### Module Types and Classes

The classes of modern object-oriented languages are an extension of module types. Access to a module instance typically looks like access to an object.

### Object Orientation

The difference between module types and classes is a powerful pair of features found together in classes and not module types— namely **inheritance** and **dynamic method dispatch**. Inheritance allows new classes to be defined as extensions or refinements of existing classes. Dynamic method dispatch allows a refined class to **override** the definition of an operation in its parent class, and for the choice among definitions to be made at run time, on the basis of whether a particular object belongs to the child class or merely to the parent.

Inheritance facilitates a programming style in which all or most operations are though of as belonging to objects, and in which new objects can inherit many of their operations from existing objects, without the need to rewrite code.

Module types and classes require only simple changes to the scope rules defined for modules. Every instance $A$ of a module type or class has a separate copy of the module or class's variable. These variables are then visible when executing one of $A$'s operations. They may also be indirectly visible to the operations of some other instance $B$ if $A$ is passed as a parameter to one of those operations.

### Modules Containing Classes

Classes are for **data abstractions**, Modules support **separate compilation** (of functionality) and serve to mimimize name conflicts.