

## Quiz 11 Inheritance and Polymorphism

1. Object-oriented programming allows you to derive new classes from existing classes. This is called \_\_\_\_\_.

- a. encapsulation
- b. inheritance
- c. abstraction
- d. generalization

2. Which of the following statements are true?

- a. A subclass is a subset of a superclass.
- b. A subclass is usually extended to contain more functions and more detailed information than its superclass.
- c. "class A extends B" means A is a subclass of B.
- d. "class A extends B" means B is a subclass of A.

3. Suppose you create a class Square to be a subclass of GeometricObject. Analyze the following code:

```
class Square extends GeometricObject {  
    double length;  
  
    Square(double length) {  
        GeometricObject(length);  
    }  
}
```

- a. The program compiles fine, but you cannot create an instance of Square because the constructor does not specify the length of the Square.
- b. The program has a compile error because you attempted to invoke the GeometricObject class's constructor illegally.
- c. The program compiles fine, but it has a runtime error because of invoking the Square class's constructor illegally.

4. Analyze the following code:

```
public class A extends B {  
  
}
```

```
class B {
    public B(String s) {
    }
}
```

- a. The program has a compile error because A does not have a default constructor.
- b. The program has a compile error because the default constructor of A invokes the default constructor of B, but B does not have a default constructor.

c. The program would compile fine if you add the following constructor into A:  
A(String s) { }

d. The program would compile fine if you add the following constructor into A:  
A(String s) { super(s); }

5. Analyze the following code:

```
public class Test extends A {
    public static void main(String[] args) {
        Test t = new Test();
        t.print();
    }
}

class A {
    String s;

    A(String s) {
        this.s = s;
    }

    public void print() {
        System.out.println(s);
    }
}
```

- a. The program does not compile because Test does not have a default constructor Test().

- b. The program has an implicit default constructor Test(), but it cannot be compiled, because its super class does not have a default constructor. The program would compile if the constructor in the class A were removed.
- c. The program would compile if a default constructor A(){ } is added to class A explicitly.
- d. The program compiles, but it has a runtime error due to the conflict on the method name print.

6. What is the output of running class C?

```
class A {  
    public A() {  
        System.out.println(  
            "The default constructor of A is invoked");  
    }  
}  
  
class B extends A {  
    public B() {  
        System.out.println(  
            "The default constructor of B is invoked");  
    }  
}  
  
public class C {  
    public static void main(String[] args) {  
        B b = new B();  
    }  
}
```

- a. Nothing displayed
- b. "The default constructor of B is invoked"
- c. "The default constructor of A is invoked" followed by "The default constructor of B is invoked"
- d. "The default constructor of B is invoked" followed by "The default constructor of A is invoked"
- e. "The default constructor of A is invoked"

7. Which of the following is incorrect?

- a. A constructor may be static.
- b. A constructor may be private.
- c. A constructor may invoke a static method.
- d. A constructor may invoke an overloaded constructor.
- e. A constructor invokes its superclass no-arg constructor by default if a constructor does not invoke an overloaded constructor or its superclass's constructor.

8. Which of the statements regarding the super keyword is incorrect?

- a. You can use super to invoke a super class constructor.
- b. You can use super to invoke a super class method.
- c. You can use super.super.p to invoke a method in superclass's parent class.
- d. You cannot invoke a method in superclass's parent class.

9. Analyze the following code:

```
public class Test {  
    public static void main(String[] args) {  
        B b = new B();  
        b.m(5);  
        System.out.println("i is " + b.i);  
    }  
}  
  
class A {  
    int i;  
    public void m(int i) {  
        this.i = i;  
    }  
}  
  
class B extends A {  
    public void m(String s) {  
    }  
}
```

- a. The program has a compile error, because m is overridden with a different signature in B.

- b. The program has a compile error, because b.m(5) cannot be invoked since the method m(int) is hidden in B.
- c. The program has a runtime error on b.i, because i is not accessible from b.
- d. The method m is not overridden in B. B inherits the method m from A and defines an overloaded method m in B.

