Inheritance and Packages

1. Write a java program to use Interface in java.

2. Write a java program to extend one interface into another interface.

3. Write a java program to perform simple inheritance.

4. Write a java program to use multilevel inheritance.

5. Write a java program to use Hierarchical inheritance.

6. Write a java program to use Abstract class.

7. Write a java program to use interface.

8. Write a java program to use Multiple inheritance using interface**.**

9. Write a java program to perform overriding of abstract class

10. Write a java program to perform overriding of abstract class.

11. Write a java program to demonstrate encapsulation.

Write a java program to use Interface in java.

Code:

```
// Define an interface
interface Shape {
    double calculateArea();
    double calculatePerimeter();
}

// Implement the interface in classes
class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}

class Rectangle implements Shape {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    @Override
    public double calculateArea() {
        return width * height;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * (width + height);
    }
```

```
}

public class InterfaceExample {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(4, 6);

        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

        System.out.println("Rectangle Area: " + rectangle.calculateArea());
        System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());
    }
}
```

Output:

```
        Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Rectangle Area: 24.0
Rectangle Perimeter: 20.0
```

Write a java program to extend one interface into another interface.

Code:

```
        // Parent interface
interface Shape {
   double calculateArea();
   double calculatePerimeter();
}

// Child interface extending the parent interface
interface ThreeDimensionalShape extends Shape {
   double calculateVolume();
}

// Class implementing the child interface
class Sphere implements ThreeDimensionalShape {
   private double radius;

   public Sphere(double radius) {
      this.radius = radius;
   }
   @Override
   public double calculateArea() {
      return 4 * Math.PI * radius * radius;
   }

   @Override
   public double calculatePerimeter() {
      return 0; // Not applicable for a sphere
   }

   @Override
   public double calculateVolume() {
      return (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
   }
}

public class InterfaceExtensionExample {
   public static void main(String[] args) {
      Sphere sphere = new Sphere(3);

      System.out.println("Sphere Area: " + sphere.calculateArea());
      System.out.println("Sphere Volume: " + sphere.calculateVolume());
   }
}
```

Output:

Sphere Area: 113.09733552923255

Sphere Volume: 113.09733552923254

Write a java program to perform simple inheritance

Code:

```
// Parent class
class Vehicle {
    void start() {
        System.out.println("The vehicle is starting.");
    }
}

// Child class inheriting from Vehicle
class Car extends Vehicle {
    void accelerate() {
        System.out.println("The car is accelerating.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.start();     // Inherited method from parent class
        myCar.accelerate(); // Method from child class
    }
}
```

Write a java program to use multilevel inheritance

Code:

```
        // Grandparent class
class Animal {
   void eat() {
      System.out.println("The  animal is eating.");
   }
}

// Parent class inheriting from Animal
class Dog extends Animal {
   void bark() {
      System.out.println("The  dog is barking.");
   }
}

// Child class inheriting from Dog
class Bulldog extends Dog {
   void guard() {
      System.out.println("The  bulldog is guarding.");
   }
}

public class Main {
   public static void main(String[] args) {
      Bulldog myBulldog = new Bulldog();
      myBulldog.eat();    // Inherited method from grandparent class
      myBulldog.bark();   // Inherited method from parent class
      myBulldog.guard();  // Method from child class
   }
}
```

Write a java program to use Hierarchical inheritance

Code:

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

class Dog extends Animal {
   void bark() {
      System.out.println("Dog  is barking.");
   }
}

class Cat extends Animal {
   void meow() {
      System.out.println("Cat  is meowing.");
   }
}

public class HierarchicalInheritanceExample {
   public static void main(String[] args) {
      Dog dog = new Dog();
      Cat cat = new Cat();

      dog.eat();
      dog.bark();
```

```
        cat.eat();
        cat.meow();
    }
}
```

Output:

```
Animal is eating.
Dog is barking.
Animal is eating.
Cat is meowing.
```

Write a java program to use Abstract class

Code:

```
        // Abstract class
abstract class Shape {
   abstract double calculateArea();
   abstract double calculatePerimeter();
}

