**Java OOPS (Object Oriented Programming) Tricky Coding Interview Questions**

# Q1 - What is the output of the following application?

class Automobile {

private String drive() {

return "Driving vehicle";

}

}

class Car extends Automobile {

protected String drive() {

return "Driving car";

}

}

public class ElectricCar extends Car {

@Override

public final String drive() {

return "Driving electric car";

}

public static void main(String[] wheels) {

final Car car = new ElectricCar();

System.out.print(car.drive());

}

}

A. Driving vehicle

B. Driving electric car

C. Driving car

D. The code does not compile

# Q2 - 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.

# 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:

class Base {

public Base() {

System.out.print("Base ");

}

public Base(String s) {

System.out.print("Base: " + s);

}

}

class Derived extends Base {

public Derived(String s) {

super(); // Stmt-1

super(s); // Stmt-2

System.out.print("Derived ");

}

}

class Test {

public static void main(String[] args) {

Base base = new Derived("Hello ");

}

}

**Select three correct options from the following list:**

a) Removing Stmt-1 will make the program compilable and it will print the following: Base Derived.

b) Removing Stmt-1 will make the program compilable and it will print the following: Base: Hello Derived.

c) Removing Stmt-2 will make the program compilable and it will print the following: Base Derived.

d) Removing both Stmt-1 and Stmt-2 will make the program compilable and it will print the following: Base Derived.

e) Removing both Stmt-1 and Stmt-2 will make the program compilable and it will print the following: Base: Hello Derived.

# Q5 - What is the output of the following application?

abstract class Car {

static {

System.out.print("1");

}

public Car(String name) {

super();

System.out.print("2");

}

{

System.out.print("3");

}

}

public class BlueCar extends Car {

{

System.out.print("4");

}

public BlueCar() {

super("blue");

System.out.print("5");

}

public static void main(String[] gears) {

new BlueCar();

}

}

A. 23451

B. 12354

C. 13245

D. The code does not compile.

# Q6 - What is the output of the following application?

class Math {

public final double secret = 2;

}

class ComplexMath extends Math {

public final double secret = 4;

}

public class InfiniteMath extends ComplexMath {

public final double secret = 8;

public static void main(String[] numbers) {

Math math = new InfiniteMath();

System.out.print(math.secret);

}

}

A. 2

B. 4

C. 8

D. The code does not compile.

# Q7 - Consider the following program and predict the output:

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

# Answers

## Q1

**Answer :**

B. Driving electric car

**Explanation:** The *drive()* method in the **Car** class does not override the version in the **Automobile** class since the method is not visible to the **Car** class.

The *drive()* method in the **ElectricCar** class is a valid override of the method in the **Car** class, with the access modifier expanding in the subclass. For these reasons, the code compiles, and Option D is incorrect.

In the *main()* method, the object created is an **ElectricCar**, even if it is assigned to a **Car** reference. Due to polymorphism, the method from the **ElectricCar** will be invoked, making **Option B** the correct answer.

## Q2

**Answer :**

B. public and protected both can be used.

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

## 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:**

b) Removing Stmt-1 will make the program compilable and it will print the following:

Base: Hello Derived.

c) Removing Stmt-2 will make the program compilable and it will print the following:

Base Derived.

d) Removing both Stmt-1 and Stmt-2 will make the program compilable and it will print

the following: Base Derived.

**Explanation:**If you remove Stmt-1, a call to super(s) will result in printing Base: Hello, and then the constructor of the Derived class invocation will print Derived. Similarly, the removal of Stmt-2 will also produce the correct program. In fact, if you remove both these statements, you will also get a compilable program.

## Q5

**Answer:**

C

**Explanation:** The class is loaded first, with the static initialization block called and 1 is outputted first. When the **BlueCar** is created in the *main()* method, the superclass initialization happens first. The instance initialization blocks are executed before the constructor, so 32 is outputted next. Finally, the class is loaded with the instance initialization blocks again being called before the constructor, outputting 45. The result is that 13245 is printed, making Option C the correct answer.

## Q6

**Answer:**

A

**Explanation:** The code compiles without issue, so Option D is incorrect. Java allows methods to be overridden, but not variables. Therefore, marking them *final* does not prevent them from being reimplemented in a subclass. Furthermore, polymorphism does not apply in the same way it would to methods as it does to variables. In particular, the reference type determines the version of the secret variable that is selected, making output 2 and Option A the correct answer.

## Q7

**Answer:**

d)int

**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; then it looks for long type; and so on. Here, since the int type parameter is specified with an overloaded method, it matches with int.