Java Software Development Quiz 9

1. Which of these field declarations are legal within the body of an interface?

Select the three correct answers.

(a) public static int answer = 42;

(b) int answer;

(c) final static int answer = 42;

(d) public int answer = 42;

(e) private final static int answer = 42;

2. Which statement is true about the following code?

```
// Filename: MyClass.java
abstract class MyClass implements Interface1, Interface2 {
   public void f() { }
   public void g() { }
}
interface Interface1 {
   int VAL_A = 1;
   int VAL_B = 2;
   void f();
   void g();
}
interface Interface2 {
   int VAL_B = 3;
   int VAL_C = 4;
   void g();
   void h();
}
```

Select the one correct answer.

(a) MyClass only implements Interface1. Implementation for void h() from Interface2 is missing.

(b) The declarations of void g() in the two interfaces conflict, therefore, the code will not compile.

(c) The declarations of int VAL_B in the two interfaces conflict, therefore, the code will not compile.

(d) Nothing is wrong with the code, it will compile without errors.

3. Which declaration can be inserted at (1) without causing a compilation error?

```
interface MyConstants {
    int r = 42;
    int s = 69;
    // (1) INSERT CODE HERE
}
```

Select the two correct answers.

(a) final double circumference = 2 * Math.PI * r;

(b) int total = total + r + s;

(c) int AREA = r * s;

(d) public static MAIN = 15;

(e) protected int CODE = 31337;

4. Given the following type and reference declarations, which assignment is legal?

```
// Type declarations:
interface I1 {}
interface I2 {}
class C1 implements I1 {}
class C2 implements I2 {}
class C3 extends C1 implements I2 {}
// Reference declarations:
C1 obj1;
C2 obj2;
C3 obj3;
```

Select the one correct answer.

(a) obj2 = obj1;

(b) obj3 = obj1;

(c) obj3 = obj2;

(d) I1 a = obj2;

(e) I1 b = obj3;

(f) I2 c = obj1;

5. Given the following class declarations and declaration statements, which assignment is legal at compile time?

```
// Class declarations:
interface A {}
class B {}
class C extends B implements A {}
class D implements A {}
// Declaration statements:
B b = new B();
C c = new C();
D d = new D();
```

Select the one correct answer.

(a) c = d;

(b) d = c;

(c) A a = d;

(d) d = (D) c;

(e) c = b;

6. When run, the following program will print all the letters I, J, C, and D. True or false?

```
public class MyClass {
    public static void main(String[] args) {
        I x = new D();
        if (x instanceof I) System.out.println("I");
        if (x instanceof J) System.out.println("J");
        if (x instanceof C) System.out.println("C");
        if (x instanceof D) System.out.println("D");
    }
}
interface I{}
interface J{}
class C implements I {}
class D extends C implements J {}
```

Select the one correct answer.

(a) True.

(b) False.

7. What will be the result of compiling and running the following program?

```
public class Polymorphism2 {
    public static void main(String[] args) {
        A ref1 = new C();
        B ref2 = (B) ref1;
        System.out.println(ref2.g());
    }
}
class A {
    private int f() { return 0; }
    public int g() { return 3; }
}
class B extends A {
    private int f() { return 1; }
    public int g() { return f(); }
}
class C extends B {
    public int f() { return 2; }
}
```

Select the one correct answer.

(a) The program will fail to compile.

(b) The program will compile and print 0, when run.

(c) The program will compile and print 1, when run.

(d) The program will compile and print 2, when run.

(e) The program will compile and print 3, when run.

8. Which statements about the program are true?

```
public interface HeavenlyBody { String describe(); }
class Star {
    String starName;
    public String describe() { return "star " + starName; }
}
class Planet extends Star {
    String name;
    public String describe() {
        return "planet " + name + " orbiting star " + starName;
    }
}
```

Select the two correct answers:

(a) The code will fail to compile.

(b) The code defines a Planet is-a Star relationship.

(c) The code will fail to compile if the name starName is replaced with the name bodyName throughout the declaration of the Star class.

(d) The code will fail to compile if the name starName is replaced with the name name throughout the declaration of the Star class.

(e) An instance of Planet is a valid instance of HeavenlyBody.

Answer

1. (a), (c), and (d)

Fields in interfaces declare named constants, and are always public, static, and final. None of these modifiers are mandatory in a constant declaration. All named constants must be explicitly initialized in the declaration.

2. (e)

The code will compile without errors. The class MyClass declares that it implements the interfaces Interface1 and Interface2. Since the class is declared abstract, it does not need to implement all abstract method declarations defined in these interfaces. Any non-abstract subclasses of MyClass must provide the missing method implementations. The two interfaces share a common abstract method declaration void g(). MyClass provides an implementation for this abstract method declaration that satisfies both Interface1 and Interface2. Both interfaces provide declarations of constants named VAL_B. This can lead to an ambiguity when referring to VAL_B by its simple name from MyClass. The ambiguity can be resolved by using fully qualified names: Interface1.VAL_B and Interface2.VAL_B. However, there are no problems with the code as it stands.

3. (a) and (c)

Declaration (b) fails, since it contains an illegal forward reference to its own named constant. The field type is missing in declaration (d). Declaration (e) tries illegally to use the protected modifier, even though named constants always have public accessibility. Such constants are implicitly public, static, and final.

4. (e)

Only the assignment I1 b = obj3 is valid. The assignment is allowed, since C3 extends C1, which implements I1. The assignment obj2 = obj1 is not legal, since C1 is not a subclass of C2. The assignments obj3 = obj1 and obj3 = obj2 are not legal, since neither C1 nor C2 is a subclass of C3. The assignment I1 a = obj2 is not legal, since C2 does not implement I1. Assignment I2 c = obj1 is not legal, since C1 does not implement I2.

5. (c)

Only A a = d is legal. The reference value in d can be assigned to a, since D implements A. The statements c = d and d = c are illegal, since there is no subtype-supertype relationship between C and D. Even though a cast is provided, the statement d = (D) c is illegal. The object referred to by c cannot possibly be of type D, since D is not a subclass of C. The statement c = b is illegal, since assigning a reference value of a reference of type B to a reference of type C requires a cast.

6. (a)

The program will print all the letters I, J, C, and D, when run. The object referred to by the reference x is of class D. Class D extends class C and class C implements interface I. This makes I, J, and C supertypes of class D. The reference value of an object of class D can be assigned to any reference of its supertypes and is, therefore, an instanceof these types.

7. (c)

The program will print 1 when run. The f() methods in A and B are private and are not accessible by the subclasses. Because of this, the subclasses cannot overload or override these methods, but simply define new methods with the same signature. The object being called is of the class C. The reference used to access the object is of the type B. Since B contains a method g(), the method call will be allowed at compile time. During execution it is determined that the object is of the class C, and dynamic method lookup will cause the overridden method g() in B to be executed. This method calls a method named f. It can be determined during compilation that this can only refer to the f() method in B, since the method is private and cannot be overridden. This method returns the value 1, which is printed.

8. (c) and (d)

The code as it stands will compile. The use of inheritance in this code does not define a Planet has-a Star relationship. The code will fail if the name of the field starName is changed in the Star class, since the subclass Planet tries to access it using the name starName. An instance of Planet is not an instance of HeavenlyBody. Neither Planet nor Star implements HeavenlyBody.