 -- at least newsize characters.
                                                                                 -- Do not lose any previously entered character.
       new\_count: (count \neq 0) implies (item (1) \neq ')
put (c: CHARACTER; i: INTEGER)
                                                                                 new\_size\_non\_negative: newsize >= 0
       -- Replace character at position i by c.
                                                                      feature -- Conversion
   reauire
       good_key: valid_index (i)
                                                                          to_boolean: BOOLEAN
                                                                                 -- Boolean value;
   ensure
                                                                                 -- "true" yields true, "false" yields false
       insertion\_done: item(i) = c
                                                                                 -- (case-insensitive)
put_substring (s: like Current; start_pos, end_pos: INTEGER)
                                                                          to_double: DOUBLE
       -- Copy the characters of s to positions
                                                                                 -- "Double" value; for example, when applied to "123.0",
       -- start_pos .. end_pos.
                                                                                 -- will yield 123.0 (double)
   require
                                                                          to_integer: INTEGER
       string_exists: s /= Void;
                                                                                 -- Integer value;
       index_small_enough: end_pos <= count;</pre>
                                                                                 -- for example, when applied to "123", will yield 123
       order_respected: start_pos <= end_pos;</pre>
                                                                         to_lower
       index_large_enough: start_pos > 0
                                                                                 -- Convert to lower case.
   ensure
                                                                          to_real: REAL
       new\_count: count = old count + s \cdot count - end\_pos +
                                                                                 -- Real value;
         start\_pos - 1
                                                                                 -- for example, when applied to "123.0", will yield 123.0
```

§5.13 CLASS *STRING* 35

```
infix "<" (other: STRING): BOOLEAN
                                                                       min (other: like Current): like Current)
                                                                              -- The smaller of current object and other
       -- Is string lexicographically lower than other?
                                                                              -- (From COMPARABLE.)
       -- (False if other is void)
                                                                          require
       -- (From COMPARABLE.)
                                                                              other_exists: other /= Void
   require
                                                                          ensure
       other exists: other /= Void
                                                                              current_if_not_greater: (Current <= other) implies
                                                                                (Result = Current)
   ensure
                                                                              other_if_greater: (Current > other) implies (Result =
       asymmetric: Result implies not (other < Current)
                                                                                other)
infix "<=" (other: like Current): BOOLEAN
                                                                       three_way_comparison (other: like Current): INTEGER)
                                                                              -- If current object equal to other, 0; if smaller,
       -- Is current object less than or equal to other?
                                                                              ---1; if greater, 1.
       -- (From COMPARABLE.)
                                                                              -- (From COMPARABLE.)
   require
                                                                          require
                                                                              other_exists: other /= Void
      other_exists: other /= Void
                                                                          ensure
   ensure
                                                                              equal\_zero: (Result = 0) = is\_equal (other);
       definition: Result = (Current < other) or is_equal
                                                                              smaller: (Result = -1) = Current < other;
                                                                              greater\_positive: (Result = 1) = Current > other
infix ">=" (other: like Current): BOOLEAN
                                                                   feature -- Status report
       -- Is current object greater than or equal to other?
                                                                       empty: BOOLEAN
       -- (From COMPARABLE.)
                                                                              -- Is string empty?
   require
                                                                       valid_index (i: INTEGER): BOOLEAN
       other_exists: other /= Void
                                                                              -- Is i within the bounds of the string?
                                                                   feature -- Element change
   ensure
       definition: Result = (other <= Current)
                                                                       append_boolean (b: BOOLEAN)
                                                                              -- Append the string representation of b at end.
infix ">" (other: like Current): BOOLEAN
                                                                       append_character (c: CHARACTER)
       -- Is current object greater than other?
                                                                              -- Append c at end.
       -- (From COMPARABLE.)
                                                                          ensure
   require
                                                                              item\_inserted: item (count) = c
                                                                              one\_more\_occurrence: occurrences (c) = old
       other_exists: other /= Void
                                                                                (occurrences(c)) + 1
   ensure
                                                                              item_inserted: has (c)
       definition: Result = (other < Current)
                                                                       append_double (d: DOUBLE)
                                                                              -- Append the string representation of d at end.
max (other: like Current): like Current)
       -- The greater of current object and other
                                                                       append_integer (i: INTEGER)
                                                                              -- Append the string representation of i at end.
       -- (From COMPARABLE.)
                                                                       append_real (r: REAL)
   require
                                                                              -- Append the string representation of r at end.
       other_exists: other /= Void
                                                                       append_string (s: STRING)
   ensure
                                                                              -- Append a copy of s, if not void, at end.
       current_if_not_smaller: (Current >= other) implies
         (Result = Current)
                                                                              new\_count: count = old count + s \cdot count
       other_if_smaller: (Current < other) implies (Result =
                                                                              -- appended: For every i in 1..s. count,
                                                                                   item (old count + i) = s \cdot item (i)
```

5.13 Class STRING

```
indexing
                                                                      feature -- Access
                                                                          hash_code: INTEGER
   description: "Sequences of characters, accessible through
                                                                                 -- Hash code value
     integer indices in a contiguous range."
                                                                                 -- (From HASHABLE.)
class interface
                                                                              ensure
   STRING
                                                                                 good_hash_value: Result >= 0
creation
                                                                          index_of (c: CHARACTER; start: INTEGER): INTEGER
                                                                                 -- Position of first occurrence of c at or after start;
   frozen make (n: INTEGER)
                                                                                 -- 0 if none.
          -- Allocate space for at least n characters.
                                                                              reauire
       require
                                                                                 start_large_enough: start >= 1;
                                                                                 start_small_enough: start <= count
          non\_negative\_size: n >= 0
                                                                              ensure
       ensure
                                                                                 non\_negative\_result: Result >= 0;
          empty_string: count = 0
                                                                                 at\_this\_position: Result > 0 implies item (Result) = c;
                                                                                 -- none_before: For every i in start..Result, item (i) \neq c
   make_from_string (s: STRING)
                                                                                 -- zero_iff_absent:
          -- Initialize from the characters of s.
                                                                                 -- (Result = 0) = For every i in 1..count, item (i) /= c
          -- (Useful in proper descendants of class STRING,
                                                                          item (i: INTEGER): CHARACTER
          -- to initialize a string-like object from a manifest string.)
                                                                                 -- Character at position i
       require
                                                                             require
          string_exists: s/= Void
                                                                                 good_key: valid_index (i)
                                                                          substring_index (other: STRING; start: INTEGER): INTEGER
feature -- Initialization
                                                                                 -- Position of first occurrence of other at or after start;
   from_c (c_string: POINTER)
                                                                                 -- 0 if none.
          -- Reset contents of string from contents of c_string,
                                                                          infix "@" (i: INTEGER): CHARACTER
          -- a string created by some external C function.
                                                                                 -- Character at position i
       reauire
          C_string_exists: c_string \neq Void
                                                                                 good_key: valid_index (i)
                                                                      feature -- Measurement
   frozen remake (n: INTEGER)
                                                                          count: INTEGER
          -- Allocate space for at least n characters.
                                                                                 -- Actual number of characters making up the string
       require
                                                                          occurrences (c: CHARACTER): INTEGER
          non\_negative\_size: n >= 0
                                                                                 -- Number of times c appears in the string
       ensure
                                                                              ensure
          empty\_string: count = 0
                                                                                 non_negative_occurrences: Result >= 0
   make_from_string (s: STRING)
                                                                      feature -- Comparison
          -- Initialize from the characters of s.
                                                                          is_equal (other: like Current): BOOLEAN
          -- (Useful in proper descendants of class STRING,
                                                                                 -- Is string made of same character sequence as other?
          -- to initialize a string-like object from a manifest string.)
                                                                                 -- (Redefined from GENERAL.)
       require
                                                                             require
          string_exists: s/= Void
                                                                                 other_exists: other /= Void
```

§5.12 CLASS ARRAY 33