10. The getValue() method is overridden in two ways. Which one is correct?

I:

```
public class Test {  
    public static void main(String[] args) {  
        A a = new A();  
        System.out.println(a.getValue());  
    }  
}  
  
class B {  
    public String getValue() {  
        return "Any object";  
    }  
}  
  
class A extends B {  
    public Object getValue() {  
        return "A string";  
    }  
}
```

II:

```
public class Test {  
    public static void main(String[] args) {  
        A a = new A();  
        System.out.println(a.getValue());  
    }  
}  
  
class B {  
    public Object getValue() {
```

```

        return "Any object";
    }
}

class A extends B {
    public String getValue() {
        return "A string";
    }
}

```

- a. I
- b. II
- c. Both I and II
- d. Neither

11. Which of the following statements are true?

- a. To override a method, the method must be defined in the subclass using the same signature and compatible return type as in its superclass.
- b. Overloading a method is to provide more than one method with the same name but with different signatures to distinguish them.
- c. It is a compile error if two methods differ only in return type in the same class.
- d. A private method cannot be overridden. If a method defined in a subclass is private in its superclass, the two methods are completely unrelated.
- e. A static method cannot be overridden. If a static method defined in the superclass is redefined in a subclass, the method defined in the superclass is hidden.

12. Which of the following statements are true?

- a. A method can be overloaded in the same class.
- b. A method can be overridden in the same class.
- c. If a method overloads another method, these two methods must have the same signature.
- d. If a method overrides another method, these two methods must have the same signature.
- e. A method in a subclass can overload a method in the superclass.

13. Analyze the following code:

```

public class Test {
    public static void main(String[] args) {

```

```

        new B();
    }
}
class A {
    int i = 7;
    public A() {
        System.out.println("i from A is " + i);
    }
    public void setI(int i) {
        this.i = 2 * i;
    }
}
class B extends A {
    public B() {
        setI(20);
        // System.out.println("i from B is " + i);
    }
    @Override
    public void setI(int i) {
        this.i = 3 * i;
    }
}

```

- a. The constructor of class A is not called.
- b. The constructor of class A is called and it displays "i from A is 7".
- c. The constructor of class A is called and it displays "i from A is 40".
- d. The constructor of class A is called and it displays "i from A is 60".

14. Analyze the following code:

```

public class Test {
    public static void main(String[] args) {
        new B();
    }
}

```

```

}
class A {
    int i = 7;
    public A() {
        setI(20);
        System.out.println("i from A is " + i);
    }
    public void setI(int i) {
        this.i = 2 * i;
    }
}
class B extends A {
    public B() {
        // System.out.println("i from B is " + i);
    }
    @Override
    public void setI(int i) {
        this.i = 3 * i;
    }
}

```

- a. The constructor of class A is not called.
- b. The constructor of class A is called and it displays "i from A is 7".
- c. The constructor of class A is called and it displays "i from A is 40".
- d. The constructor of class A is called and it displays "i from A is 60".

15. Analyze the following code:

```

public class Test {
    public static void main(String[] args) {
        Object a1 = new A();
        Object a2 = new Object();
        System.out.println(a1);
        System.out.println(a2);
    }
}

```



```

    }
}
class A {
    int x;

    @Override
    public String toString() {
        return "A's x is " + x;
    }
}

```

- a. The program cannot be compiled, because `System.out.println(a1)` is wrong and it should be replaced by `System.out.println(a1.toString());`
- b. When executing `System.out.println(a1)`, the `toString()` method in the `Object` class is invoked.
- c. When executing `System.out.println(a2)`, the `toString()` method in the `Object` class is invoked.
- d. When executing `System.out.println(a1)`, the `toString()` method in the `A` class is invoked.

17. Given the following code, find the compile error.

```

public class Test {
    public static void main(String[] args) {
        m(new GraduateStudent());
        m(new Student());
        m(new Person());
        m(new Object());
    }

    public static void m(Student x) {
        System.out.println(x.toString());
    }
}

class GraduateStudent extends Student {
}

class Student extends Person {
}

```

```

@Override
public String toString() {
    return "Student";
}
}
class Person extends Object {
    @Override
    public String toString() {
        return "Person";
    }
}

```

- a. m(new GraduateStudent()) causes an error
- b. m(new Student()) causes an error
- c. m(new Person()) causes an error
- d. m(new Object()) causes an error

18. What is the output of the following code?

```

public class Test {
    public static void main(String[] args) {
        new Person().printPerson();
        new Student().printPerson();
    }
}
class Student extends Person {
    @Override
    public String getInfo() {
        return "Student";
    }
}
class Person {
    public String getInfo() {
        return "Person";
    }
}

```

```

    }

    public void printPerson() {
        System.out.println(getInfo());
    }
}

```

- a. Person Person
- b. Person Student
- c. Student Student
- d. Student Person

19. What is the output of the following code?

```

public class Test {
    public static void main(String[] args) {
        new Person().printPerson();
        new Student().printPerson();
    }
}

class Student extends Person {
    private String getInfo() {
        return "Student";
    }
}

class Person {
    private String getInfo() {
        return "Person";
    }

    public void printPerson() {
        System.out.println(getInfo());
    }
}

```

- a. Person Person
- b. Person Student
- c. Student Student
- d. Student Person

16. Which of the following statements is false?

- a. You can always pass an instance of a subclass to a parameter of its superclass type. This feature is known as polymorphism.
- b. The compiler finds a matching method according to parameter type, number of parameters, and order of the parameters at compile time.
- c. A method may be implemented in several subclasses. The Java Virtual Machine dynamically binds the implementation of the method at runtime.
- d. Dynamic binding can apply to static methods.
- e. Dynamic binding can apply to instance methods.

20. Which of the following are Java keywords?

- a. instanceof
- b. instanceof
- c. cast
- d. casting

21. Assume Cylinder is a subtype of Circle. Analyze the following code:

```
Cylinder cy = new Cylinder(1, 1);
```

```
Circle c = cy;
```

- a. The code has a compile error.
- b. The code has a runtime error.
- c. The code is fine.

22. Assume Cylinder is a subtype of Circle. Analyze the following code:

```
Circle c = new Circle (5);
```

```
Cylinder c = cy;
```

- a. The code has a compile error.
- b. The code has a runtime error.
- c. The code is fine.

23. Given the following classes and their objects:

```
class C1 {};
```

```
class C2 extends C1 {};
```

```
class C3 extends C1 {};
```

```
C2 c2 = new C2();
```

```
C3 c3 = new C3();
```

Analyze the following statement:

```
c2 = (C2)((C1)c3);
```

- a. c3 is cast into c2 successfully.
- b. You will get a runtime error because you cannot cast objects from sibling classes.
- c. You will get a runtime error because the Java runtime system cannot perform multiple casting in nested form.
- d. The statement is correct.

24. Given the following code, which of the following expressions evaluates to false?

```
class C1 {}
```

```
class C2 extends C1 { }
```

```
class C3 extends C2 { }
```

```
class C4 extends C1 {}
```

```
C1 c1 = new C1();
```

```
C2 c2 = new C2();
```

```
C3 c3 = new C3();
```

```
C4 c4 = new C4();
```

- a. c1 instanceof C1
- b. c2 instanceof C1
- c. c3 instanceof C1
- d. c4 instanceof C2

25. Analyze the following code:

```
public class Test {
```

```
    public static void main(String[] args) {
```

```
        String s = new String("Welcome to Java");
```

```
        Object o = s;
```

```
String d = (String)o;  
}  
}
```

- a. When assigning s to o in Object o = s, a new object is created.
- b. When casting o to s in String d = (String)o, a new object is created.
- c. When casting o to s in String d = (String)o, the contents of o is changed.
- d. s, o, and d reference the same String object.

26. You can assign \_\_\_\_\_ to a variable of Object[] type.

- a. new char[100]
- b. new int[100]
- c. new double[100]
- d. new String[100]
- e. new java.util.Date[100]

27. The equals method is defined in the Object class. Which of the following is correct to override it in the String class?

- a. public boolean equals(String other)
- b. public boolean equals(Object other)
- c. public static boolean equals(String other)
- d. public static boolean equals(Object other)

28. Which of the following statements are true?

- a. Override the equals(Object) method in the Object class whenever possible.
- b. Override the toString() method in the Object class whenever possible.
- c. A public default no-arg constructor is assumed if no constructors are defined explicitly.
- d. You should follow standard Java programming style and naming conventions. Choose informative names for classes, data fields, and methods.

