**Programming with Java - Inheritance and Polymorphism**

**Key Terms**

• **abstract class**—A class that is declared with the reserved word abstract. An object of an abstract class cannot be instantiated.

• **abstract method**—A method that has a heading but no body.

• **aggregation**—refer to composition

• **base class**—The existing class, or superclass from which other classes are derived.

• **composition**—A "has a" relationship between two classes.

• **derived class**—A subclass created from an existing class.

• **dynamic binding**—The method to be executed is determined at execution time rather than compile time. Also called run-time binding.

• **inheritance**—A meaningful way to relate two or more classes with the "is-a" relationship between them.

• **interface**—A type of class that contains only abstract methods and/or named constants and is defined using the reserved word interface rather than class.

• **late binding**— refer to dynamic binding

• **multiple inheritance**—The subclass is derived from more than one superclass. Java does not support multiple inheritance, but instead uses interfaces.

• **overriding**—Declaring a method in a subclass with the same signature as a method in its superclass.

• **polymorphic reference variable**—A reference variable that can refer to objects of its own class or to objects of the subclasses inherited from its class.

• **polymorphism**—Assigning multiple meanings to the same method name.

• **redefining**—refer to overriding

• **run-time binding**— refer to dynamic binding

• **single inheritance**—The subclass is derived from only one existing superclass.

• **subclass**—A new class created from an existing class.

• **superclass**—The existing class or base class from which other classes are derived.

**Programming with Java - Inheritance and Polymorphism**

Inheritance

1. Inheritance lets you create new classes from existing classes without changing the existing class.

2. Any new class that you create from an existing class is called a subclass or derived class, and existing classes are called superclasses or base classes.

3. The subclass inherits the properties of the superclass, and rather than create completely new classes from scratch, you can take advantage of inheritance and reduce software complexity.

4. Inheritance can be viewed as a tree-like, or hierarchical, structure, wherein a superclass is shown with its subclasses.

5. The general syntax for deriving a class from an existing class uses the reserved word extends.

6. Each subclass can, in turn, become a superclass for a future subclass.

7. With single inheritance, the subclass is derived from a single superclass, whereas in multiple inheritance, the subclass is derived from more than one superclass.

8. Java only supports single inheritance. That is, in Java, a class can extend the definition of only one class.

**Programming Tip**

The definition of the subclass is typically placed in a separate file.

**Using Methods of the Superclass in a Subclass**

1. A subclass contains its own methods plus the methods of the superclass.

2. To override or redefine a public method of the superclass in the subclass, the corresponding method in the subclass must have the same name, the same type, and the same formal parameter list.

3. If the corresponding method in the superclass and the subclass has the same name but different parameters, this is method overloading in the subclass, which is also allowed.

4. Java supports method overriding and method overloading.

5. If the subclass overrides a public method of a superclass, then you specify a call to that public method of the superclass by using the reserved word super followed by the dot operator followed by the method name with an appropriate parameter list.

6. If the subclass does not override a public method of the superclass, you can specify a call to that public method by using the name of the method and an appropriate parameter list.

**Programming Tip**

Reserved word super can be used to refer to a class’s immediate superclass. To access protected and public members of the superclass, you can use super followed by a dot (.). This can be useful when a superclass’s methods have been overridden in the subclass, but you still have reason to access the superclass’s methods.

Constructors of the Superclass and Subclass

1. A subclass can have its own private data members and its own constructors.

2. A constructor typically serves to initialize instance variables, and when we instantiate a subclass object, this object inherits the instance variables of the superclass, but the subclass object cannot directly access the private instance variables of the superclass.

3. The methods of the subclass cannot directly access the private members of the superclass.

4. As a consequence, the constructors of the subclass can (directly) initialize only the instance variables of the subclass.

5. When a subclass object is instantiated, to initialize the (private) instance variables, it must also automatically execute one of the constructors of the superclass.

6. A call to a constructor of the superclass is specified in the definition of a subclass constructor by using the reserved word super.

7. The call to the constructor of the superclass must be the first statement in the subclass constructor.

**Protected Members of a Class**

1. Subclasses cannot access private members of the superclass directly.

2. If a member of a superclass needs to be accessed in a subclass yet still prevent access outside of the class, it must be declared using the modifier protected.

**Programming Tip**

In an inheritance hierarchy, the public and protected members of a superclass are directly accessible in a subclass across any number of generations, that is, at any level in the hierarchy.

**Programming Tip**

Even though the public and protected data members of a superclass are directly accessible in a subclass, it should be the responsibility of the superclass to properly initialize these data members.

**Protected Access vs Package Access**

1. When a member of the class is not given any modifier, then the Java system gives that member default package access. This means that any class within the package can access this member.

**Review Quiz 1**

1. True or False: The modifier private is used if a member of a superclass needs to be accessed by a subclass.

**Answer: False**

2. Subclasses inherit properties from \_\_\_\_\_\_\_\_\_\_.

**Answer: superclasses**

3. If the corresponding method in the superclass and the subclass has the same name but different parameters, this is an example of method \_\_\_\_\_\_\_\_\_\_.

**Answer: overloading**

4. If the corresponding method in the superclass and the subclass has the same name and same parameter list, this is an example of method \_\_\_\_\_\_\_\_\_\_.

**Answer: overriding**

5. True or False: Methods of a subclass can override or overload members of a superclass.

**Answer: True**

**class Object**

1. Every Java class is automatically provided with the method toString, and if a class does not provide its own toString method, the default toString is invoked.

2. The method toString comes from the Java class Object and is a public member of this class.

3. If you define a class and do not use the reserved word extends to derive it from an existing class, then the class you define is automatically considered to be derived from the class Object.

4. The class Object directly or indirectly becomes the superclass of every class in Java.

5. Using the mechanism of inheritance, every public member of the class Object can be overridden and/or can be invoked by every object of any class type.

**Programming Tip**

It is always a good idea to override the equals and toString method, even if it is not apparent in the design of the class that these methods will be of any use. Both of these methods can be useful while debugging code.

**Abstract Methods and Classes**

1. An abstract method is a method that has only the heading with no body, and the heading of an abstract method contains the reserved word abstract and ends with a semicolon.

2. An abstract class is a class that is declared with the reserved word abstract in its heading.

3. Abstract classes are used as superclasses from which other subclasses within the same context can be derived, and they can be used to force subclasses to provide certain methods.

**Review Quiz 2**

1. The class InputStreamReader is derived from the class Reader, which is derived from the class \_\_\_\_\_\_\_\_\_\_.

**Answer: Object**

2. To be able to handle a variety of events, Java allows a class to implement more than one \_\_\_\_\_\_\_\_\_\_.

**Answer: interface**

3. An abstract method is a method that has only the \_\_\_\_\_\_\_\_\_\_ with no body.

**Answer: heading**

4. True or False: Composition is an “is-a” relationship.

**Answer: False**

5. True or False: You cannot instantiate an object of an abstract class type. You can only declare a reference variable of an abstract class type.

**Answer: True**

**Additional Resources**

1. An article comparing abstract classes to interfaces in Java:

<www.javaworld.com/javaworld/javaqa/2001-04/03-qa-0420-abstract.html>

2. More information about interfaces in Java:

<http://download.oracle.com/javase/tutorial/java/concepts/interface.html>

3. More information about polymorphism in Java:

<http://download.oracle.com/javase/tutorial/java/IandI/polymorphism.html>

4. Dynamic vs. static binding in Java:

<http://geekexplains.blogspot.com/2008/06/dynamic-binding-vs-static-binding-in.html>