```
frozen put (v: like item; i: INTEGER)
           -- Replace i-th entry, if in index interval, by v.
          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
          no_low_lost: lower = minindex.min (old lower);
          no\_high\_lost: upper = maxindex \cdot 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
                                                                          frozen item (i: INTEGER): G
                                                                                  -- Entry at index i, if in index interval
   description: "Sequences of values, all of the same type or of a
     conforming one, accessible through integer indices in a
                                                                              require
     contiguous interval"
                                                                                 good_key: valid_index (i)
class interface
                                                                          frozen infix "@" (i: INTEGER): G
                                                                                  -- Entry at index i, if in index interval
   ARRAY[G]
                                                                              require
creation
                                                                                 good_key: valid_index (i)
   make (minindex, maxindex: INTEGER)
                                                                      feature -- Measurement
          -- Allocate array; set index interval to
          -- minindex .. maxindex; set all values to default.
                                                                          count: INTEGER
          -- (Make array empty if minindex > maxindex.)
                                                                                 -- Number of available indices
       ensure
                                                                          lower: INTEGER
          no\_count: (minindex > maxindex) implies (count = 0);
                                                                                  -- Minimum index
          count_constraint: (minindex <= maxindex) implies</pre>
                                                                          upper: INTEGER
            (count = maxindex - minindex + 1)
                                                                                  -- Maximum index
   make_from_array (a: ARRAY [G])
                                                                      feature -- Comparison
          -- Initialize from the items of a.
          -- (Useful in proper descendants of class ARRAY,
                                                                          is_equal (other: like Current): BOOLEAN
          -- to initialize an array-like object from a manifest array.)
                                                                                 -- Is array made of the same items as other?
                                                                                 -- (Redefined from GENERAL.)
feature -- Initialization
                                                                      feature -- Status report
   make (minindex, maxindex: INTEGER)
          -- Set index interval to minindex .. maxindex;
                                                                          valid_index (i: INTEGER): BOOLEAN
          -- reallocate if necessary; set all values to default.
                                                                                 -- Is i within the bounds of the array?
          -- (Make array empty if minindex > maxindex.)
                                                                       feature -- Element change
       ensure
                                                                          enter (v: G; i: INTEGER)
          no\_count: (minindex > maxindex) implies (count = 0);
                                                                                  -- Replace i-th entry, if in index interval, by v.
          count constraint: (minindex <= maxindex) implies
                                                                                 -- (Redefinable synonym for put.)
            (count = maxindex - minindex + 1)
                                                                              require
   make from array (a: ARRAY [G])
                                                                                 good_key: valid_index (i)
          -- Initialize from the items of a; reallocate if
          -- necessary. (Useful in proper descendants of
                                                                              ensure
          -- class ARRAY, to initialize an array-like object
                                                                                 inserted: item(i) = v
          -- from a manifest array.)
                                                                          force (v: like item; i: INTEGER)
feature -- Access
                                                                                 -- Assign item v to i-th entry.
                                                                                 -- Always applicable: resize the array if i falls out of
   entry (i: INTEGER): G
                                                                                 -- currently defined bounds; preserve existing items.
          -- Entry at index i, if in index interval
          -- (Redefinable synonym for item and infix "@".)
                                                                              ensure
       require
                                                                                 inserted: item (i) = v;
          good_key: valid_index (i)
                                                                                 higher_count: count >= old count
```

§5.11 CLASS POINTER 31

5.11 Class POINTER

```
indexing
   description: "References to objects meant to be exchanged with
     non-Eiffel software."
expanded class interface
   POINTER
feature -- Access
   hash_code: INTEGER
         -- Hash code value
         -- (From HASHABLE.)
      ensure
          good\_hash\_value : Result >= 0
feature -- Output
   out: STRING
          -- Printable representation of pointer value
          -- (From GENERAL.)
end
```

```
prefix "-": like Current
          -- Unary minus
          -- (From NUMERIC.)
          result_exists: Result /= Void
feature -- Output
   out: STRING
          -- Printable representation of double value
          -- (From GENERAL.)
invariant
   irreflexive_comparison: not (Current < Current);</pre>
   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
```

§5.10 CLASS DOUBLE 29

feature Status report	infix "*" (other: like Current): like Current
divisible (other: like Current): BOOLEAN	Product by <i>other</i>
May current object be divided by <i>other</i> ?	(From <i>NUMERIC</i> .)
(From <i>NUMERIC</i> .)	require
require	other_exists: other /= Void
other_exists: other /= Void	ensure
ensure	result_exists: Result /= Void
not_exact_zero: Result implies (other $\neq 0.0$)	infix "+" (other: like Current): like Current
exponentiable (other: NUMERIC): BOOLEAN	Sum with <i>other</i>
May current object be elevated to the power <i>other</i> ?	(From <i>NUMERIC</i> .)
(From <i>NUMERIC</i> .)	require
require	other_exists: other /= Void
other_exists: other /= Void	ensure
ensure	result_exists: Result /= Void;
safe_values: (other • conforms_to (0) or	commutative: equal (Result, other + Current)
(other • conforms_to (Current) and (Current >= 0 • 0))) implies Result	
•	infix "_" (other: like Current): like Current
feature Conversion	Result of subtracting <i>other</i>
ceiling: INTEGER	(From <i>NUMERIC</i> .)
Smallest integral value no smaller than current object	require
ensure	other_exists: other /= Void
result_no_smaller: Result >= Current;	ensure
close_enough: Result – Current < one	result_exists: Result /= Void
floor: INTEGER	infix "/" (other: like Current): like Current
Greatest integral value no greater than current object	Division by <i>other</i>
ensure	(From <i>NUMERIC</i> .)
result_no_greater: Result <= Current;	require
close_enough: Current - Result < one	other_exists: other /= Void;
rounded: INTEGER	good_divisor: divisible (other)
Rounded integral value	
ensure	ensure
definition: Result = $sign * ((abs + 0.5).floor)$	result_exists: Result /= Void
	infix "^" (other: like Current): like Current
truncated_to_integer: INTEGER Integer part (same sign, largest absolute	Current double to the power <i>other</i>
value no greater than current object's)	(From <i>NUMERIC</i> .)
	require
truncated_to_real: REAL Real part (same sign, largest absolute	<pre>other_exists: other /= Void;</pre>
value no greater than current object's)	<pre>good_exponent: exponentiable (other)</pre>
feature Basic operations	ensure
jeature basic operations	result_exists: Result /= Void
abs: like Current	prefix "+": like Current
Absolute value	Unary plus
ensure	(From <i>NUMERIC</i> .)
non_negative: Result >= 0;	ensure
same_absolute_value: (Result = Current)	result_exists: Result /= Void
Current	resuu_eaisis. Resuu /— voiu

5.10 Class DOUBLE

```
indexing
                                                                         infix ">=" (other: like Current): BOOLEAN
                                                                                -- Is current object greater than or equal to other?
   description: "Real values, double precision"
                                                                                -- (From COMPARABLE.)
expanded class interface
                                                                            require
   DOUBLE
                                                                                other_exists: other /= Void
feature -- Access
                                                                            ensure
                                                                                definition: Result = (other <= Current)
   hash_code: INTEGER
                                                                         infix ">" (other: like Current): BOOLEAN
          -- Hash code value
                                                                                -- Is current object greater than other?
          -- (From HASHABLE.)
                                                                                -- (From COMPARABLE.)
       ensure
                                                                            require
          good\_hash\_value : Result >= 0
                                                                                other_exists: other /= Void
   one: like Current
                                                                            ensure
          -- Neutral element for "*" and "/"
                                                                                definition: Result = (other < Current)
          -- (From NUMERIC.)
                                                                         max (other: like Current): like Current
       ensure
                                                                                -- The greater of current object and other
          Result_exists: Result /= Void;
                                                                                -- (From COMPARABLE.)
          value: Result = 1.0
                                                                            require
   sign: INTEGER
                                                                                other exists: other /= Void
          -- Sign value (0, --1 or 1)
                                                                            ensure
       ensure
                                                                                current_if_not_smaller: (Current >= other) implies
          three_way: Result = three_way_comparison (zero)
                                                                                  (Result = Current)
   zero: like Current
                                                                                other_if_smaller: (Current < other) implies (Result =
          -- Neutral element for "+" and "-"
          -- (From NUMERIC.)
                                                                         min (other: like Current): like Current
                                                                                -- The smaller of current object and other
          Result exists: Result /= Void;
                                                                                -- (From COMPARABLE.)
          value: Result = 0.0
                                                                            require
feature -- Comparison
                                                                                other_exists: other /= Void
   infix "<" (other: like Current): BOOLEAN
                                                                            ensure
          -- Is other greater than current double?
                                                                                current_if_not_greater: (Current <= other) implies
          -- (From COMPARABLE.)
                                                                                  (Result = Current)
       require
                                                                                other_if_greater: (Current > other) implies (Result =
                                                                                  other)
          other_exists: other /= Void
                                                                         three_way_comparison (other: like Current): INTEGER
                                                                                -- If current object equal to other, 0; if smaller,
          asymmetric: Result implies not (other < Current)
                                                                                -- −1; if greater, 1.
   infix "<=" (other: like Current): BOOLEAN
                                                                            require
          -- Is current object less than or equal to other?
                                                                                other_exists: other /= Void
          -- (From COMPARABLE.)
                                                                                -- (From COMPARABLE.)
       require
                                                                            ensure
          other_exists: other /= Void
                                                                                equal\_zero: (Result = 0) = is\_equal (other);
       ensure
                                                                                smaller: (Result = -1) = Current < other;
          definition: Result = (Current < other) or is_equal
                                                                                greater\_positive: (Result = 1) = Current > other
            (other);
```

§5.9 CLASS REAL

```
prefix "-": like Current
          -- Unary minus
          -- (From NUMERIC.)
          result_exists: Result /= Void
feature -- Output
   out: STRING
          -- Printable representation of real value
          -- (From GENERAL.)
invariant
   irreflexive_comparison: not (Current < Current);</pre>
   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
```

```
infix "*" (other: like Current): like Current
feature -- Status report
                                                                                   -- Product by other
    divisible (other: like Current): BOOLEAN
                                                                                   -- (From NUMERIC.)
          -- May current object be divided by other?
                                                                               require
          -- (From NUMERIC.)
                                                                                   other_exists: other /= Void
       require
          other_exists: other /= Void
                                                                                   result_exists: Result /= Void
       ensure
                                                                            infix "+" (other: like Current): like Current
          not\_exact\_zero: Result implies (other \neq 0.0)
                                                                                   -- Sum with other
    exponentiable (other: NUMERIC): BOOLEAN
                                                                                   -- (From NUMERIC.)
          -- May current object be elevated to the power other?
                                                                               require
          -- (From NUMERIC.)
                                                                                   other_exists: other /= Void
       require
                                                                               ensure
          other_exists: other /= Void
                                                                                   result_exists: Result /= Void;
       ensure
                                                                                   commutative: equal (Result, other + Current)
          safe_values: (other • conforms_to (0) or
             (other \cdot conforms\_to(Current)  and (Current >= 0 \cdot 0)))
                                                                            infix "-" (other: like Current): like Current
               implies Result
                                                                                   -- Result of subtracting other
                                                                                   -- (From NUMERIC.)
feature -- Conversion
                                                                               require
    ceiling: INTEGER
                                                                                   other_exists: other /= Void
          -- Smallest integral value no smaller than current object
                                                                               ensure
       ensure
                                                                                   result_exists: Result /= Void
          result_no_smaller: Result >= Current;
                                                                            infix "/" (other: like Current): like Current
          close_enough: Result - Current < one
                                                                                   -- Division by other
   floor: INTEGER
                                                                                   -- (From NUMERIC.)
          -- Greatest integral value no greater than current object
                                                                               require
       ensure
                                                                                   other_exists: other /= Void;
          result_no_greater: Result <= Current;</pre>
                                                                                   good_divisor: divisible (other)
          close_enough: Current - Result < one
    rounded: INTEGER
                                                                                   result_exists: Result /= Void
          -- Rounded integral value
                                                                            infix "^" (other: NUMERIC): DOUBLE
       ensure
                                                                                   -- Current real to the power other
          definition: Result = sign * ((abs + 0.5).floor)
                                                                                   -- (From NUMERIC.)
    truncated_to_integer: INTEGER
                                                                               require
          -- Integer part (same sign, largest absolute
                                                                                   other_exists: other /= Void;
          -- value no greater than current object's)
                                                                                   good_exponent: exponentiable (other)
feature -- Basic operations
                                                                                   result_exists: Result /= Void
    abs: like Current
                                                                            prefix "+": like Current
          -- Absolute value
                                                                                   -- Unary plus
       ensure
                                                                                   -- (From NUMERIC.)
          non\_negative: Result >= 0;
          same_absolute_value: (Result = Current) or (Result = -
                                                                                   result_exists: Result /= Void
```

§5.9 CLASS *REAL* 25

5.9 Class REAL

```
indexing
                                                                         infix ">=" (other: like Current): BOOLEAN
                                                                                -- Is current object greater than or equal to other?
   description: "Real values, single precision"
                                                                                -- (From COMPARABLE.)
expanded class interface
                                                                            require
   REAL
                                                                                other_exists: other /= Void
                                                                            ensure
feature -- Access
                                                                                definition: Result = (other <= Current)
   hash_code: INTEGER
                                                                         infix ">" (other: like Current): BOOLEAN
          -- Hash code value
                                                                                -- Is current object greater than other?
          -- (From HASHABLE.)
                                                                                -- (From COMPARABLE.)
       ensure
                                                                            require
          good\_hash\_value : Result >= 0
                                                                                other_exists: other /= Void
   one: like Current
                                                                            ensure
          -- Neutral element for "*" and "/"
                                                                                definition: Result = (other < Current)
          -- (From NUMERIC.)
                                                                         max (other: like Current): like Current
       ensure
                                                                                -- The greater of current object and other
          Result_exists: Result /= Void;
                                                                                -- (From COMPARABLE.)
          value: Result = 1.0
                                                                            require
   sign: INTEGER
                                                                                other exists: other /= Void
          -- Sign value (0, --1 or 1)
                                                                            ensure
       ensure
                                                                                current_if_not_smaller: (Current >= other) implies
          three_way: Result = three_way_comparison (zero)
                                                                                  (Result = Current)
   zero: like Current
                                                                                other_if_smaller: (Current < other) implies (Result =
          -- Neutral element for "+" and "-"
          -- (From NUMERIC.)
                                                                         min (other: like Current): like Current
                                                                                -- The smaller of current object and other
          Result exists: Result /= Void;
                                                                                -- (From COMPARABLE.)
          value: Result = 0.0
                                                                            require
feature -- Comparison
                                                                                other_exists: other /= Void
   infix "<" (other: like Current): BOOLEAN
                                                                            ensure
          -- Is other greater than current real?
                                                                                current_if_not_greater: (Current <= other) implies
          -- (From COMPARABLE.)
                                                                                  (Result = Current)
       require
                                                                                other_if_greater: (Current > other) implies (Result =
                                                                                  other)
          other_exists: other /= Void
                                                                         three_way_comparison (other: like Current): INTEGER
                                                                                -- If current object equal to other, 0; if smaller,
          asymmetric: Result implies not (other < Current)
                                                                                -- -1; if greater, 1.
   infix "<=" (other: like Current): BOOLEAN
                                                                                -- (From COMPARABLE.)
          -- Is current object less than or equal to other?
                                                                            reauire
          -- (From COMPARABLE.)
                                                                                other_exists: other /= Void
       require
                                                                            ensure
          other_exists: other /= Void
                                                                                equal\_zero: (Result = 0) = is\_equal (other);
       ensure
                                                                                smaller: (Result = -1) = Current < other;
          definition: Result = (Current < other) or is_equal
                                                                                greater\_positive: (Result = 1) = Current > other
            (other);
```

```
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
```

§5.8 CLASS INTEGER 23

three_way_comparison (other: like Current): INTEGER If current object equal to other, 0; if smaller,1; if greater, 1.	<pre>infix "-" (other: like Current): like Current Result of subtracting other (From NUMERIC.)</pre>
(From COMPARABLE.)	require
require	other_exists: other /= Void
other_exists: other /= Void	ensure
ensure	result_exists: Result /= Void
$equal_zero: (Result = 0) = is_equal (other);$	_
smaller: (Result = 1) = Current < other;	<pre>infix "/" (other: like Current): DOUBLE Division by other</pre>
$greater_positive: (Result = -I) = Current > other$	require
feature Status report	other_exists: other /= Void;
•	good_divisor: divisible (other)
divisible (other: like Current): BOOLEAN May appropriately be divided by ather?	ensure
 May current object be divided by <i>other</i>? (From <i>NUMERIC</i>.)	result_exists: Result /= Void
require	_
other_exists: other/= Void	infix "//" (other: like Current): like Current
ensure	 Integer division of Current by <i>other</i> (From "/" in <i>NUMERIC</i>.)
value: $Result = (other \neq 0)$	require
	other_exists: other /= Void;
exponentiable (other: NUMERIC): BOOLEAN	good_divisor: divisible (other)
May current object be elevated to the power <i>other</i> ?	ensure
(From NUMERIC.)	result_exists: divisible (other)
require	_ , ,
other_exists: other /= Void ensure	infix "\\" (other: like Current): like Current Remainder of the integer division of Current by other
safe_values: (other • conforms_to (Current) or	require
$(other \cdot conforms_{to}(0 \cdot 0) \ and \ (Current >= 0)))$	other_exists: other /= Void;
implies Result	good_divisor: divisible (other)
feature Basic operations	ensure
abs: like Current	result_exists: Result /= Void
Absolute value	infix "^" (other: NUMERIC): DOUBLE
ensure	Integer power of Current by other
non_negative: Result >= 0;	(From <i>NUMERIC</i> .)
same_absolute_value: (Result = Current) or (Result = -	require
Current)	other_exists: other /= Void;
infix "*" (other: like Current): like Current	good_exponent: exponentiable (other)
Product by <i>other</i>	ensure
(From <i>NUMERIC</i> .)	result_exists: Result /= Void
require	prefix "+": like Current
other_exists: other /= Void	Unary plus
infix "+" (other: like Current): like Current	(From <i>NUMERIC</i> .)
Sum with <i>other</i>	ensure
(From <i>NUMERIC</i> .)	result_exists: Result /= Void
require	prefix "_": like Current
other_exists: other /= Void	Unary minus
ensure	(From <i>NUMERIC</i> .)
result_exists: Result /= Void;	ensure
commutative: equal (Result, other + Current)	result_exists: Result /= Void

5.8 Class INTEGER

```
indexing
                                                                         infix "<=" (other: like Current): BOOLEAN</pre>
                                                                                -- Is current object less than or equal to other?
   description: "Integer values"
                                                                                -- (From COMPARABLE.)
expanded class interface
                                                                            require
                                                                               other_exists: other /= Void
   INTEGER
feature -- Access
                                                                                definition: Result = (Current < other) or is_equal
                                                                                  (other);
   hash_code: INTEGER is
                                                                         infix ">=" (other: like Current): BOOLEAN
          -- Hash code value
                                                                               -- Is current object greater than or equal to other?
          -- (From HASHABLE.)
                                                                                -- (From COMPARABLE.)
       ensure
                                                                            require
          good_hash_value: Result >= 0
                                                                                other_exists: other /= Void
                                                                            ensure
   one: like Current
                                                                               definition: Result = (other <= Current)
          -- Neutral element for "*" and "/"
                                                                         infix ">" (other: like Current): BOOLEAN
          -- (From NUMERIC.)
                                                                               -- Is current object greater than other?
                                                                               -- (From COMPARABLE.)
          Result_exists: Result /= Void;
                                                                            reauire
          value: Result = 1
                                                                                other_exists: other /= Void
                                                                            ensure
   sign: INTEGER
                                                                               definition: Result = (other < Current)
          -- Sign value (0, --1 or 1)
                                                                         max (other: like Current): like Current
       ensure
                                                                                -- The greater of current object and other
          three_way: Result = three_way_comparison (zero)
                                                                                -- (From COMPARABLE.)
                                                                            require
   zero: like Current
                                                                               other_exists: other /= Void
          -- Neutral element for "+" and "-"
                                                                            ensure
          -- (From NUMERIC.)
                                                                                current_if_not_smaller: (Current >= other) implies
       ensure
                                                                                  (Result = Current)
          Result_exists: Result /= Void;
                                                                               other_if_smaller: (Current < other) implies (Result =
                                                                                  other)
          value: Result = 0
                                                                         min (other: like Current): like Current
feature -- Comparison
                                                                                -- The smaller of current object and other
   infix "<" (other: like Current): BOOLEAN
                                                                                -- (From COMPARABLE.)
          -- Is other greater than current integer?
                                                                            require
          -- (From COMPARABLE.)
                                                                               other_exists: other /= Void
                                                                            ensure
       require
                                                                               current_if_not_greater: (Current <= other) implies
          other_exists: other /= Void
                                                                                  (Result = Current)
       ensure
                                                                               other_if_greater: (Current > other) implies (Result =
          asymmetric: Result implies not (other < Current)
                                                                                  other)
```

§5.7 CLASS CHARACTER 21

feature -- Output

out: STRING

-- Printable representation of character

-- (From GENERAL.)

invariant

 $irreflexive_comparison$: not (Current < Current)

end

5.7 Class CHARACTER