29. What is the output of the following code?

```
public class Test {  
    public static void main(String[] args) {  
        Object o1 = new Object();  
        Object o2 = new Object();  
        System.out.print((o1 == o2) + " " + (o1.equals(o2)));  
    }  
}
```

```
}
```

```
}
```

a. false false

b. true true

c. false true

d. true false

30. What is the output of the following code?

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = new String("Java");  
        String s2 = new String("Java");  
        System.out.print((s1 == s2) + " " + (s1.equals(s2)));  
    }  
}
```

a. false false

b. true true

c. false true

d. true false

31. Given two reference variables t1 and t2, if t1 == t2 is true, t1.equals(t2) must be \_\_\_\_\_.

a. true

b. false

32. Given two reference variables t1 and t2, if t1.equals(t2) is true, t1 == t2 \_\_\_\_\_.

a. is always true

b. is always false

c. may be true or false

33. Analyze the following code.

```
// Program 1:
```

```
public class Test {  
    public static void main(String[] args) {  
        Object a1 = new A();
```

```

        Object a2 = new A();

        System.out.println(a1.equals(a2));
    }
}

```

```

class A {
    int x;

    public boolean equals(Object a) {
        return this.x == ((A)a).x;
    }
}

```

// Program 2:

```

public class Test {
    public static void main(String[] args) {
        Object a1 = new A();
        Object a2 = new A();
        System.out.println(a1.equals(a2));
    }
}

class A {
    int x;

    public boolean equals(A a) {
        return this.x == a.x;
    }
}

```

- a. Program 1 displays true and Program 2 displays true
- b. Program 1 displays false and Program 2 displays true
- c. Program 1 displays true and Program 2 displays false
- d. Program 1 displays false and Program 2 displays false

34. Analyze the following code.

// Program 1:



```

public class Test {

    public static void main(String[] args) {

        Object a1 = new A();

        Object a2 = new A();

        System.out.println(a1.equals(a2));

    }

}

```

```

class A {

    int x;

    public boolean equals(A a) {

        return this.x == a.x;

    }

}

```

// Program 2:

```

public class Test {

    public static void main(String[] args) {

        A a1 = new A();

        A a2 = new A();

        System.out.println(a1.equals(a2));

    }

}

class A {

    int x;

    public boolean equals(A a) {

        return this.x == a.x;

    }

}

```

- a. Program 1 displays true and Program 2 displays true
- b. Program 1 displays false and Program 2 displays true
- c. Program 1 displays true and Program 2 displays false
- d. Program 1 displays false and Program 2 displays false

35. Analyze the following code.

// Program 1

```
public class Test {  
    public static void main(String[] args) {  
        Object a1 = new A();  
        Object a2 = new A();  
        System.out.println(((A)a1).equals((A)a2));  
    }  
}  
  
class A {  
    int x;  
  
    public boolean equals(A a) {  
        return this.x == a.x;  
    }  
}
```

// Program 2

```
public class Test {  
    public static void main(String[] args) {  
        A a1 = new A();  
        A a2 = new A();  
        System.out.println(a1.equals(a2));  
    }  
}  
  
class A {  
    int x;  
    public boolean equals(A a) {  
        return this.x == a.x;  
    }  
}
```

- a. Program 1 displays true and Program 2 displays true
- b. Program 1 displays false and Program 2 displays true
- c. Program 1 displays true and Program 2 displays false
- d. Program 1 displays false and Program 2 displays false

36. You can create an ArrayList using \_\_\_\_\_.

- a. new ArrayList[]
- b. new ArrayList[100]
- c. new ArrayList<>()
- d. ArrayList()

37. Invoking \_\_\_\_\_ removes all elements in an ArrayList x.

- a. x.remove()
- b. x.clean()
- c. x.delete()
- d. x.empty()
- e. x.clear()

38. Suppose ArrayList x contains two strings [Beijing, Singapore]. Which of the following methods will cause the list to become [Beijing, Chicago, Singapore]?

- a. x.add("Chicago")
- b. x.add(0, "Chicago")
- c. x.add(1, "Chicago")
- d. x.add(2, "Chicago")

39. Suppose ArrayList x contains two strings [Beijing, Singapore]. Which of the following method will cause the list to become [Beijing]?

- a. x.remove("Singapore")
- b. x.remove(0)
- c. x.remove(1)
- d. x.remove(2)

40. Suppose ArrayList x contains two strings [Beijing, Singapore]. Which of the following method will cause runtime errors?

- a. x.get(1)
- b. x.set(2, "New York");

- c. x.get(2)
- d. x.remove(2)
- e. x.size()

41. Invoking \_\_\_\_\_ returns the first element in an ArrayList x.

- a. x.first()
- b. x.get(0)
- c. x.get(1)
- d. x.get()

42. Invoking \_\_\_\_\_ returns the number of the elements in an ArrayList x.

- a. x.getSize()
- b. x.getLength(0)
- c. x.length(1)
- d. x.size()

43. Analyze the following code:

