NEW FEATURE FOR MODULE PROTECTION IN SIMULA

By Jacob Palme, Swedish National Defense Research Institute
FOA 1
S-104 50 Stockholm 80
Sweden

On its meeting in Brighton in September 1975, the SIMULA Development Group approved of an extension to SIMULA called the HIDDEN PROTECTED specification.

The HIDDEN PROTECTED specification will be available in Release 3 of the DECsystem-10 SIMULA system, which will be released in February or March 1976.

The HIDDEN PROTECTED specification is a feature to increase the reliability and security of large software packages by dividing the package into smaller submodules and controlling the interface between the submodules.

Many programming errors occur because a programmer, when writing code in one submodule, is not aware of the effect of that code on other submodules. If the interface between the modules is controlled, the compiler and program can automatically guard against many such programming errors.

This is especially important when several programmers are working on the same project, or when one programmer is writing a program using one or several packages developed previously by someone else.

The natural submodule in SIMULA is a CLASS. This CLASS is sometimes separately compiled However, programs using a separately compiled CLASS do not ordinarily use directly all the attributes of the separately compiled CLASS. Some attributes are internal to the CLASS and only used by code inside the CLASS. Other attributes are "ports of entry" to the CLASS. Often, these "ports of entry" are PROCEDURE attributes to the CLASS, since PROCEDUREs can include code to check the correctness of the input parameters before performing the requested actions.

The simplest use of the HIDDEN PROTECTED specification is to declare as HIDDEN PROTECTED those attributes which are not to be accessible outside the body of the CLASS. Suppose for example that we have a CLASS "data base" where the user of the CLASS need only access the features of the CLASS through the attribute PROCEDUREs "put" and "get". All other attributes of the CLASS are then declared HIDDEN PROTECTED as in the following simplified excerpt:

```
CLASS data_base;
HIDDEN PROTECTED diskdata, has_been_read, filesize,
filepos, internal;
BEGIN
    REF (directfile) diskdata;
    BOOLEAN has_been_read;
    INTEGER filesize, filepos;
    PROCEDURE internal; COMMENT this PROCEDURE is only
    used internally inside the class data_base;;
    PROCEDURE PUT; COMMENT port accessible from
    outside the CLASS ...;;
    PROCEDURE GET; COMMENT port accessible from
    outside the CLASS ...;;
    COMMENT etc. ;
END of data_base;
```

If most of the attributes of a CLASS are to be HIDDEN PROTECTED, and only a few attributes are to be non-protected ports, then it is easier to list the non-protected attributes. This can be done using the NOT HIDDEN PROTECTED specification. The CLASS in the example above could then instead begin:

```
CLASS database;
NOT HIDDEN PROTECTED put, get;
BEGIN ...
```

The NOT HIDDEN PROTECTED specification indicates that all attributes which can be specified HIDDEN PROTECTED at that place in the program, and which are not listed in the specification, are specified to be HIDDEN PROTECTED.
<virtual part> <class body>

SEMANTICS OF THE HIDDEN PROTECTED SPECIFICATION

Attributes declared PROTECTED are only visible inside the body of the class where they have been specified PROTECTED, and inside the body of subclasses to this class and inside the body of blocks prefixed with this class or a subclass to it.

Attributes declared HIDDEN are invisible in subclasses to the class where the HIDDEN specification occurs and in blocks prefixed with this class or subclasses to it.

The combined specification HIDDEN PROTECTED will thus make the variables invisible everywhere except in the body of the class with the HIDDEN PROTECTED specification.

EXAMPLES OF USE OF THE HIDDEN PROTECTED SPECIFICATION

Example i: WRITE-PROTECTED ATTRIBUTES

The attributes "length", "width" and "height" of the following CLASS can be given values when a new CLASS object is generated, but can never be changed in an existing CLASS object.

The attribute "density" can only be changed through the procedure "setdensity" which checks that the new value is within a legal range.

The attribute values can all be used outside the CLASS body, but not redefined, since the externally accessible ports are all PROCEDUREs.

```
    CLASS box (internal_length, internal_width,
        internal_height, internal density);
    REAL internal_length, internal_width,
        internal_height, internal density;
    NOT HIDDEN PROTECTED length, width, height,
        density, set_density;
    BEGIN
        REAL PROCEDURE length; length:= internal_length;
        REAL PROCEDURE width; width:= internal_width;
        REAL PROCEDURE height; height:= internal_height;
        REAL PROCEDURE density; density:= internal_density;
        PROCEDURE set_density(new_density);
        IF new_density < 0 OR new_density > 10 THEN abort
        ELSE internal_density:= new_density;

        set_density(internal_density); COMMENT to check
        initial value;
        ...
    END;
```

Example ii: SIMSET AS IN SIMULA COMMON BASE

```
CLASS simset;
BEGIN
    CLASS linkage;
    PROTECTED i_suc, i_pred;
    BEGIN
        REF (linkage) i_suc, i_pred;
    END;
    linkage CLASS link;
    HIDDEN i_suc, i_pred;
    BEGIN
        PROCEDURE out;
        BEGIN
            IF i_suc =/= NONE THEN
            BEGIN
            END;
        END;
    END;
END;
```

As is seen above, the internal, HIDDEN attributes i_suc and i_pred are invisible outside LINKAGE, except within the subclasses LINK (and HEAD, analoguous to LINK) since they are declared PROTECTED in LINKAGE, but declared HIDDEN in LINK (and HEAD).