# **Correct output in highlight with RED color**

# **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