**INSTITUTE OF ENGINEERING & MANAGEMENT**

**Department of Computer Science & Engineering**



|  |  |
| --- | --- |
| **Name** | **: Saptarshi Mondal** |
| **Class Roll** | **: 27** |
| **Enrollment No.** | **: 12019002002039** |
| **Subject Name** | **: OOP Lab** |
| **Assignment No.** | **: Day 7** |
| **Date** | **: 11/08/2021** |

# Design an interface named Figure with a field PI (=3.14). Create two concrete subclasses of it named Circle and Rectangle respectively. Create objects of the two subclasses and calculate their areas.

# Ans:

import *java*.*util*.*\**;

*interface* Figure {

    double PI = 3.14;

    double area();

}

*class* Circle *implements* Figure {

    double rad;

    Circle(double rad) {

        this.*rad* = rad;

    }

*public* double area() {

        return Figure.*PI* \* rad \* rad;

    }

}

*class* Rectangle *implements* Figure {

    double length, breadth;

    Rectangle(double length, double breadth) {

        this.*length* = length;

        this.*breadth* = breadth;

    }

*public* double area() {

        return length \* breadth;

    }

}

*public* *class* CalcArea {

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

        Scanner ob = new Scanner(System.*in*);

        System.*out*.print("\nEnter the radius of the circle : ");

        double r = ob.nextDouble();

        System.*out*.print("\nEnter the length and breadth of the Rectangle respectively : ");

        double l = ob.nextDouble();

        double b = ob.nextDouble();

        Figure fig;

        fig = new Circle(r);

        System.*out*.println("\nThe area of the circle is : " + fig.area());

        fig = new Rectangle(l, b);

        System.*out*.println("\nThe area of the Rectangle is : " + fig.area());

        System.*out*.println();

        ob.close();

    }

}

# Output:

# 

# 2. Check without having any abstract method whether an interface is possible. If so, then give coding example.

# Ans:

*interface* NewIntr {

*default* void disp() {

        System.*out*.println("\nDefault method in interface but not abstract method.\n");

    }

*static* int cube(int x) {

        return x \* x \* x;

    }

}

*public* *class* NoAbstractMethod *implements* NewIntr {

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

        System.*out*.println("\nCube of number: " + NewIntr.cube(5));

        NewIntr n = new NoAbstractMethod();

        n.disp();

    }

}

# Output:

# 

# 3. Create an interface with three abstract methods check whether you can override only

# few methods (not all methods) in its concrete subclass or not.

# Ans:

*interface* Intr {

    void methodOne();

    void methodTwo();

    void methodThree();

}

*abstract* *class* Abstract *implements* Intr {

    @Override

*public* void methodOne() {

        System.*out*.println("Abstract class inheriting method one from interface");

    }

    @Override

*public* void methodTwo() {

        System.*out*.println("Abstract class inheriting method two from interface");

    }

}

*class* ExtendAbstract *extends* Abstract {

    @Override

*public* void methodThree() {

        System.*out*.println("Abstract class inheriting method three from interface via Abstract Class");

    }

}

*public* *class* AbstractInherit {

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

        ExtendAbstract obj = new ExtendAbstract();

        System.*out*.println();

        obj.methodOne();

        obj.methodTwo();

        obj.methodThree();

        System.*out*.println();

    }

}

# Output:

# 

# 4. Does an interface can inherit an abstract class? If so, then give coding example.

# Ans:

*interface* A {

    void methodOne();

    void methodTwo();

    void methodThree();

}

*class* B *implements* A {

*public* void methodOne() {

        System.*out*.println("Method One of Interface.");

    }

*public* void methodTwo() {

        System.*out*.println("Method Two of Interface.");

    }

*public* void methodThree() {

        System.*out*.println("Method Three of Interface.");

    }

}

*public* *class* FewMethod {

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

        B ab = new B();

        System.*out*.println();

        ab.methodOne();

        ab.methodTwo();

        ab.methodThree();

        System.*out*.println();

    }

}

# 

# 5. Create two interfaces, each with two methods. Inherit a new interface from the two,

# adding a new method. Create a class by implementing the new interface and inheriting

# from a concrete class. In main() method, create an object of the derived class and call

# all the methods.

# Ans:

*interface* Add\_Sub {

*public* void add(double x, double y);

*public* void subtract(double x, double y);

}

*interface* Mul\_Div {

*public* void multiply(double x, double y);

*public* void divide(double x, double y);

}

*class* ExtendClass {

    void show() {

        System.*out*.println("Extended class via Execute.");

    }

}

*interface* Calculator *extends* Add\_Sub, Mul\_Div {

*public* void printResult(double result);

}

*public* *class* Execute *extends* ExtendClass *implements* Calculator {

*public* void add(double x, double y) {

        double result = x + y;

        printResult(result);

    }

*public* void subtract(double x, double y) {

        double result = x - y;

        printResult(result);

    }

*public* void multiply(double x, double y) {

        double result = x \* y;

        printResult(result);

    }

*public* void divide(double x, double y) {

        double result = x / y;

        printResult(result);

    }

*public* void printResult(double result) {

        System.*out*.println("The result is : " + result);

    }

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

        Execute c = new Execute();

        System.*out*.println();

        c.add(5, 10);

        c.subtract(35, 15);

        c.multiply(6, 9);

        c.divide(45, 6);

        c.show();

        System.*out*.println();

    }

}

# Output: