Inheritance

Exercises and solutions

1. What keyword do you use in a class declaration to inherit your class from another class?

**Answer:**

The keyword is extends.

1. What are the names of the superclass and subclass in the following class declaration?  
     
   public class Letter extends Document

**Answer:**

The superclass name is Document and the subclass is Letter.

1. Write the fully qualified name of the superclass of class A, which is declared as follows:  
     
   public class A {  
   }

**Answer:**

The class A implicitly inherits from the java.lang.Object class, its superclass name is java.lang.Object.

1. How many superclasses can a class have in Java?

**Answer:**

A class can have only one superclass. A class in Java, except the java.lang.Object class, always has one superclass.

1. What keyword do you use to call the constructor of the superclass? Write the statement that calls the superclass constructor, which takes a string as an argument. The argument value is "Hello".

**Answer:**

The keyword super is used to call the constructor of the superclass. The following statement calls the superclass' constructor with an argument "Hello":

super(“Hello”);

1. What types of members of a superclass are inherited by a subclass: public, private, protected, and package-level?

**Answer:**

A subclass inherits non-private members of its superclass. The public and protected members are always inherited by a subclasses. A package-level class member is inherited only if the superclass and subclass are in the same package.

1. Name the annotation that you should use when you override a method in your class, so the compiler can verify your intent of overriding the method.

**Answer:**

@Override

1. How do you call an overridden instance method of a superclass from a subclass? Consider the following snippet of code:   
     
   public class A {  
    public void print() {  
    System.out.println("A");  
    }  
   }  
     
   public class B extends A {  
    @Override  
    public void print() {  
    /\* Your one line code goes here \*/  
     
    System.out.println("B");  
    }  
     
    public static void main(String[] args) {  
    new B().print();

}  
}  
  
Complete the code inside the print() method of class B, so when you run class B, it should print the following. You are to call the print() method of class A:  
  
A  
B

**Solution:**

public class B extends A {

@Override

public void print() {

**super.print();**

System.out.println("B");

}

public static void main(String[] args) {  
 new B().print();

}

}

1. Write the reasons why the following class declaration does not compile.  
     
   public abstract final class A {  
    // Code goes here   
   }

**Answer:**

An abstract class is meant to be subclassed. A final class cannot be subclassed. The class A does not compile because it has been declared abstract as well as final.

1. Write the reasons why the following declarations of class B and class C do not compile.   
     
   public class A {  
    public A(int x) {  
    }  
   }  
     
   public class B extends A {  
   }  
     
   public class C extends A {  
    public C() {

}  
}

**Answer:**

Both classes B and C inherit from class A. A subclass needs to call one of the constructors of its superclass. Both classes B and C attempt to call a no-args constructor in class A; however, class A does not contain a no-args constructor. This is the reason that these classes do not compile.

1. What is the difference between method overloading and method overriding?

**Answer:**

Refer table 20-2 in this chapter.

1. Consider the following declarations for class A and class B. What will be printed when class B is run? Is the declaration of method m1() in class B is a case of method overriding or method overloading? Explain your answer.  
     
   public class A {  
    public void m1(int x) {  
    System.out.println("A.m1(): " + x);  
    }  
   }  
     
   public class B extends A {  
    public void m1(Integer x) {   
    System.out.println("B.m1(): " + x);  
    }   
     
    public static void main(String[] args) {  
    B b = new B();  
    b.m1(100);  
    }  
   }

**Answer:**

The output is as follows:

A.m1(): 100

The m1(Integer) method in class B is a case of method overloading. Class B inherits m1(int) from its superclass A and it declares m1(Integer).

1. Consider the following two class declarations:  
     
   public class A {  
   }  
     
   public class B extends A {  
   }  
     
   One of the following statements does not compile. Describe the reason behind the compile-time error and fix it. Identify examples of upcasting and downcasting in the following statements.  
     
   A a = new B();  
   B b = new B();  
   a = b;  
   b = a;

**Answer:**

The last statement

b = a;

fails to compile because as it tries to assign variable a of type A (superclass) to variable b to type B (subclass). You can fix this compile-time error using a cast as follows:

b = (B) a;

The assignment "a = b" is a case of upcasting and the assignment "b = a" is a case of downcasting.

1. What is the difference between early binding and late binding? Which type of binding is solely decided by the compiler?

**Answer:**

Early binding occurs at compile time and is also known as static binding or compile-time binding. It is used for static and non-static fields, static methods and non-static final methods.

Late binding occurs at runtime and is also known as dynamic binding or runtime binding. It is used for all non-static, non-final methods.

1. Write the output when the following class B is run. This exercise is to test your knowledge of early binding and late binding.  
     
   public class A {  
    public void m1() {  
    System.out.println("A.m1()");  
    }  
     
    public static void m2() {  
    System.out.println("A.m2()");  
    }  
   }  
     
   public class B extends A {  
    @Override  
    public void m1() {  
    System.out.println("B.m1()");  
    }  
     
    public static void m2() {  
    System.out.println("B.m2()");  
    }  
     
    public static void main(String[] args) {  
    A a = new B();  
    a.m1();  
    a.m2();  
    ((B)a).m2();  
    A.m2();  
    B.m2();  
    }  
   }

**Answer:**

B.m1()

A.m2()

B.m2()

A.m2()

B.m2()

1. Name the operator that you are supposed to use before downcasting a reference, so the downcasting always succeeds.

Answer:

instanceof

1. Write the output of the following snippet of code:  
     
   public class A {   
   }  
   public class B extends A {  
   }  
     
   A a = new B();   
   System.out.println("a instanceof A: " + (a instanceof A));  
   System.out.println("a instanceof B: " + (a instanceof B));  
   System.out.println("a instanceof Object: " + (a instanceof Object));  
   System.out.println("null instanceof A: " + (null instanceof A));  
   System.out.println("null instanceof B: " + (null instanceof B));

**Answer:**

a instanceof A: true

a instanceof B: true

a instanceof Object: true

null instanceof A: false

null instanceof B: false

1. Explain why the following declaration for class B does not compile.  
     
   public abstract class A {  
    public abstract void print();  
   }  
     
   public class B extends A {  
   }

**Answer:**

The declaration for class B generates a compile-time error because class B is neither abstract nor does it override the abstract method print() in A

Explain why the following declaration for class B does not compile.  
  
public class A {  
 private A() {  
 System.out.println("Hello");  
 }  
}  
  
public class B extends A {   
}

**Answer:**

Class B needs to call the constructor of class A, which it cannot call because class A declares it as private. This is the reason for the compile-time error.

1. Write the output when the following class B is run. This exercise is to test your knowledge of field hiding, method overriding, and use of the super keyword to call the method of the superclass.  
     
   public class A {  
    protected int x = 100;  
     
    public A() {  
    System.out.println("x = " + x);  
    }  
     
    public void print() {  
    System.out.println("x = " + x);  
    }  
   }  
     
   public class B extends A {   
    private final int x = 200;  
     
    public B() {  
    System.out.println("x = " + x);  
    }  
     
    @Override  
    public void print() {  
    super.print();  
    System.out.println("x = " + x);  
    }  
     
    public static void main(String[] args) {  
    A a = new B();  
    a.print();  
    }  
   }

**Answer:**

x = 100

x = 200

x = 100

x = 200