```
indexing
                                                                         infix ">" (other: like Current): BOOLEAN
                                                                                -- Is current object greater than other?
   description: "Characters, with comparison operations and an
                                                                                -- (From COMPARABLE.)
     ASCII code"
                                                                            require
expanded class interface
                                                                                other_exists: other /= Void
   CHARACTER
                                                                            ensure
feature -- Access
                                                                                definition: Result = (other < Current)
   code: INTEGER
                                                                         max (other: like Current): like Current
          -- Associated integer value
                                                                                -- The greater of current object and other
   hash_code: INTEGER
                                                                                -- (From COMPARABLE.)
          -- Hash code value
                                                                            require
          -- (From HASHABLE.)
                                                                                other_exists: other /= Void
       ensure
          good\_hash\_value : Result >= 0
                                                                                current_if_not_smaller: (Current >= other) implies
feature -- Comparison
                                                                                  (Result = Current)
                                                                                other_if_smaller: (Current < other) implies (Result =
   infix "<" (other: like Current): BOOLEAN
                                                                                  other)
          -- Is other greater than current character?
          -- (From COMPARABLE.)
                                                                         min (other: like Current): like Current
                                                                                -- The smaller of current object and other
       require
                                                                                -- (From COMPARABLE.)
          other_exists: other /= Void
                                                                            require
       ensure
          asymmetric: Result implies not (other < Current)
                                                                                other_exists: other /= Void
                                                                            ensure
   infix "<=" (other:like Current): BOOLEAN
                                                                                current_if_not_greater: (Current <= other) implies</pre>
          -- Is current character less than or equal to other?
                                                                                  (Result = Current)
          -- (From COMPARABLE.)
                                                                                other_if_greater: (Current > other) implies (Result =
       require
                                                                                  other)
          other_exists: other /= Void
                                                                         three_way_comparison (other: like Current): INTEGER
       ensure
                                                                                -- If current object equal to other, 0; if smaller,
          definition: Result = (Current < other) or is_equal
                                                                                -- -1; if greater, 1.
            (other);
                                                                                -- (From COMPARABLE.)
   infix ">=" (other: like Current): BOOLEAN
                                                                            require
          -- Is current object greater than or equal to other?
                                                                                other_exists: other /= Void
          -- (From COMPARABLE.)
       require
                                                                            ensure
                                                                                equal\_zero: (Result = 0) = is\_equal (other);
          other_exists: other /= Void
                                                                                smaller: (Result = -1) = Current < other;
       ensure
          definition: Result = (other <= Current)
                                                                                greater\_positive: (Result = 1) = Current > other
```

§5.6 CLASS BOOLEAN

5.6 Class *BOOLEAN*

```
indexing
                                                                        infix "or" (other: BOOLEAN): BOOLEAN
                                                                               -- Boolean disjunction with other
   description: "Truth values, with the boolean operations"
                                                                            require
expanded class interface
                                                                               other_exists: other /= Void
   BOOLEAN
                                                                            ensure
feature -- Access
                                                                               Result_exists: Result /= Void;
   hash_code: INTEGER
          -- Hash code value
                                                                               de_morgan: Result = not (not Current and (not other));
          -- (From HASHABLE.)
                                                                               commutative: Result = (other or Current);
       ensure
                                                                               consistent_with_semi_strict: Result implies (Current or
          good_hash_value: Result >= 0
                                                                                 else other)
feature -- Basic operations
                                                                        infix "or else" (other: BOOLEAN): BOOLEAN
   infix "and" (other: BOOLEAN): BOOLEAN
                                                                               -- Boolean semi-strict disjunction with other
          -- Boolean conjunction with other
                                                                            require
      require
          other_exists: other /= Void
                                                                               other_exists: other /= Void
       ensure
                                                                            ensure
          Result_exists: Result /= Void;
                                                                               Result exists: Result /= Void;
          de_morgan: Result = not (not Current or (not other));
                                                                               de_morgan: Result = not (not Current and then (not
          commutative: Result = (other and Current);
                                                                                 other));
          consistent_with_semi_strict: Result implies (Current
                                                                        infix "xor" (other: BOOLEAN): BOOLEAN
            and then other)
                                                                               -- Boolean exclusive or with other
   infix "and then" (other: BOOLEAN): BOOLEAN
          -- Boolean semi-strict conjunction with other
                                                                            require
       require
                                                                               other_exists: other /= Void
          other_exists: other /= Void
                                                                            ensure
       ensure
                                                                               definition: Result = ((Current or other) and not (Current
          Result_exists: Result /= Void;
                                                                                 and other))
          de_morgan: Result = not (not Current or else (not
            other));
                                                                    feature -- Output
   infix "implies" (other: BOOLEAN): BOOLEAN
                                                                        out: STRING
          -- Boolean implication of other
                                                                               -- Printable representation of boolean
          -- (semi-strict)
                                                                     invariant
       require
          other_exists: other /= Void
                                                                        involutive_negation: is_equal (not (not Current));
                                                                        non_contradiction: not (Current and (not Current));
          definition: Result = (not Current or else other)
                                                                        completeness: Current or (not Current)
   prefix "not": BOOLEAN
          -- Negation.
                                                                     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 17

5.5 Class NUMERIC

```
indexing
                                                                          infix "-" (other: like Current): like Current
                                                                                  -- Result of subtracting other
   description: "Objects to which numerical operations are
                                                                              require
                                                                                 other_exists: other /= Void
   note: "The model is that of a commutative ring."
                                                                              deferred
deferred class interface
                                                                              ensure
   NUMERIC
                                                                                 result_exists: Result /= Void
                                                                          infix "*" (other: like Current): like Current
feature -- Access
                                                                                  -- Product by other
   one: like Current
                                                                              require
          -- Neutral element for "*" and "/"
                                                                                 other_exists: other /= Void
       deferred
                                                                              deferred
       ensure
                                                                              ensure
          Result_exists: Result /= Void
                                                                                 result_exists: Result /= Void
   zero: like Current
                                                                          infix "/" (other: like Current): like Current
          -- Neutral element for "+" and "-"
                                                                                  -- Division by other
       deferred
                                                                              require
       ensure
                                                                                 other_exists: other /= Void;
          Result_exists: Result /= Void
                                                                                 good_divisor: divisible (other)
feature -- Status report
                                                                              deferred
                                                                              ensure
   divisible (other: like Current): BOOLEAN
                                                                                 result_exists: Result /= Void
          -- May current object be divided by other?
                                                                          infix "^" (other: NUMERIC): NUMERIC
       require
                                                                                  -- Current object to the power other
          other_exists: other /= Void
                                                                              require
       deferred
                                                                                 other_exists: other /= Void;
   exponentiable (other: NUMERIC): BOOLEAN
                                                                                 good_exponent: exponentiable (other)
          -- May current object be elevated to the power other?
                                                                              deferred
       require
                                                                              ensure
          other_exists: other /= Void
                                                                                 result_exists: Result /= Void
       deferred
                                                                          prefix "+": like Current
feature -- Basic operations
                                                                                 -- Unary plus
   infix "+" (other: like Current): like Current
                                                                              deferred
          -- Sum with other (commutative).
                                                                              ensure
       require
                                                                                 result_exists: Result /= Void
          other_exists: other /= Void
                                                                          prefix "-": like Current
       deferred
                                                                                 -- Unary minus
       ensure
                                                                              deferred
          result_exists: Result /= Void;
                                                                              ensure
          commutative: equal (Result, other + Current)
                                                                                 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."

deferred class interface
  HASHABLE

feature -- Access
  hash_code: INTEGER
     -- Hash code value
  deferred
  ensure
     good_hash_value: Result >= 0
end
```

