Top 10 Java Tricky Coding Interview Questions

Author: Ramesh Fadatare

Interview Java Quiz

This post contains frequently asked Java tricky coding interview questions with answers(with explanation).

The answer and explanation of each question have been given at end of this post.

YouTube Video

Q1 - Consider the following program and predict the output:

```java
public class Test {
    public void print(Integer i) {
        System.out.println("Integer");
    }

    public void print(int i) {
        System.out.println("int");
    }

    public void print(long i) {
        System.out.println("long");
    }

    public static void main(String args[]) {
        Test test = new Test();
        test.print(10);
    }
}
```

a) The program results in a compiler error (“ambiguous overload”).

b) long

c) Integer

d) int

Q2 - What is the output of the following program?

```
public class Test {

    public static void main(String[] args) {
        String s1 = "hello";
        String s2 = new String("hello");

        s2 = s2.intern();
        System.out.println(s1 == s2);
    }
}
```

a) false

b) true

c) None

Q3 - What will be the output of the following program?

```
class Base {
    public Base() {
        System.out.println("Base");
    }
}

class Derived extends Base {
    public Derived() {
        System.out.println("Derived");
    }
}

class DeriDerived extends Derived {
    public DeriDerived() {
        System.out.println("DeriDerived");
    }
}

public class Test {
    public static void main(String[] args) {
        Derived b = new DeriDerived();
    }
}
```

a)

```
Base
Derived
DeriDerived
```

b)

```
Derived
DeriDerived
```

c)

```
DeriDerived
Derived
Base
```

d)

```
DeriDerived
Derived
```

Q4 - Consider the following program:

```
public class Overloaded {
    public static void foo(Integer i) {
        System.out.println("foo(Integer)");
    }

    public static void foo(short i) {
        System.out.println("foo(short)");
    }

    public static void foo(long i) {
        System.out.println("foo(long)");
    }

    public static void foo(int... i) {
        System.out.println("foo(int ...)");
    }
    public static void main(String[] args) {
        foo(10);
    }
}
```

Which one of the following options correctly describes the output of this program?

a) foo(Integer)

b) foo(short)

c) foo(long)

d) foo(int ...)

Q5 - Look at the following code and choose the right option for the word :

```
// Shape.java
public class Shape {
    protected void display() {
        System.out.println("Display-base");
    }
}
// Circle.java
public class Circle extends Shape { <
    < access - modifier > void display() {
        System.out.println("Display-derived");
    }
}
```

a. Only `protected` can be used.

B. `public` and `protected` both can be used.

C. `public`, `protected`, and `private` can be used.

d. Only `public` can be used.

Q6 - Consider the following program:

```
public class BaseClass {
    private void foo() {
        System.out.println("In BaseClass.foo()");
    }

    void bar() {
        System.out.println("In BaseClass.bar()");
    }

    public static void main(String[] args) {
        BaseClass po = new DerivedClass();
        po.foo(); // BASE_FOO_CALL
        po.bar();
    }
}

class DerivedClass extends BaseClass {
    void foo() {
        System.out.println("In Derived.foo()");
    }

    void bar() {
        System.out.println("In Derived.bar()");
    }
}
```

Which one of the following options correctly describes the behavior of this program?

a)

```
This program results in a compiler error in the line marked with the comment
BASE_FOO_CALL.
```

b) This program prints the following:

```
In BaseClass.foo()
In BaseClass.bar()
```

c) This program prints the following:

```
In BaseClass.foo()
In Derived.bar()
```

d) This program prints the following:

```
In Derived.foo()
In Derived.bar()
```

Q7 - Consider the following program and predict the output:

```
class MyThread extends Thread {
    @Override
    public void run() {
        System.out.println("In run method; thread name is: " +
Thread.currentThread().getName());
    }
}

public class ThreadTest {

    public static void main(String args[]) {
        Thread myThread = new MyThread();
        myThread.run(); // #1
        System.out.println("In main method; thread name is: " +
Thread.currentThread().getName());
    }
}
```

a) The program results in a compiler error at statement #1.

b) The program results in a runtime exception.

c) The program prints the following:

   In run method; thread name is: main

   In main method; thread name is: main

d) The program prints:

   In the run method; the thread name is: thread-0

   In the main method; the thread name is: main

Q8 - Consider the following program and choose the correct option from the list of options:

```java
class Base {
    public void test() {
    }
}

class Base1 extends Base {
    public void test() {
         System.out.println("Base1");
    }
}

class Base2 extends Base {
    public void test() {
         System.out.println("Base2");
    }
}

class Test {
    public static void main(String[] args) {
        Base obj = new Base1();
        ((Base2) obj).test(); // CAST
    }
}
```

a) The program will print the following: Base1.

b) The program will print the following: Base2.

c) The compiler will report an error in the line marked with comment CAST.

d) The program will result in an exception (ClassCastException).