// Concrete class extending the abstract class
class Circle extends Shape {
   private double radius;

   public Circle(double radius) {
      this.radius = radius;
   }

   @Override
   double calculateArea() {
      return Math.PI * radius * radius;
   }

   @Override
   double calculatePerimeter() {
      return 2 * Math.PI * radius;
   }
}

// Concrete class extending the abstract class
class Rectangle extends Shape {
   private double length;
   private double width;

   public Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }

   @Override
   double calculateArea() {
      return length * width;
   }
```

```
    @Override
    double calculatePerimeter() {
        return 2 * (length + width);
    }
}

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5.0);
        System.out.println("Circle  Area: " + circle.calculateArea());
        System.out.println("Circle  Perimeter:  " + circle.calculatePerimeter());

        Rectangle rectangle = new Rectangle(4.0, 6.0);
        System.out.println("Rectangle  Area: " + rectangle.calculateArea());
        System.out.println("Rectangle  Perimeter:  " + rectangle.calculatePerimeter());
    }
}
```

Write a java program to use interface.

Code:

```
        // Define an interface
interface Shape {
   double calculateArea();
}

// Implement the interface in a class
class Circle implements Shape {
   double radius;

   public Circle(double radius) {
      this.radius = radius;
   }

   @Override
   public double calculateArea() {
      return Math.PI * radius * radius;
   }
}

class Rectangle implements Shape {
   double width, height;

   public Rectangle(double width, double height) {
      this.width = width;
      this.height = height;
   }

   @Override
   public double calculateArea() {
      return width * height;
   }
}

public class Main {
   public static void main(String[] args) {
      Circle circle = new Circle(5.0);
      Rectangle rectangle = new Rectangle(4.0, 6.0);
```

```
        System.out.println("Circle  Area: " + circle.calculateArea());
        System.out.println("Rectangle  Area: " + rectangle.calculateArea());
    }
}
```

Write a java program to use Multiple inheritance using interface.

Code:

```
        // Define interfaces
interface A {
  void methodA();
}

interface B {
  void methodB();
}

// Implement the interfaces in a class
class MultipleInheritance implements A, B {
  @Override
  public void methodA() {
    System.out.println("Method A");
  }

  @Override
  public void methodB() {
    System.out.println("Method B");
  }
}

public class Main {
  public static void main(String[] args) {
    MultipleInheritance obj = new MultipleInheritance();
    obj.methodA();
    obj.methodB();
  }
}
```

Output:

```
Method A
Method B
```

Write a java program to perform overriding of abstract class.

Code:

```
        // Base class
class Animal {
   void makeSound() {
      System.out.println("Animal  makes a sound");
   }
}

// Subclass that overrides the method
class Dog extends Animal {
   @Override
   void makeSound() {
      System.out.println("Dog  barks");
   }
}

public class Main {
   public static void main(String[] args) {
      Animal animal = new Animal();
      Dog dog = new Dog();

      animal.makeSound(); // Output: Animal makes a sound
      dog.makeSound();    // Output: Dog barks
   }
}
```

Output:

```
        Animal makes a sound
Dog barks
```

Here's an example of method overriding in an abstract class:

Code:

```
// Abstract base class
abstract class Shape {
   abstract double calculateArea();
}

// Concrete subclass that overrides the abstract method
class Circle extends Shape {
   double radius;

   public Circle(double radius) {
      this.radius = radius;
   }

   @Override
   double calculateArea() {
      return Math.PI * radius * radius;
   }
}

public class Main {
   public static void main(String[] args) {
      Circle circle = new Circle(5.0);
      System.out.println("Circle Area: " + circle.calculateArea());
   }
}
```

Output:

```
Circle Area: 78.53981633974483
```

Write a java program to demonstrate encapsulation

Code:

```
class Person {
private String name;
private int age;

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getAge() { return age; }

public void setAge(int age) { this.age = age; }
}

public class Main {
public static void main(String[] args)
{
Person person = new Person();
person.setName("John");
person.setAge(30);

System.out.println("Name: " + person.getName());
System.out.println("Age: " + person.getAge());
}
}
```

Output:

```
Name: John
Age: 30
```