**Lab 10, Packages and Interfaces**

**Name: Shahzaman Abbasi**

**CMS ID: 023-20-0122**

**Exercise 1:**

**Code:**

package pkg1;

public class ClassOne {

    public void methodOne() {

        System.out.println("Method One from ClassOne in PackageOne");

    }

}

public class ClassTwo {

    public void methodTwo() {

        System.out.println("Method Two from ClassTwo in PackageOne");

    }

}

package pkg2;

public class ClassThree {

    public void methodThree() {

        System.out.println("Method Three from ClassThree in PackageTwo");

    }

}

public class ClassFour {

    public void methodFour() {

        System.out.println("Method Four from ClassFour in PackageTwo");

    }

}

package pkg3;

public class ClassFive {

    public void methodFive() {

        System.out.println("Method Five from ClassFive in PackageThree");

    }

}

public class ClassSix {

    public void methodSix() {

        System.out.println("Method Six from ClassSix in PackageThree");

    }

}

import pkg1.\*;

import pkg2.\*;

import pkg3.\*;

public class PackagePractice {

    public static void main(String[] *args*) {

        ClassOne one = new ClassOne();

        ClassTwo two = new ClassTwo();

        ClassThree three = new ClassThree();

        ClassFour four = new ClassFour();

        ClassFive five = new ClassFive();

        ClassSix six = new ClassSix();

        one.methodOne();

        two.methodTwo();

        three.methodThree();

        four.methodFour();

        five.methodFive();

        six.methodSix();

    }

}

**Question: 2 (Interfaces)**

What is wrong with the following interface?

public interface SomethingIsWrong {

void aMethod(int aValue){

System.out.println("Hi Mom");

}

}

1. Fix the interface in question.

2. Is the following interface valid?

public interface Marker {

}

**Solution:**

1. **Interfaces cannot have method bodies.**

**Corrected Version:**

**public interface SomethingIsWrong {**

**void aMethod(int aValue);**

**}**

1. **Yes it’s valid.**

**Exercise 3:**

**Code:**

interface Animal {

    void legs();

    void eat();

}

class Spider implements Animal {

    @Override

    public void legs() {

        System.out.println("A spider has 8 legs.");

    }

    @Override

    public void eat() {

        System.out.println("A spider eats insects.");

    }

}

class Caterpillar implements Animal {

    @Override

    public void legs() {

        System.out.println("A caterpillar has many legs.");

    }

    @Override

    public void eat() {

        System.out.println("A caterpillar eats leaves.");

    }

}

class Cat implements Animal {

    @Override

    public void legs() {

        System.out.println("A cat has 4 legs.");

    }

    @Override

    public void eat() {

        System.out.println("A cat eats fish and meat.");

    }

}

 public class Exercise3 {

    public static void main(String[] *args*) {

        Spider spider = new Spider();

        spider.legs();

        spider.eat();

        Caterpillar caterpillar = new Caterpillar();

        caterpillar.legs();

        caterpillar.eat();

        Cat cat = new Cat();

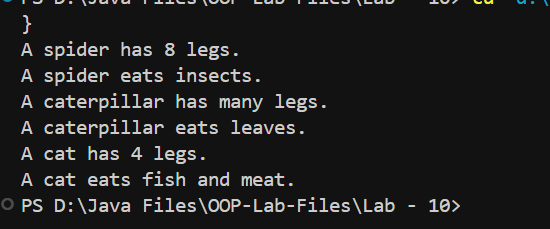
        cat.legs();

        cat.eat();

    }

}

**Output:**

****

**Exercise 4:**

**Code:**

interface Employee {

    void details();

}

interface Officer {

    void info();

}

class Person implements Employee, Officer {

    public void details() {

        System.out.println("Employee details");

    }

    public void info() {

        System.out.println("Officer info");

    }

}

public class Exercise4 {

    public static void main(String[] *args*) {

        Person person = new Person();

        person.details();

        person.info();

    }

**Exercise 5:**

**Code:**

interface Tossable {

    public String toss();

  }

  class Ball implements Tossable {

    private String brandName;

    public Ball(String *brandName*) {

      this.brandName = brandName;

    }

    public String getBrandName() {

      return brandName;

    }

    public String bounce() {

      return "The ball bounces";

    }

    @Override

    public String toss() {

      return "The ball is tossed";

    }

  }

  class Baseball extends Ball {

    public Baseball(String *brandName*) {

      super(brandName);

    }

    @Override

    public String toss() {

      return "The baseball is tossed";

    }

    @Override

    public String bounce() {

      return "The baseball bounces on the ground";

    }

  }

  class Football extends Ball {

    public Football(String *brandName*) {

      super(brandName);

    }

    @Override

    public String toss() {

      return "The football is tossed in the air";

    }

    @Override

    public String bounce() {

      return "The football bounces erratically";

    }

  }

**Exercise 6:**

**Code:**

*//Exercise 6*

interface Printable {

    void print();

}

class Rectangle implements Printable {

    int length, width;

    Rectangle(int *length*, int *width*) {

        this.length = length;

        this.width = width;

    }

    public void print() {

        System.out.println("Rectangle: length = " + length + " width = " + width);

    }

}

class SportCar implements Printable {

    String name;

    int number;

    SportCar(String *name*, int *number*) {

        this.name = name;

        this.number = number;

    }

    public void print() {

        System.out.println("SportCar: name = " + name + " number = " + number);

    }

}

class Manager implements Printable {

    String name;

    int age;

    Manager(String *name*, int *age*) {

        this.name = name;

        this.age = age;

    }

    public void print() {

        System.out.println("Manager: name = " + name + " age = " + age);

    }

}

public class PrintableDemo {

    public static void main(String[] *args*) {

        Printable vec[] = {

                new Rectangle(110, 80),

                new SportCar("Toyota", 989621),

                new Rectangle(34, 32),

                new Manager("John", 40),

                new Rectangle(54, 10),

                new SportCar("Audi", 2365644),

                new SportCar("Mazda", 4322343),

                new Manager("Joji", 22)

        };

        for (int index = 0; index < vec.length; index++) {

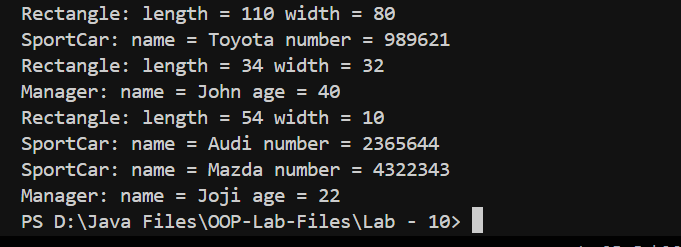
            vec[index].print();

        }

    }

}

**Output:**

****