

## Chapter 8 Class Inheritance and Interfaces

1. The printout is  
The default constructor of A is invoked
2. The default constructor of B attempts to invoke the default of constructor of A, but class A's default constructor is not defined.
3. Find these terms in this chapter.
4. The following lines are erroneous:

```
{  
    radius = radius; // Must use this.radius = radius  
}
```

class Cylinder extends Circle (missing extends)

```
{  
    Circle(radius); // Must use super(radius)  
    length = length; // Must use this.length = length  
}
```

```
public double findArea()  
{  
    return findArea()*length; // super.findArea()  
}
```

```
{  
    radius = radius; // Must use this.radius = radius  
}
```

class Cylinder extends Circle (missing extends)

```
{  
    Circle(radius); // Must use super(radius)  
    length = length; // Must use this.length = length  
}
```

```
public double findArea()  
{  
    return findArea()*length; // super.findArea()  
}
```

5. Indicate true or false for the following statements:

1. True.

2. False. (But yes in a subclass that extends the class where the protected datum is defined.)

3. True.

4. A final class can have instances.

**Answer:** True

5. An abstract class can have instances created using the constructor of the abstract class.

**Answer:** No, but an instance of its concrete subclass is also an instance of the parent abstract class.

6. A final class can be extended.

**Answer:** False

7. An abstract class can be extended.

**Answer:** True

8. A final method can be overridden.

**Answer:** False

9. You can always successfully cast a subclass to a superclass.

**Answer:** True

10. You can always successfully cast a superclass to a subclass.

**Answer:** False

11. An interface can be a separate unit and can be compiled into a bytecode file.

**Answer:** True

12. The order in which modifiers appear before a method is important.

**Answer:** False

13. A subclass of a non-abstract superclass cannot be abstract.

**Answer:** False

14. A subclass cannot override a concrete method in a superclass to declare it abstract.

**Answer:** False, This is rare, but useful when the implementation of the method in the superclass becomes invalid in the subclass. In this case, the subclass must be declared abstract.

6. Method overloading defines methods of the same name in a class. Method overriding modifies the methods that are defined in the superclasses.
7. Yes, because these two methods are defined in the Object class; therefore, they are available to all Java classes. The subclasses usually override these methods to provide specific information for these methods.

The toString() method returns a string representation of the object; the equals() method compares the contents of two objects to determine whether they are the same.

8. default visibility modifier.

9. protected.

10. c.

11. (circle instanceof Cylinder)

**Answer:** False

(cylinder instanceof Circle)

**Answer:** True

12. Yes, because you can always cast from subclass to superclass.

13. Yes.

- 14.

Is fruit instanceof Orange true? false

Is fruit instanceof Apple true? true

Is fruit instanceof GoldDelicious true? true

Is fruit instanceof Macintosh true? false

Is orange instanceof Orange true? true

Is orange instanceof Fruit true? true

Is orange instanceof Apple true? false

Suppose the method makeApple is defined in the Apple class. Can fruit invoke this method? Yes

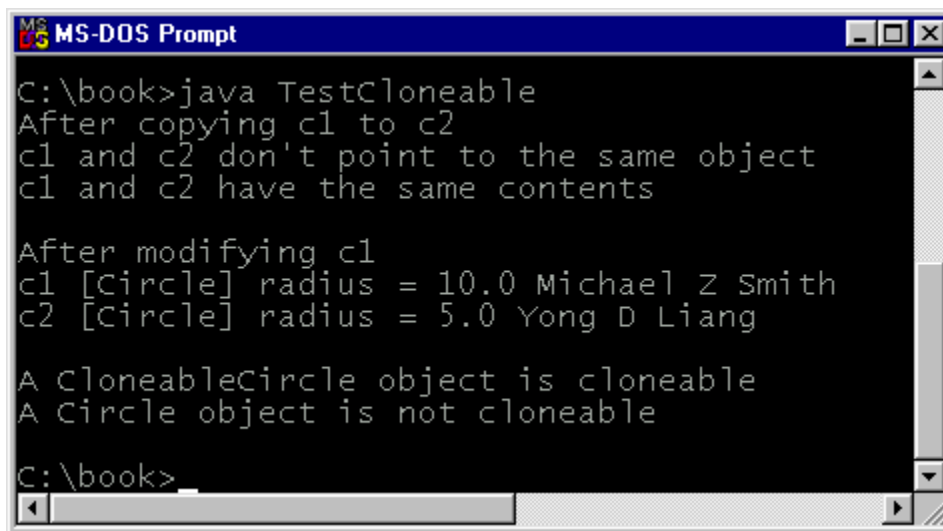
Can orange invoke this method? No

Suppose the method makeOrangeJuice is defined in the Orange class. Can orange invoke this method? Yes.

Can fruit invoke this method? No.

15. d.

16.



```
MS-DOS Prompt
C:\book>java TestCloneable
After copying c1 to c2
c1 and c2 don't point to the same object
c1 and c2 have the same contents

After modifying c1
c1 [Circle] radius = 10.0 Michael Z Smith
c2 [Circle] radius = 5.0 Yong D Liang

A CloneableCircle object is cloneable
A Circle object is not cloneable

C:\book>
```

17. a. Test is a subclass of A. A does not have a default constructor. Test does not have a default constructor unless it is explicitly defined.

18. B's constructor is invoked

A's constructor is invoked

19. d.

20. Yes.

21. No. The scope of the inner class is within its defining class.

22. Yes.

23. See the italicized lines.

```
// TestInterface.java: Use the Comparable interface
// and the generic max method to find max objects
public class TestInterfaceReviewQuestion23 {
    /** Main method */
    public static void main(String[] args) {
        // Create two comparable circles
        ComparableCircle circle1 = new ComparableCircle(5);
        ComparableCircle circle2 = new ComparableCircle(4);

        // Display the max circle
        Comparable circle = (Comparable)Max.max(circle1, circle2);
        System.out.println("The max circle's radius is " +
            ((Circle)circle).getRadius());
        System.out.println(circle);

        // Create two comparable cylinders
        ComparableCylinder cylinder1 = new ComparableCylinder(5, 2);
        ComparableCylinder cylinder2 = new ComparableCylinder(4, 5);

        // Display the max cylinder
        Comparable cylinder = (Comparable)Max.max(cylinder1, cylinder2);
        System.out.println();
        System.out.println("cylinder1's volume is " +
            cylinder1.findVolume());
        System.out.println("cylinder2's volume is " +
            cylinder2.findVolume());
        System.out.println("The max cylinder's \tradius is " +
            ((Cylinder)cylinder).getRadius() + "\n\t\t\tlength is " +
            ((Cylinder)cylinder).getLength() + "\n\t\t\tvolume is " +
            ((Cylinder)cylinder).findVolume());
        System.out.println(cylinder);
    }
}
```