

Global Objects

Manifest Constants

- More commonly known as **literals**

Objects with their name being their value

- > **Numbers** 0, -1, 5, 5.123, -4.3⁻⁶, ...
- > **Strings** "abcd", "I am a string", ...
- > **Characters** 'a', '0', ...

- **Symbolic Constant Principle**

Do not use a manifest constant, other than zero or identity elements of basic operations, in any construct other than a symbolic constant declaration

- > **File_not_found : STRING** is "Cannot find file"
- > **Char_newline : CHARACTER** is '%N'

Global Constants

- Group into appropriate classes

```
class EDITOR_CONSTANTS feature  
  Insert : CHARACTER is 'i'  
  Delete : CHARACTER is 'd'  
end
```

- Use – multiple inheritance as required

```
class EDITOR inherit EDITOR_CONSTANTS  
feature  
  ... reference by name ... Insert , Delete  
end
```

- But: **EDITOR** is not an **EDITOR_CONSTANTS**
 » **Unlikely to substitute, still a bit jarring**

Global Constants – 2

- Group into appropriate classes

```
class EDITOR_CONSTANTS feature  
  Insert : CHARACTER is 'i'  
  Delete : CHARACTER is 'd'  
end
```

- Have an attribute for the shared constants

```
class EDITOR  
feature  
  ed_const : EDITOR_CONSTANTS  
  ...  
  create ed_const  
  ed_const.Insert      -- indirect reference  
  ...  
end
```

User Type Constants

- Need a mechanism to create and access constants for any type a user may create.
- Once routine

```
constant : UserType is  
  once  
    create Result.make (...)  
  end
```

- Example

```
i : Complex is  
  once  
    create Result.make_cartesian ( 0, 1)  
  end
```

Replaces do

Once Routine

- The body is executed only once
 - » **Result is saved and returned on every call**
 - » **For expanded variables, have true constants**
 - » **For references, have shared objects**
 - > **The referenced object can be modified**
- Using the make facility guarantees the constant satisfies the class invariants
- To prevent changes (e.g. in the value of complex i)
 - » **Add to class invariant**
i.x = 0 and i.y = 1

Shared Objects

- Example of a message window in an interactive system
 - > **Many classes will want to use the same message window – constant**
 - > **The displayed message changes, thus the window as an object changes**

**Message_window : Window is
once
 create Result.make (... param for window ...)
end**

... Example use ...

Message_window.put_text("The message")

Once Procedures

- Can use the once mechanism to execute a procedure once – no value is returned
 - > **Display help windows**
 - » **An initialization routine may be called from different classes depending upon what a user does**
 - » **Do not execute if the user does not execute a method from a specific set**
 - » **But only execute once even if user executes multiple methods from the set**
- Better than using a flag to control once only use as compiler enforces it

Once Function Rule

- » **The result type of a once function may not be anchored and may not involve formal generic parameters**

Unique Values

- Unique values are often used to distinguish cases

> **A frequent use of symbolic constants**

IO_completion_code : INTEGER

successful_open : INTEGER is 1

successful_close : INTEGER is 2

...

- Let compiler select values, rather than programmer

**successful_open, successful_close
: INTEGER is unique**

- Values are unique and ascending if defined in one statement

if code > successful_open then ...