```
ArrayList<String> list = new ArrayList<String>();  
list.add("Beijing");  
list.add("Tokyo");  
list.add("Shanghai");  
list.set(3, "Hong Kong");
```

- a. The last line in the code causes a runtime error because there is no element at index 3 in the array list.
- b. The last line in the code has a compile error because there is no element at index 3 in the array list.
- c. If you replace the last line by list.add(3, "Hong Kong"), the code will compile and run fine.
- d. If you replace the last line by list.add(4, "Hong Kong"), the code will compile and run fine.

44. What is the output of the following code?

```
ArrayList<java.util.Date> list = new ArrayList<java.util.Date>();  
java.util.Date d = new java.util.Date();  
list.add(d);  
list.add(d);
```

```
System.out.println((list.get(0) == list.get(1)) + " "  
    + (list.get(0)).equals(list.get(1)));
```

- a. true false
- b. false true
- c. true true
- d. false false

45. What is the output of the following code?

```
ArrayList<String> list = new ArrayList<String>();  
String s1 = new String("Java");  
String s2 = new String("Java");  
list.add(s1);  
list.add(s2);  
System.out.println((list.get(0) == list.get(1)) + " "  
    + (list.get(0)).equals(list.get(1)));
```

- a. true false
- b. false true
- c. true true
- d. false false

46. Suppose an ArrayList list contains {"red", "green", "red", "green"}. What is the list after the following code?

```
list.remove("red");
```

- a. {"red", "green", "red", "green"}
- b. {"green", "red", "green"}
- c. {"green", "green"}
- d. {"red", "green", "green"}

47. Suppose an ArrayList list contains {"red", "red", "green"}. What is the list after the following code?

```
String element = "red";  
for (int i = 0; i < list.size(); i++)  
    if (list.get(i).equals(element))  
        list.remove(element);
```

- a. {"red", "red", "green"}
- b. {"red", "green"}
- c. {"green"}
- d. {}

48. Suppose an ArrayList list contains {"red", "red", "green"}. What is the list after the following code?

```
String element = "red";
for (int i = 0; i < list.size(); i++)
    if (list.get(i).equals(element)) {
        list.remove(element);
        i--;
    }
```

- a. {"red", "red", "green"}
- b. {"red", "green"}
- c. {"green"}
- d. {}

49. Suppose an ArrayList list contains {"red", "red", "green"}. What is the list after the following code?

```
String element = "red";
for (int i = list.size() - 1; i >= 0; i--)
    if (list.get(i).equals(element))
        list.remove(element);
```

- a. {"red", "red", "green"}
- b. {"red", "green"}
- c. {"green"}
- d. {}

50. The output from the following code is \_\_\_\_\_.

