

Preliminary Specification for Compilation Units in Objective Modula-2

Document version: 1.06 -- Date: 2005-08-15 -- Author: BK

1) Compilation Units

There are four types of compilation units in Objective Modula-2: program modules, library modules, class modules and protocol modules.

```
compilation-unit = program-module | library-module |  
                  class-module | protocol-module .
```

2) Program Modules

Program modules correspond directly to program modules as they are defined in Modula-2.

```
program-module = MODULE ident [ priority ] ";"  
               { import } block ident "." .  
  
import = non-qualified-import | qualified-import |  
        aliased-import | qualified-aliased-import .  
  
non-qualified-import = IMPORT ident-list ";" .  
  
qualified-import = FROM ident non-qualified-import .  
  
aliased-import = IMPORT ident AS ident { "," ident AS ident } ";" .  
  
qualified-aliased-import = FROM ident aliased-import .
```

Example:

```
MODULE HelloWorld;  
  
FROM StdIO IMPORT WriteLn;  
  
BEGIN  
    WriteLn("Hello World");  
END HelloWorld.
```

3) Library Modules

Library modules correspond to library modules as they are defined in Modula-2. Definition modules define library interfaces and implementation modules contain the corresponding implementation. In addition to conventional Modula-2 libraries, in Objective Modula-2, library modules may also be used for Cocoa/GNUstep class clusters and frameworks.

a) Library Interface Modules

```
definition-module = DEFINITION MODULE ident ";" [ "<*FRAMEWORK*>" ]  
                  { import } { definition } END ident "." .
```

Example 1: M2 Library

```
DEFINITION MODULE FooLib;
```

```

VAR foo: Foo;

PROCEDURE getFoo : Foo;

PROCEDURE setFoo (f : Foo);

END FooLib.

```

Example 2: Cocoa/GNUstep Framework

```

DEFINITION MODULE FooAndBarClassLib; <*FRAMEWORK*>
(* directive instructs compiler to build this library as framework *)

EXPORT FooClass, BarClass;

INTERFACE MODULE FooClass : NSObject; <*EXTERNAL=ObjM2*>
(* the actual module is an external ObjM2 module *)

INTERFACE MODULE BarClass : NSObject; <*EXTERNAL=ObjC*>
(* the actual module is an external ObjC module *)

END FooAndBarClassLib.

```

b) Library Implementation Modules

implementation-module = IMPLEMENTATION program-module .

Example: M2 Library

```

IMPLEMENTATION MODULE FooLib;

PROCEDURE getFoo : Foo;
BEGIN
    RETURN foo;
END;

PROCEDURE setFoo (f : Foo);
BEGIN
    foo := f;
END;

END FooLib.

```

4) Class Modules

Class modules define classes and categories which are extensions of classes, following the Objective-C object model.

a) Class Interface Modules

```

class-interface-module = INTERFACE MODULE ident
    [ ( ":" ident ) | ( EXTENDS ident ) ]
    [ ADOPTS ident-list ] ";"
    { import } { class-definition } END ident "." .

```

```

ident-list = ident { "," ident } .

class-defintion = constant-declaration |
                  type-declaration |
                  class-variable-declaration |
                  instance-variable-declaration |
                  method-declaration .

method-declaration = class-method-declaration |
                     instance-method-declaration .

class-variable-declaration = CLASS VAR variable-declaration-list .

instance-variable-declaration = INSTANCE VAR variable-declaration-list .

variable-declaration-list = [ access-mode ] variable-declaration
                           { variable-declaration } .

access-mode = PUBLIC TO ( ALL | SUBCLASSES | NEIGHBORS ) | PRIVATE .

variable-declararion = identifier ":" type ";"

class-method-declaration = CLASS METHOD method-header ":" ident ";" .

instance-method-declaration = INSTANCE METHOD method-header
                             [ : ident ] ";" .

```

NB: Objective-C does not define any facility to implement class variables. The ObjM2 project intends to provide such a facility within the limits of compatibility to the ObjC runtime.

Example 1: Declaration of a new class

```

INTERFACE MODULE FooBarClass : NSObject;

INSTANCE VAR
    foo: Foo; // private by default
    bar: Bar; // private by default

// Constructor
CLASS METHOD initWithFoo: (foo : Foo)
    andBar: (bar : Bar) : FooBarClass;

// Accessors
INSTANCE METHOD foo : Foo;

INSTANCE METHOD bar : Bar;

// Mutators
INSTANCE METHOD setFoo: (foo : Foo);

INSTANCE METHOD setBar: (bar : Bar);

END FooBarClass.

```

Example 2: Declaration of instance variables with different access modes

```

INTERFACE MODULE AccessModeExpl : NSObject;

INSTANCE VAR

```

```

doo: Doo; // default is private
PUBLIC TO ALL
foo: Foo; // visible anywhere without restrictions
PUBLIC TO SUBCLASSES
bar: Bar; // visible to subclasses and their subclasses
PUBLIC TO NEIGHBORS
far: Bar; // visible to classes declared within the same module
PRIVATE
boo: Bar; // visible only within this class and categories of it

```

Example 3: Declaration of a category

```

INTERFACE MODULE AdditionsToNSString EXTENDS NSString;

// Instance Methods
INSTANCE METHOD hasWhitespace : BOOLEAN;

INSTANCE METHOD stringByCollapsingWhitespace : NSString;

END AdditionsToNSString.

```

Example 4: Adopting a protocol

```

INTERFACE MODULE MyClass ADOPTS NSCopying;
// Variable declarations
// Method declarations
END MyClass.

```

b) Class Implementation Modules

```

class-implementation-module = IMPLEMENTATION MODULE ident ";"
                             { import } { method-implementation }
                             END ident "." .

method-implementation = class-method-implementation |
                        instance-method-implementation .

class-method-implementation = class-method-declaration block ";" .

instance-method-implementation = instance-method-declaration block ";" .

```

Example:

```

IMPLEMENTATION MODULE FooBarClass;

// Constructor
CLASS METHOD initWithFoo: (foo : Foo)
                    andBar: (bar : Bar) : OBJECT;
VAR
    thisInstance : POINTER TO FooBarClass;
BEGIN
    thisInstance := [[FooBarClass alloc] init];
    thisInstance^.foo := foo;
    thisInstance^.bar := bar;
    RETURN thisInstance;
END;

```

```

// Accessor foo
INSTANCE METHOD foo : Foo;
BEGIN
    RETURN SELF^.foo;
END;

(* other methods *)

END FooBarClass.

```

5) Protocol Modules

Protocol modules define interfaces for methods unattached to any class but which any given class may implement. This facility, too, follows the Objective-C object model.

```

protocol-interface-module = PROTOCOL INTERFACE MODULE ident
                           ":" ident [ ADOPTS ident-list ] ";"
                           { method-declaration } END ident "." .

```

Example:

```

PROTOCOL INTERFACE MODULE FooProtocol : NSObject;

INSTANCE METHOD fooWithBar: (bar : Bar) : Foo;

END FooProtocol.

```

END OF DOCUMENT