Forward Declarations

Both interfaces and classes can be forward declared. Forward declarations permit the creation of mutually dependent objects, for example:

Slice

```
module Family {
    interface Child;            // Forward declaration

    sequence<Child*> Children;  // OK

    interface Parent {
        Children getChildren(); // OK
    };

    interface Child {           // Definition
        Parent* getMother();
        Parent* getFather();
    };
};
```

Without the forward declaration of `Child`, the definition obviously could not compile because `Child` and `Parent` are mutually dependent interfaces. You can use forward-declared interfaces and classes to define types (such as the `Children` sequence in the previous example). Forward-declared interfaces and classes are also legal as the type of a structure, exception, or class member, as the value type of a dictionary, and as the parameter and return type of an operation. However, you cannot inherit from a forward-declared interface or class until after its definition has been seen by the compiler:

Slice

```
interface Base;                         // Forward declaration

interface Derived1 extends Base {};     // Error!

interface Base {};                      // Definition

interface Derived2 extends Base {};     // OK, definition was seen
```

Not inheriting from a forward-declared base interface or class until its definition is seen is necessary because, otherwise, the compiler could not enforce that derived interfaces must not redefine operations that appear in base interfaces.

A multi-pass compiler could be used, but the added complexity is not worth it.

See Also

- Interfaces, Operations, and Exceptions
- Classes