```
java.util.ArrayList<String> list
    = new java.util.ArrayList<String>();
list.add("New York");
java.util.ArrayList<String> list1 = list;
list.add("Atlanta");
```

```
list1.add("Dallas");
```

```
System.out.println(list1);
```

- a. [New York]
- b. [New York, Atlanta]
- c. [New York, Atlanta, Dallas]
- d. [New York, Dallas]

51. Show the output of the following code:

```
String[] array = {"red", "green", "blue"};
```

```
ArrayList<String> list = new ArrayList<>(Arrays.asList(array));
```

```
list.add(0, "red");
```

```
System.out.println(list);
```

- a. ["red", "green", "blue", "red"]
- b. ["red", "green", "blue"]
- c. ["red", "red", "green", "blue"]
- d. ["red", "green", "red", "blue"]

52. Analyze the following code:

```
Double[] array = {1, 2, 3};
```

```
ArrayList<Double> list = new ArrayList<>(Arrays.asList(array));
```

```
System.out.println(list);
```

- a. The code is correct and displays [1, 2, 3].
- b. The code is correct and displays [1.0, 2.0, 3.0].
- c. The code has a compile error because an integer such as 1 is automatically converted into an Integer object, but the array element type is Double.
- d. The code has a compile error because asList(array) requires that the array elements are objects.

53. Analyze the following code:

```
double[] array = {1, 2, 3};
```

```
ArrayList<Double> list = new ArrayList<>(Arrays.asList(array));
```

```
System.out.println(list);
```

- a. The code is correct and displays [1, 2, 3].
- b. The code is correct and displays [1.0, 2.0, 3.0].

- c. The code has a compile error because an integer such as 1 is automatically converted into an Integer object, but the array element type is Double.
- d. The code has a compile error because `asList(array)` requires that the array elements are objects.

54. Analyze the following code:

```
double[] c = {1, 2, 3};  
System.out.println(java.util.Collections.max(c));
```

- a. The code is correct and displays 3.
- b. The code is correct and displays 3.0.
- c. The code has a compile error on `Collections.max(c)`. `c` cannot be an array.
- d. The code has a compile error on `double[] c = {1, 2, 3}`.

55. Analyze the following code:

```
Integer[] c = {3, 5};  
java.util.Collections.shuffle(c);  
System.out.println(java.util.Arrays.toString(c));
```

- a. The code is correct and displays [3, 5].
- b. The code is correct and displays [5, 3].
- c. The code has a compile error on `Collections.shuffle(c)`. `c` cannot be an array.
- d. The code has a compile error on `Integer[] c = {3, 5}`.

56. What modifier should you use on a class so that a class in the same package can access it but a class (including a subclass) in a different package cannot access it?

- a. `public`
- b. `private`
- c. `protected`
- d. Use the default modifier.

57. What modifier should you use on the members of a class so that they are not accessible to another class in a different package, but are accessible to any subclasses in any package?

- a. `public`
- b. `private`
- c. `protected`
- d. Use the default modifier.

58. The visibility of these modifiers increases in this order:



- a. private, protected, none (if no modifier is used), and public.
- b. private, none (if no modifier is used), protected, and public.
- c. none (if no modifier is used), private, protected, and public.
- d. none (if no modifier is used), protected, private, and public.

59. A class design requires that a particular member variable must be accessible by any subclasses of this class, but otherwise not by classes which are not members of the same package. What should be done to achieve this?

- a. The variable should be marked public.
- b. The variable should be marked private.
- c. The variable should be marked protected.
- d. The variable should have no special access modifier.
- e. The variable should be marked private and an accessor method provided.

60. Which of the following statements is false?

- a. A public class can be accessed by a class from a different package.
- b. A private method cannot be accessed by a class in a different package.
- c. A protected method can be accessed by a subclass in a different package.
- d. A method with no visibility modifier can be accessed by a class in a different package.

61. Which statements are most accurate regarding the following classes?

```
class A {
    private int i;
    protected int j;
}

class B extends A {
    private int k;
    protected int m;
}
```

- a. An object of B contains data fields i, j, k, m.
- b. An object of B contains data fields j, k, m.
- c. An object of B contains data fields j, m.
- d. An object of B contains data fields k, m.

62. Which statements are most accurate regarding the following classes?

- In the class B, an instance method can only access i, j, k, m.
- In the class B, an instance method can only access j, k, m.
- In the class B, an instance method can only access j, m.
- In the class B, an instance method can only access k, m.

- a. class A { }
- b. class A {&nbsp;&nbsp;  private A() {&nbsp;&nbsp;&nbsp;&}}
- c. final class A { }
- d. class A {&nbsp;&nbsp;  protected A() {&nbsp;&nbsp;&}}