





Objectives 5

OO Concepts




Encapsulation, IS-A, HAS-A (Objective 5.1)

- Encapsulation helps hide implementation behind an interface (or API).
- Encapsulated code has two features:
 - Instance variables are kept protected (usually with the private modifier).
 - Getter and setter methods provide access to instance variables.
- IS-A refers to inheritance.



Encapsulation, IS-A, HAS-A (Objective 5.1) [contd.]


- IS-A is expressed with the keyword extends.
- IS-A, "inherits from," and "is a subtype of" are all equivalent expressions.
- HAS-A means an instance of one class "has a" reference to an instance of another class.



Coupling and Cohesion

(Objective 5.1)

- Coupling refers to the degree to which one class knows about or uses members of another class.
- Loose coupling is the desirable state of having classes that are well encapsulated, minimize references to each other, and limit the breadth of API usage.
- Tight coupling is the undesirable state of having classes that break the rules of loose coupling.



Coupling and Cohesion (Objective 5.1) [contd.]

- Cohesion refers to the degree in which a class has a single, well-defined role or responsibility.
- High cohesion is the desirable state of a class whose members support a single, well-focused role or responsibility.
- Low cohesion is the undesirable state of a class whose members support multiple, unfocused roles or responsibilities.



Polymorphism (Objective 5.2)


- Polymorphism means 'many forms'.
- A reference variable is always of a single, unchangeable type, but it can refer to a subtype object.
- A single object can be referred to by reference variables of many different types—as long as they are the same type or a supertype of the object.



Polymorphism

(Objective 5.2) [contd.]

- The reference variable's type (not the object's type), determines which methods can be called!
- Polymorphic method invocations apply only to overridden *instance* methods.



Reference Variable Casting (Objective 5.2)

- There are two types of reference variable casting: downcasting and upcasting.
- Downcasting: If you have a reference variable that refers to a subtype object, you can assign it to a reference variable of the subtype. You must make an explicit cast to do this, and the result is that you can access the subtype's members with this new reference variable.
- Upcasting: You can assign a reference variable to a supertype reference variable explicitly or implicitly. This is an inherently safe operation because the assignment restricts the access capabilities of the new variable.



Overriding and Overloading (Objectives 1.5 and 5.4)

- Methods can be overridden or overloaded; constructors can be overloaded but not overridden.
- Abstract methods must be overridden by the first concrete (nonabstract) subclass.
- final methods cannot be overridden.



Overriding and Overloading (Objectives 1.5 and 5.4) [contd.]

- With respect to the method it overrides, the overriding method
 - Must have the same argument list
 - Must have the same return type, except that as of Java 5, the return type can be a subclass—this is known as a covariant return.
 - Must not have a more restrictive access modifier
 - May have a less restrictive access modifier
 - Must not throw new or broader checked exceptions
 - May throw fewer or narrower checked exceptions, or any unchecked exception.



Overriding and Overloading (Objectives 1.5 and 5.4) [contd.]

- Only inherited methods may be overridden, and remember that private methods are not inherited.
- A subclass uses `super.overriddenMethodName()` to call the superclass version of an overridden method.
- Overloading means reusing a method name, but with different arguments.
- Overloaded methods
 - Must have different argument lists
 - May have different return types, if argument lists are also different
 - May have different access modifiers
 - May throw different exceptions



Overriding and Overloading (Objectives 1.5 and 5.4) [contd.]

- Methods from a superclass can be overloaded in a subclass.
- Polymorphism applies to overriding, not to overloading
- Object type (not the reference variable's type), determines which overridden method is used at runtime.
- Reference type determines which overloaded method will be used at compile time.



Inheritance

(Objective 5.5)

- Inheritance is a mechanism that allows a class to be a subclass of a superclass, and thereby inherit variables and methods of the superclass.
- Inheritance is a key concept that underlies IS-A, polymorphism, overriding, overloading, and casting.
- All classes (except class Object), are subclasses of type Object, and therefore they inherit Object's methods.