Q9 - Consider the following program:

```java
public class StrEqual {
    public static void main(String[] args) {
        String s1 = "hello";
        String s2 = new String("hello");
        String s3 = "hello";
        if (s1 == s2) {
            System.out.println("s1 and s2 equal");
        } else {
            System.out.println("s1 and s2 not equal");
        }
        if (s1 == s3) {
            System.out.println("s1 and s3 equal");
        } else {
            System.out.println("s1 and s3 not equal");
        }
    }
}
```

Which one of the following options provides the output of this program when executed?

a)

```
s1 and s2 equal
s1 and s3 equal
```

b)

```
s1 and s2 equal
s1 and s3 not equal
```

c)

```
s1 and s2 not equal
s1 and s3 equal
```

d)

```
s1 and s2 not equal
s1 and s3 not equal
```

Q10 - Consider the following program and predict the output:

```
public class Test {
    public static void main(String[] args) {
        String s = new String("5");
        System.out.println(1 + 10 + s + 1 + 10);
    }
}
```

a) 11511

b) 1105110

c) 115110

d) 27

Q1

Answer:

```
d) int
```

Explanation: If Integer and long types are specified, a literal will match to *int*. So, the program prints *int*.

Q2

Answer:

```
b) true
```

Explanation: We know that the `intern()` method will return the String object reference from the string pool since we assign it back to **s2** and now both **s1** and **s2** are having the same reference. It means that **s1** and **s2** references point to the same object.

Q3

Answer :

a)

```
Base
Derived
DeriDerived
```

Explanation: Whenever a class gets instantiated, the constructor of its base classes (the constructor of the root of the hierarchy gets executed first) gets invoked before the constructor of the instantiated class.

Q4

Answer:

```
c) foo(long)
```

Explanation: For an integer literal, the JVM matches in the following order: `int`, `long`, `Integer`, `int...`. In other words, it first looks for an `int` type parameter; if it is not provided, then it looks for `long` type; and so on. Here, since the `int` type parameter is not specified with an overloaded method, it matches with `foo(long)`.

Q5

Answer:

```
B. public and protected both can be used.
```

(You can provide only a less restrictive or same-access modifier when overriding a method.)

Q6

c)

```
In BaseClass.foo()
In Derived.bar()
```

Explanation: The foo() method in *BaseCase* is a private method and we can't override the private method in the *DerivedClass* subclass so JVM will call only overridden methods in subclass at runtime that is why the foo() method in *DerivedClass* is not an overridden method so JVM will call *BaseClass* foo() method. If you remove the private access modifier of the foo() method in *BaseClass* then it will invoke the *DerivedClass* foo() method because it is being overridden in the *DerivedClass* subclass.
Q7

Answer:

```
c) The program prints the following:
In run method; thread name is: main
In main method; thread name is: main
```

Explanation: The correct way to invoke a thread is to call the `start()` method on a `Thread` object. If you directly call the `run()` method, the method will run just like any other method (in other words, it will execute sequentially in the same thread without running as a separate thread).

Q8

Answer:

```
d) The program will result in an exception (ClassCastException).
```

Explanation: The dynamic type of variable obj is Base1 that you were trying to cast into Base2. This is not supported and so results in an exception.

Q9

Answer:

c)

```
s1 and s2 not equal
s1 and s3 equal
```

Explanation: JVM sets a constant pool in which it stores all the string constants used in the type. If two references are declared with a constant, then both refer to the same constant object. The `==` operator checks the similarity of objects themselves (and not the values in it). Here, the first comparison is between two distinct objects, so we get **s1** and **s2** not equal. On the other hand, since references to **s1** and **s3** refer to the same object, we get **s1** and **s3** equal.

Q10

Answer:

```
c) 115110
```

Explanation: The string concatenation operator works as follows: if both the operands are numbers, it performs the addition; otherwise it concats the arguments by calling the toString() method if needed. It evaluates from left to right. Hence, the expression in the program results in the string **115110**.