Southern University Bangladesh

Department of Computer Science and Engineering

Faculty of Science and Engineering

Object Oriented Programming

Course code: CS-3011/ MS 5202

Assignment :01

Submitted by:

Name : Sharmistha Chowdhury

ID: 666-61-60

Submitted to :

Mohammed Arif Hasan Chowdhury, Assistant Professor

Dept. of Computer Science and Engineering

Date : 21/05/2024

Questions

1. What is method overriding? Write the rules for method overriding and Give an example with code
2. What is inheritance in Java? Develop a code segment which demonstrates the Java inheritance. You can observe two classes namely Calculation and My Calculation as super and sub class respectively.
3. When Super keyword is used? Apply super keyword by using an appropriate example.
4. Describe abstract class with an example and why abstract class is used?
5. Explain with a proper example how parameterized constructor can be used to initialize value of a program

Answers

1. **Method Overriding in Java**

   Method overriding is a feature in Java that allows a subclass to provide a specific implementation of a method that is already provided by its parent class. The method in the subclass must have the same name, return type, and parameters as the method in the parent class.

   Rules for Method Overriding:

   - The method must have the same name as in the parent class.
   - The method must have the same parameter as in the parent class.
   - There must be an IS-A relationship (inheritance).

   Example:

```
class Animal {
   void eat() {
      System.out.println("eating...");
   }
}

class Dog extends Animal {
   void eat() {
      System.out.println("eating bread...");
   }
}

class Test {
   public static void main(String args[]) {
      Dog d = new Dog();
      d.eat();
   }
}
```

2. **Inheritance in Java**

Inheritance in Java is a mechanism where one class acquires the properties (fields) and behaviors (methods) of another class. The class which inherits the properties of another class is known as the subclass (or derived class, or child class), and the class whose properties are inherited is known as the superclass (or base class, or parent class).

Example:

```
class Calculation {
   int z;

   public void addition(int x, int y) {
      z = x + y;
      System.out.println("The sum is: " + z);
   }

   public void subtraction(int x, int y) {
      z = x - y;
      System.out.println("The difference is: " + z);
   }
}

public class MyCalculation extends Calculation {
   public void multiplication(int x, int y) {
      z = x * y;
      System.out.println("The product is: " + z);
   }

   public static void main(String args[]) {
      int a = 20, b = 10;
      MyCalculation demo = new MyCalculation();
      demo.addition(a, b);
      demo.subtraction(a, b);
      demo.multiplication(a, b);
   }
}
```

3. **Super Keyword in Java**

The super keyword in Java is a reference variable that is used to refer to the immediate parent class object. It can be used to invoke the immediate parent class method.

Example:

```
class Animal {
   void eat() {
      System.out.println("eating...");
   }
}

class Dog extends Animal {
   void eat() {
      System.out.println("eating bread...");
   }

   void bark() {
      System.out.println("barking...");
   }

   void work() {
      super.eat();
      bark();
   }
}

class Test {
   public static void main(String args[]) {
      Dog d = new Dog();
      d.work();
   }
}
```

4. **Abstract Class in Java**

An abstract class in Java is a class that cannot be instantiated (you cannot create objects of an abstract class). It is used to provide a base for subclasses to extend and implement the abstract methods and override or use the implemented methods in the abstract class.

Example:

```
abstract class Animal {
   abstract void makeSound();

   public void eat() {
      System.out.println("I can eat.");
   }
}

class Dog extends Animal {
   public void makeSound() {
      System.out.println("Bark bark.");
   }
}

class Main {
   public static void main(String[] args) {
      Dog d1 = new Dog();
      d1.makeSound();
      d1.eat();
   }
}
```

5. **Parameterized Constructor in Java**

   A parameterized constructor is a constructor that accepts arguments. We can provide different values to the distinct objects by using a parameterized constructor.

   Example:

```
class Student {
   int id;
   String name;

   Student(int i, String n) {
      id = i;
      name = n;
   }

   void display() {
      System.out.println(id + " " + name);
   }

   public static void main(String args[]) {
      Student s1 = new Student(111, "John");
      Student s2 = new Student(222, "Mike");
      s1.display();
      s2.display();
   }
}
```

This code demonstrates the use of a parameterized constructor to initialize the id and name fields of the Student class. When we create a Student object, we pass the id and name values to the constructor, and these values are then used to initialize the fields of the object. This allows us to create multiple Student objects with different id and name values. The display() method is used to print the id and name of each Student object.