PROGRAMMING IN JAVA

Assignment 3

TYPE OF QUESTION: MCQ

Number of questions: 10 Total marks**: 10 × 1 = 10**

QUESTION 1:

Which of the following statement is true regarding the order of execution of constructors in an inheritance hierarchy?

a. **Base class constructor will be called followed by the derived class constructor.**
b. **Derived class constructor will be called followed by the base class constructor.**
c. **Only Base class constructor will be called.**
d. **Only derived class constructor will be called.**

Correct Answer:

a. **Base class constructor will be called followed by the derived class constructor.**

Detailed Solution:

On object creation of derived class, first base class constructor and then the derived class constructor will be called.

QUESTION 2:

The super() method is used to:

a. **Call constructor of friend class**
b. **Is a declared method**
c. **Call constructor of the parent class**
d. **Call constructor**

Correct Answer:

c. **Call constructor of the parent class**

Detailed Solution:

In Java programming language, the super() is a reference variable that is used to refer parent class constructors. The super can be used to call parent class's variables and methods. The super() can be used to call parent class' constructors only.

QUESTION 3:

What will be the output of the following Java program?

```
class A {

  int i;

  void display() {
    System.out.println(i);
  }
}

class B extends A {

  int j;

  void display() {
    System.out.println(j);
  }
}

class inheritance_demo {

  public static void main(String args[]) {
    B obj = new B();
    obj.i = 1;
    obj.j = 2;
    obj.display();
  }
}
```

a. **0**
b. **1**
c. **2**
d. **Compilation Error**

Correct Answer:

c. **2**

Detailed Solution:

Class A & class B both contain display() method, class B inherits class A, when display() method is called by object of class B, display() method of class B is executed rather than that of Class A.

QUESTION 4:

In Java, is it possible to override a static method?

a. **Yes, we can override a static method just like we do with instance methods.**
b. **No, static methods cannot be overridden because they belong to the class, not the object.**
c. **It depends on whether the static method is declared as final or not.**
d. **It depends on the access modifier of the static method.**

Correct Answer:

b. **No, static methods cannot be overridden because they belong to the class, not the object.**

Detailed Solution:

In Java, a static method is bound to the class and not to the instance. Hence, it is not part of the state of the object and doesn't participate in polymorphism and dynamic dispatch, which are necessary for method overriding.

QUESTION 5:

What is the output of the following Java program?

```
public class Vehicle {
    public void move() {
        System.out.println("The vehicle moves");
    }
}

public class Car extends Vehicle {
    public void move() {
        System.out.println("The car moves");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle vehicle = new Car();
        vehicle.move();
    }
}
```

a. **"The vehicle moves"**
b. **"The car moves"**
c. **The code does not compile**
d. **None of the above**

Correct Answer:

b. **"The car moves"**

Detailed Solution:

In Java, a subclass can override methods from its superclass. In this example, the Car class is overriding the move method of the Vehicle class. Since the object is instantiated as a Car, the overridden move method in the Car class is called, producing the output "The car moves"

QUESTION 6:

What is the output of the below Java program with inheritance?

```
class Sweet {

  void price() {
    System.out.print("Sweet=$10 ");
  }
}

class Sugar extends Sweet {

  void price() {
    super.price();
    System.out.print("Sugar=$20");
  }
}

public class JavaInheritance1 {

  public static void main(String[] args) {
    Sugar su = new Sugar();
    su.price();
  }
}
```

a. **Sweet=$10 Sugar=$20**
b. **Sweet=$10 Sugar=$10**
c. **Sweet=$20 Sugar=$20**
d. **Compiler error**

Correct Answer:

a. **Sweet=$10 Sugar=$20**

Detailed Solution:

Notice the use of the keyword "super". Using this keyword, you can call superclass's methods and variables.

QUESTION 7:

What is the purpose of method hiding in Java inheritance?

a. **To prevent a subclass from inheriting methods**
b. **To override superclass methods with new implementations**
c. **To expose private methods of the superclass**
d. **To define methods with the same name in both the superclass and subclass**

Correct Answer:

d. **To define methods with the same name in both the superclass and subclass**

Detailed Solution:

Method hiding occurs when a subclass defines a static method with the same name as a static method in the superclass.

QUESTION 8:

What is the output of the following Java program?

```
class Parent {

  String name = "parent";

  String message() {
    return "from parent";
  }
}

class Child extends Parent {

  String name = "child";

  String message() {
    return "from child";
  }
}

public class Main {

  public static void main(String[] args) {
    Parent p = new Child();
    System.out.println(p.name + " " + p.message());
  }
}
```

a. **"parent from parent"**
b. **"child from child"**
c. **"parent from child"**
d. **"child from parent"**

Correct Answer:

c. **"parent from child"**

Detailed Solution:

In Java, while methods are overridden (dynamic binding), variables are not overridden (static binding). Therefore, p.name refers to the Parent class variable, and p.message() refers to the Child class method.

QUESTION 9:

Can a class be marked as both “final” and “abstract” in Java?

a. **Yes, but only if it has no methods.**
b. **Yes, a class can be marked as both “final” and “abstract.”**
c. **No, a class cannot be both “final” and “abstract.”**
d. **Yes, but only if it is marked as “protected.”**

Correct Answer:

c. **No, a class cannot be both “final” and “abstract.”**

Detailed Solution:

A class marked as “final” cannot be extended (subclassed), while an abstract class is meant to be extended. Therefore, they are contradictory modifiers.

QUESTION 10:

In Java, is it possible to override a static method?

a. **Yes, we can override a static method just like we do with instance methods.**
b. **No, static methods cannot be overridden because they belong to the class, not the object.**
c. **It depends on whether the static method is declared as final or not.**
d. **It depends on the access modifier of the static method.**

Correct Answer:

b. **No, static methods cannot be overridden because they belong to the class, not the object.**

Detailed Solution:

In Java, a static method is bound to the class and not to the instance. Hence, it is not part of the state of the object and doesn't participate in polymorphism and dynamic dispatch, which are necessary for method overriding.