§5.3 CLASS COMPARABLE 15

5.3 Class COMPARABLE

```
indexing
                                                                           is_equal (other: like Current): BOOLEAN
                                                                                  -- Is other attached to an object considered equal
   description: "Objects that may be compared according to a
                                                                                  -- to current object?
     total order relation";
                                                                                  -- (Redefined from GENERAL.)
   note: "The basic operation is `< '(less than); others are defined
                                                                              require
     in terms of this operation and `is_equal'."
                                                                                  other_not_void: other /= Void
                                                                              ensure
deferred class interface
                                                                                  symmetric: Result implies other.is_equal (Current);
   COMPARABLE
                                                                                  consistent: standard_is_equal (other) implies Result;
                                                                                  trichotomy: Result = (not (Current < other) and not
feature -- Comparison
                                                                                    (other < Current))
   infix "<" (other: like Current): BOOLEAN
                                                                           max (other: like Current): like Current
          -- Is current object less than other?
                                                                                  -- The greater of current object and other
       require
                                                                              require
                                                                                  other_exists: other /= Void
          other_exists: other /= Void
       deferred
                                                                                  current_if_not_smaller: (Current >= other) implies
       ensure
                                                                                    (Result = Current)
          asymmetric: Result implies not (other < Current)
                                                                                  other_if_smaller: (Current < other) implies (Result =
                                                                                    other)
   infix "<=" (other: like Current): BOOLEAN
                                                                           min (other: like Current): like Current
          -- Is current object less than or equal to other?
                                                                                  -- The smaller of current object and other
       require
                                                                              require
          other_exists: other /= Void
                                                                                  other_exists: other /= Void
                                                                              ensure
                                                                                  current_if_not_greater: (Current <= other) implies
          definition: Result = (Current < other) or is\_equal
                                                                                    (Result = Current)
            (other);
                                                                                  other_if_greater: (Current > other) implies (Result =
   infix ">=" (other: like Current): BOOLEAN
                                                                                    other)
          -- Is current object greater than or equal to other?
                                                                           three_way_comparison (other: like Current): INTEGER)
       require
                                                                                  -- If current object equal to other, 0; if smaller,
                                                                                  -- -1; if greater, 1.
          other_exists: other /= Void
                                                                              require
       ensure
                                                                                  other_exists: other /= Void
           definition: Result = (other <= Current)
                                                                              ensure
   infix ">" (other: like Current): BOOLEAN
                                                                                  equal\_zero: (Result = 0) = is\_equal (other);
           -- Is current object greater than other?
                                                                                  smaller\_negative: (Result = -1) = (Current < other);
                                                                                  greater\_positive: (Result = 1) = (Current > other)
       require
          other_exists: other /= Void
                                                                       invariant
                                                                           irreflexive_comparison: not (Current < Current)</pre>
       ensure
          definition: Result = (other < 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

§5.1 CLASS GENERAL 13

