Inheritance and Polymorphism

1. Inheritance is mechanism of creating a new class from an existing one. The new class created is called the subclass, or child class or derived class and the existing class is called the super class or the parent class.
2. The extends keyword is used to create a subclass. Only one class can be listed after the keyword extends.
3. There is no limit to how many times a class can be a parent class. However, a subclass can have only one parent class.
4. The subclass inherits ALL of the public and protected state and behavior of the class that it extends.
5. Constructors are NOT inherited by a subclass.
6. A subclass can inherit from superclasses many levels away. (ex. If Dog is a subclass of Mammal, and Mammal is a subclass of Animal, then Dog inherits from both Mammal and Animal classes.
7. A subclass CANNOT access the private members of any of its superclasses.
8. The keyword super provides a reference to the superclass in the same way the this keyword provides access to the object in which it is used.
9. When a subclass is instantiated, Java will attempt to invoke the default constructor of all the superclasses, starting with the constructor in the highest superclass in the heritance hierarchy down to the lowest.
10. If you are extending a class that does not have a default constructor, you must use super in the constructor of the subclass to indicate which constructor of the superclass to invoke.
11. Methods with the same name, same signature but reside in different classes are called overridden methods. The methods of the subclass always overrides the superclass version.
12. Using the keyword final in a method stops a method from being overridden.
13. Using the keyword final placed before the class keyword in the class definition prevents the class from being inherited.
14. In an abstract class, one or more methods are defined, but no implementation is provided.
15. An abstract method does not have a method body, and the declaration of the method ends with a semi colon.
16. It is not possible to create an instance of an abstract class. An abstract class exists so other classes can extend it.
17. An abstract class cannot declare a constructor.
18. An interface is a set of method headers and constant definitions. It contains absolutely no executable code.
19. The purpose of an interface is to specify the common behaviors that a group of classes are required to implement.
20. All methods in an interface are public even if the public modifier is omitted.
21. An interface cannot define instance variable, but can define constants that are declared both static and final.
22. An interface cannot be instantiated.
23. A class that implements an interface must provide implementation for ALL the methods of the interface
24. Class can implement more than one interface but can extend only one class.
25. Using super(); in the constructor of the subclass is redundant. If you omit the call to the super constructor, it will be automatically invoked. If you want to explicitly call the super default constructor (which is really a matter of style), it must be the first line of the subclass constructor.
26. If there is *no* superclass default constructor but there is a subclass default constructor, an error will occur.
27. When the super constructor with parameters is used to pass in parameters from the subclass, it must be the first line of the subclass constructor.

Additional resources:

http://www.codejava.net/java-core/the-java-language/12-rules-and-examples-about-inheritance-in-java

**Exercise**

public class Superclass {

public Superclass () {

System.out.println (“Superclass constructor”); }

public void commonMethod() {

System.out.println (“commonMethod from Superclass”); }

public void MethodA() {

commonMethod();

MethodB() ;

}

public void MethodB() {

System.out.println (“MethodB from SuperClass”);

}

}

public class Subclass {

public Subclass () {

System.out.println (“Subclass constructor”);

}

public void commonMethod() {

System.out.println (“commonMethod from Subclass”);

}

public void onlyInSub() {

System.out.println (“onlyInSub from Subclass”);

}

}

public class Tester {

public static void main (String [] args) {

}

**1. What is the output of the following tester program?**

Subclass subc = new Subclass();

System.out.print (“1. “); subc.commonMethod();

System.out.print (“2. “); subc.MethodB();

System.out.print (“3. “); subc.MethodA();

System.out.println();

Superclass superRef = new Subclass();

System.out.print (“4. “); superRef.commonMethod();

System.out.print (“5. “); superRef.MethodB();

System.out.print (“6. “); superRef.MethodA();

System.out.print (“7. “); superRef.onlyInSub();

System.out.print (“8. “); ((Subclass)superRef).onlyInSub();

**2. Add an instance variable and a parameterized constructor to both Superclass and Subclass. Remove the Superclass default constructor but leave the subclass default constructor. Describe what happens.**

**Note:**

* The constructor of Superclass is implicitly invoked when the subclass object is instantiated.
* Line 7 will cause a compiler error because the onlyInSub method is not defined in Superclass.
* Line 8 will correct the problem. Since the dot operator has precedence, an extra set of parentheses is necessary to specify the typecast happens before the execution of the method.