```
frozen standard_is_equal (other: like Current): BOOLEAN
                                                                            frozen default_pointer: POINTER
                                                                                   -- Default value of type POINTER
          -- Is other attached to an object of the same type as
                                                                                   -- (Avoid the need to write p \cdot default for some p
          -- current object, and field-by-field identical to it?
                                                                                   -- of type POINTER.)
       require
                                                                               ensure
          other_not_void: other /= Void
                                                                                   Result = Result \cdot default
       ensure
                                                                            default_rescue
          same_type: Result implies same_type (other);
                                                                                   -- Handle exception if no Rescue clause.
          symmetric: Result implies other. standard_is_equal
                                                                                   -- (Default: do nothing.)
             (Current)
                                                                            frozen do_nothing
feature -- Duplication
                                                                                   -- Execute a null action.
   frozen clone (other: GENERAL): like other
                                                                            frozen Void: NONE
           -- Void if other is void; otherwise new object
                                                                                   -- Void reference
           -- equal to other.
                                                                        feature -- Output
       ensure
                                                                            io: STD_FILES
          equal: equal (Result, other)
                                                                                   -- Handle to standard file setup
   copy (other: like Current)
                                                                            out: STRING
           -- Update current object using fields of object attached
                                                                                   -- New string containing terse printable representation
          -- to other, so as to yield equal objects.
                                                                                   -- of current object
       require
                                                                            print (some: GENERAL)
          other_not_void: other /= Void;
                                                                                   -- Write terse external representation of some on
          type_identity: same_type (other)
                                                                                   -- standard output.
                                                                            frozen tagged_out: STRING
          is_equal: is_equal (other)
                                                                                   -- New string containing printable representation of
   frozen deep_clone (other: GENERAL): like other
                                                                                   -- current object, each field preceded by its attribute
                                                                                   -- name, a colon and a space.
           -- Void if other is void: otherwise, new object structure
           -- recursively duplicated from the one attached to other
                                                                        invariant
                                                                            reflexive_equality: standard_is_equal (Current);
          deep_equal: deep_equal (other, Result)
                                                                            reflexive_conformance: conforms_to (Current);
   frozen standard_clone (other: GENERAL): like other
                                                                            involutive_object_id: id_object (object_id) = Current
           -- Void if other is void; otherwise new object
                                                                         end
          -- 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)
          is_standard_equal: standard_is_equal (other)
feature -- Basic operations
   frozen default: like Current
           -- Default value of current type
```

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. conformance: conforms_to (other) 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, chapter 13)? require $other_not_void$: other/=Voidfrozen 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 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 some): BOOLEAN
          -- Are some and other either both void or attached
          -- to objects considered equal?
          definition: Result = (some = Void and other = Void) or
            else ((some /= Void and other /= Void) and then
            some \cdot is\_equal\ (other));
   is_equal (other: like Current): BOOLEAN
          -- Is other attached to an object considered equal
          -- 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)
   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?

ensure

-- Always uses the default object comparison criterion.

definition: Result = (some = Void and other = Void) or

else ((some /= Void and other /= Void) and then

 $some \cdot standard_is_equal\ (other))$

4.16	• INTEGER_REF is a proper descendant of NUMERIC.
4.17	• POINTER is a proper descendant of POINTER_REF.
4.18	• POINTER_REF is a proper descendant of HASHABLE.
4.19	• <i>REAL</i> is a proper descendant of <i>REAL_REF</i> .
4.20	• REAL_REF is a proper descendant of COMPARABLE.
4.21	• REAL_REF is a proper descendant of HASHABLE.
4.22	• STRING is a proper descendant of COMPARABLE.
4.23	• STRING is a proper descendant of HASHABLE.
4.24	• STRING is a proper descendant of HASHABLE.

[4.1 follows from *Eiffel: The Language*; the language description is considered to be amended in such a way that *PLATFORM* is a class without privileges, to be inherited explicitly by classes which need access to its features.]

4.15

3.17	• MEMORY [flatshort form in section 5.17].
3.18	• EXCEPTIONS [flatshort form in section 5.18].
3.19	• ARGUMENTS [flatshort form in section 5.19].
3.20	• <i>PLATFORM</i> [flatshort form in section 5.20].
3.21	• BOOLEAN_REF [flatshort form in section 5.21].
3.22	• CHARACTER_REF [flatshort form in section 5.22].
3.23	• DOUBLE_REF [flatshort form in section 5.23].
3.24	• INTEGER_REF [flatshort form in section 5.24].
3.25	• POINTER_REF [flatshort form in section 5.25].
3.26	• REAL_REF [flatshort form in section 5.26].
	[The classes appear in this section and section and section 5 in the following order: universal classes: deferred classes for basic classes: basic classes: arrays and strings: operating system

4 REQUIRED ANCESTRY LINKS

The following constitute the required ancestry links [ordered alphabetically, after the first

interface; auxiliary reference classes for the definition of basic classes.]

	rule, by the name of the applicable descendant class]:
4.1	 Every Required Class except GENERAL is a descendant of ANY
4.2	• ANY is an heir of GENERAL.
4.3	• BOOLEAN is a proper descendant of BOOLEAN_REF.
4.4	• BOOLEAN_REF is a proper descendant of HASHABLE.
4.5	• CHARACTER is a proper descendant of CHARACTER_REF.
4.6	• CHARACTER_REF is a proper descendant of COMPARABLE.
4.7	• CHARACTER_REF is a proper descendant of HASHABLE.
4.8	• DOUBLE is a proper descendant of DOUBLE_REF.
4.9	• DOUBLE_REF is a proper descendant of COMPARABLE.
4.10	• DOUBLE_REF is a proper descendant of HASHABLE.
4.11	• DOUBLE_REF is a proper descendant of NUMERIC.
4.12	• FILE is a proper descendant of MEMORY.
4.13	• INTEGER is a proper descendant of INTEGER_REF.
4.14	• INTEGER_REF is a proper descendant of COMPARABLE.

• *INTEGER_REF* is a proper descendant of *HASHABLE*.

2.4.1.8 • For a feature that appears in both ic and sc, an argument type in ic that is different from the corresponding type in sc but conforms to it. 2.4.1.9 • For a feature that appears in both ic and sc, a line that appears in the Header_comment of *ic* but not in that of *sc*. 2.4.1.10 • An Index_clause that appears in *ic* but not in *sc*. 2.4.1.11 • A difference regarding the order in which a feature appears in ic and sc, the Feature_ clause to which it belongs, the Header_comment of such a Feature_clause, or the presence in ic of a Feature clause that has no counterpart in sc. [As a consequence of section 2.4.1.11, the division of classes into one Feature_clause or more,

and the labels of these clauses, appear in this document for the sole purpose of readability and ease of opdreference, but are not part of this Standard.]

[The goal pursued by the preceding definition is to make sure that an Eiffel system that follows this Standard will be correctly processed by any compatible implementation, without limiting the implementors' freedom to provide more ambitious facilities.]

3 REQUIRED CLASSES

3.15 3.16 asses [ordered from the general to the

	The Required Classes are the following twenty class specific, as in section 5]:
3.1	• GENERAL [flatshort form in section 5.1].
3.2	• ANY [flatshort form in section 5.2].
3.3	• COMPARABLE [flatshort form in section 5.3].
3.4	• HASHABLE [flatshort form in section 5.4].
3.5	• <i>NUMERIC</i> [flatshort form in section 5.5].
3.6	• BOOLEAN [flatshort form in section 5.6].
3.7	• CHARACTER [flatshort form in section 5.7].
3.8	• <i>INTEGER</i> [flatshort form in section 5.8].
3.9	• <i>REAL</i> [flatshort form in section 5.9].
3.10	• DOUBLE [flatshort form in section 5.10].
3.11	• <i>POINTER</i> [flatshort form in section 5.10].
3.12	• ARRAY [flatshort form in section 5.12].
3.13	• STRING [flatshort form in section 5.13).
3.14	• STD_FILES [flatshort form in section 5.14].

• FILE [flatshort form in section 5.15].

• STORABLE [flatshort form in section 5.16].

2.3.1.5	• The header comment of any inherited feature coming from a Required Class A and having the same name in C as in A shall end with a line of the form:
	(From <i>A</i> .)
2.3.1.6	• The header comment of any inherited feature coming from a Required Class <i>A</i> and having a name in <i>C</i> different from its name <i>x</i> in <i>A</i> shall end with a line of the form:
	(From x in A .)
	[The comments defined in the last two rules are applicable regardless of whether C redeclares the feature.]
2.3.1.7	• If deferred, C shall appear as deferred class.
2.3.1.8	• Any deferred feature of C shall be marked as deferred.
2.3.1.9	• In case of precondition redeclaration, the successive preconditions shall appear as a single Precondition clause, separated by semicolons.
2.3.1.10	• In case of postcondition redeclaration, the successive preconditions shall appear as a single Postcondition clause, separated by <i>and then</i> .

2.4 Flatshort Compatibility

2.4.1 Definition

2.4.1.1

2.4.1.4

2.4.1.5

2.4.1.6

A class appearing in an Eiffel implementation is said to be Flatshort-Compatible with a class of the same name listed in this Standard if and only if any difference that may exist between its flatshort form *ic* and the flatshort form *sc* of the corresponding class as it appears in section 5, where both flatshort forms follow the conventions of section 2.3, belongs to one of the following eleven categories:

- A feature that appears in *ic* but not in *sc*, whose Header_comment includes, as its last line, the mention:
 - -- (Feature not in Kernel Library Standard.)
- An invariant clause that appears in *ic* but not in *sc*.
- For a feature that appears in both *ic* and *sc*, a postcondition clause that appears in *ic* but not in *sc*.
 - For a feature that appears in both *ic* and *sc*, a precondition in *sc* that implies the precondition in *ic*, where the implication is readily provable using rules of mathematical logic.
 - For a feature that appears in both *ic* and *sc*, a postcondition or invariant clause in *ic* that implies the corresponding clause in *sc*, where the implication is readily provable using rules of mathematical logic.
 - A difference between the Tag_mark of an Assertion_clause in *ic* and its counterpart in *sc*.
- For a feature that appears in both *ic* and *sc*, an argument type in *sc* that is different from the corresponding type in *ic* but conforms to it.

2.2 Kernel compatibility

2.2.1 Definition

An Eiffel implementation will be said to be kernel-compatible if and only if it includes a set of classes satisfying the following five conditions:

- For each of the Required Classes, the implementation includes a class with the same name.
- 2.2.1.2 All the Required Ancestry Links are present between these classes.
- The flatshort form of each one of these classes is Flatshort-Compatible with the corresponding Required Flatshort Form.
- All the dependents of the Required Classes in the implementation are also included in the implementation.
- None of the features appearing in the Required Flatshort Forms appears in a Rename clause of any of the implementation's Required Classes.

[These conditions allow a kernel-compatible implementation to include inheritance links other than the ones described in this Standard; condition 2.2.1.4 indicates that for any such link the additional proper ancestors must also be provided by the implementors, since the dependents of a class include its parents.]

[Condition 2.2.1.4 guarantees that if a feature name appears in this Standard both in the Flatshort form of a Required Class and in the flatshort form of one of its proper ancestors, it corresponds to the same feature or to a redefinition of it.]

2.3 Flatshort Conventions

2.3.1 Definition

2.3.1.2

2.3.1.3

2.3.1.4

In the process of assessing for Flatshort Compatibility a class *C* from a candidate implementation, the following ten conventions, which have been applied to the Required Flatshort Forms as they appear in this Standard, shall be applied:

- No feature shall be included unless it is generally available (as defined in *Eiffel: The Language*, page 100) or is a general creation procedure (as defined in *Eiffel: The Language*, page 285).
 - The Creation clause of the flatshort specification shall include the full specification of all general creation procedures of *C*.
 - Any feature of *C* not inherited from *GENERAL* shall be included in one of the Feature clauses.

[As a consequence of the last two rules the specification of a creation procedure that is also generally exported will appear twice: in the Creation clause and in a Feature clause. Also note that the "features of a class" include inherited as well as immediate features, so that all features inherited from an ancestor other than *GENERAL* must appear in the flatshort form.]

• A feature f from GENERAL shall be included if and only if C redeclares f.

1.2 Scope of this Standard

This Standard defines a number of library-related conditions that an Eiffel implementation must satisfy. These conditions affect a set of classes known as the kernel library. An implementation that satisfies the conditions described in this Standard will be said to be **kernel-compatible**, a phrase that is abbreviated in this Standard as just "compatible".

[In other contexts it may be preferable to use the full phrase, since the compatibility of an Eiffel implementation also involves other aspects, such as language compatibility.]

[The terms "compatibility" and "compatible" may be felt to be less clear than "conformance" and "conformant". The former are used here, however, since talking about conformance might cause confusions with the Eiffel notion of a type conforming to another.]

1.3 Other documents

The phrase *Eiffel: The Language* as used in this Standard refers to the second printing of the book *Eiffel: The Language*, Prentice Hall, 1992, ISBN 0-13-245-925-7.

For the purposes of this Standard, the definition of the Eiffel language is the definition given by *Eiffel: The Language*.

In case of contradictions between the library specifications given by *Eiffel: The Language* and those given in this Standard, the latter shall take precedence.

2 COMPATIBILITY CONDITIONS

2.1 Definitions

2.1.1 Required Classes

In this Standard, the phrase "Required Classes" denotes a set of classes whose names are those listed in section 3.

2.1.2 Required Flatshort Form

In this Standard, the phrase "Required Flatshort Forms" denotes the flatshort forms given for the Required Classes in section 3.

2.1.3 Flatshort Compatibility

In this Standard, a class is said to be Flatshort-Compatible with one of the short forms given in this Standard if it satisfies the conditions given in section 2 of this Standard.

2.1.4 Required Ancestry Links

In this Standard, the expression "Required Ancestry Links" denotes the inheritance links specified in section 4 of this Standard.

[The term "Ancestry" is used rather than "Inheritance" because the required links may be implemented by indirect rather than direct inheritance, except for *ANY* which must be a direct heir of *GENERAL* as per rule 4.2, given on page 10.]

§1.1 INTRODUCTION 5

0 INTRODUCTION

[This introduction is not part of the Standard.]

0.1

To favor the interoperability between implementations of Eiffel, it is necessary, along with a precise definition of the language, to have a well-defined set of libraries covering needs that are likely to arise in most applications. This library is known in Eiffel as the Kernel Library.

0.2

The present document defines a standard for the Kernel Library. If an Eiffel implementation satisfies this Standard — under the precise definition of *Kernel Compatibility* given in section 2.2 — it will be able to handle properly any Eiffel system whose use of the Kernel Library only assumes the library properties defined in this Standard.

0.3

The Eiffel Library standardization process, as described in Appendix A of the present document, is based on a dynamic view which, in the spirit of Eiffel's own "feature obsolescence" mechanism, recognizes the need to support evolution while preserving the technology investment of Eiffel users. One of the consequences of this dynamic view is to define *vintages* corresponding to successive improvements of the Standard. The present document describes **Vintage 95**, valid for the calendar year 1995.

1 CONTENTS OF THIS STANDARD

1.1 Definition: this Standard

The Eiffel Kernel Library Standard, denoted in the present document by the phrase "this Standard", is made up of the contents of sections 1 to 5 of the present document, with the exception of elements appearing between square brackets [...] which are comments.

[As a result of the preceding definition the following elements are not part of this Standard: section 0, the table of contents, Appendix A in section 6 (the Kernel Library Standardization process), Appendix B in section 7 (list of differences), the Index in section 8, and elements playing a pure typesetting role such as page headers.]

4	CONTENTS
---	----------

	5.22 Class CHARACTER_REF	48	
	5.23 Class DOUBLE_REF	49	
	5.24 Class INTEGER_REF	50	
	5.25 Class POINTER_REF	51	
	5.26 Class REAL_REF	52	
6	APPENDIX A: THE KERNEL STANDARDIZATION PROCESS	53	
	6.1 Why plan a process?	53	
	6.2 Cycle time	53	
	6.3 Vintages	53	
	6.4 Yearly schedule	53	
	6.5 Intermediate corrections	54	
	6.6 Eiffel Kernel Supplier requirements	54	
7	APPENDIX B: DIFFERENCES	54	
8	INDEX	57	

CONTENTS

0	INTRODUCTION	5
1	CONTENTS OF THIS STANDARD	5
	1.1 Definition: this Standard	5
	1.2 Scope of this Standard	6
	1.3 Other documents	6
2	COMPATIBILITY CONDITIONS	6
	2.1 Definitions	6
	2.2 Kernel compatibility	7
	2.3 Flatshort Conventions	7
	2.4 Flatshort Compatibility	8
3	REQUIRED CLASSES	9
4	REQUIRED ANCESTRY LINKS	10
5	SHORT FORMS OF REQUIRED CLASSES	12
	5.1 Class GENERAL	12
	5.2 Class ANY	14
	5.3 Class COMPARABLE	15
	5.4 Class HASHABLE	16
	5.5 Class NUMERIC	17
	5.6 Class BOOLEAN	19
	5.7 Class CHARACTER	20
	5.8 Class INTEGER	22
	5.9 Class REAL	25
	5.10 Class DOUBLE	28
	5.11 Class POINTER	31
	5.12 Class ARRAY	32
	5.13 Class STRING	34
	5.14 Class STD_FILES	38
	5.15 Class FILE	39
	5.16 Class STORABLE	42
	5.17 Class MEMORY	43
	5.18 Class EXCEPTIONS	44
	5.19 Class ARGUMENTS	45
	5.20 Class PLATFORM	46
	5.21 Class BOOLEAN REF	47

Report identification

TR-EI-48/KL: *The Eiffel Library Standard*. (Earlier title: *The Proposed Eiffel Library Kernel Standard*). Prepared for the Nonprofit International Consortium for Eiffel by:

Interactive Software Engineering Inc. 270 Storke Road, Suite 7 Goleta, CA 93117 USA Phone 805-685-1006, Fax 805-685-6869 Electronic Mail <info@eiffel.com>

Publication history

First published: 14 July 1994. Version 5: 25 September 1994. Version 6: 10 October 1994.

Version 7: 12 January 1995, integrating the result of the votes of the NICE library committee (December 1994) on classes ANY, PLATFORM and GENERAL..

The present document is version 8, 4 June 1995, designed as vintage 95 library standard.

Authors

The document was written by Bertrand Meyer and for versions up to 7 benefited from numerous discussions with Michael Schweitzer.

The substance (the kernel specification) is the result of a collective effort with contributions from:

Eric Bezault (ISE, Santa Barbara)

Roger Browne (Everything Eiffel, Great Britain)*

Didier Dupont (SOL, Paris)

Fred Hart (Tower Technology, Atlanta)*

Aaron Hillegas (Tower Technology, Atlanta)

Rock Howard (Tower Technology, Austin)*

Paul Johnson (GEC-Marconi, Great Britain)**

Xavier Le Vourch (ISE, Santa Barbara)

David Morgan (ISE, Santa Barbara)

Bertrand Meyer (ISE, Santa Barbara)

Christine Mingins (Monash University, Melbourne), chairman of the NICE library committee

Michael Schweitzer (Swissoft, Göttingen)*

Philippe Stephan (SCENEC, Paris)

Robert Switzer (Swissoft, Göttingen, and University of Göttingen)*

Steve Tynor (Tower Technology, Atlanta)*

Dino Valente (ISE, Santa Barbara)

The influence of the Gustave kernel proposal (authors marked * above) and the GEC-Marconi kernel proposal (author marked ** above) is gratefully acknowledged.

Copyright notice

Copyright © Nonprofit International Consortium for Eiffel (NICE), 1995. Use and duplication of this document for the work of NICE, in conformance with the bylaws and regulations of NICE, or by members of NICE for any purpose, is permitted provided the present copyright notice is included in every copy. Any other use or duplication requires the permission of the copyright holder:

Nonprofit International Consortium for Eiffel

P.O. Box 6884

Santa Barbara, CA 93160 USA

Electronic mail <nice@atlanta.twr.com>.

THE EIFFEL LIBRARY STANDARD

VINTAGE 95