5. Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.

// Shape class (superclass)

**class** Shape {

**public** **void** draw() {

System.***out***.println("Drawing a shape");

}

**public** **void** erase() {

System.***out***.println("Erasing a shape");

}

}

// Circle class (subclass)

**class** Circle **extends** Shape {

@Override

**public** **void** draw() {

System.***out***.println("Drawing a circle");

}

@Override

**public** **void** erase() {

System.***out***.println("Erasing a circle");

}

}

// Triangle class (subclass)

**class** Triangle **extends** Shape {

@Override

**public** **void** draw() {

System.***out***.println("Drawing a triangle");

}

@Override

**public** **void** erase() {

System.***out***.println("Erasing a triangle");

}

}

// Square class (subclass)

**class** Square **extends** Shape {

@Override

**public** **void** draw() {

System.***out***.println("Drawing a square");

}

@Override

**public** **void** erase() {

System.***out***.println("Erasing a square");

}

}

**public** **class** Program5 {

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

// Creating objects of each shape

Shape circle = **new** Circle();

Shape triangle = **new** Triangle();

Shape square = **new** Square();

// Demonstrating polymorphism

// Calling draw() and erase() methods on different shapes

circle.draw();

circle.erase();

triangle.draw();

triangle.erase();

square.draw();

square.erase();

}

}

Output:

Drawing a circle

Erasing a circle

Drawing a triangle

Erasing a triangle

Drawing a square

Erasing a square

6. Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

// Abstract Shape class

**abstract** **class** Shape1 {

**abstract** **double** calculateArea();

**abstract** **double** calculatePerimeter();

}

// Circle class extending Shape

**class** Circle1 **extends** Shape1 {

**private** **double** radius;

**public** Circle1(**double** radius) {

**this**.radius = radius;

}

@Override

**double** calculateArea() {

**return** Math.***PI*** \* radius \* radius;

}

@Override

**double** calculatePerimeter() {

**return** 2 \* Math.***PI*** \* radius;

}

}

// Triangle class extending Shape

**class** Triangle1 **extends** Shape1 {

**private** **double** side1;

**private** **double** side2;

**private** **double** side3;

**public** Triangle1(**double** side1, **double** side2, **double** side3) {

**this**.side1 = side1;

**this**.side2 = side2;

**this**.side3 = side3;

}

@Override

**double** calculateArea() {

// Heron's formula to calculate area of a triangle

**double** s = (side1 + side2 + side3) / 2;

**return** Math.*sqrt*(s \* (s - side1) \* (s - side2) \* (s - side3));

}

@Override

**double** calculatePerimeter() {

**return** side1 + side2 + side3;

}

}

**public** **class** Program6 {

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

// Creating a circle and calculating its area and perimeter

Circle1 circle = **new** Circle1(5);

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

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

// Creating a triangle and calculating its area and perimeter

Triangle1 triangle = **new** Triangle1(3, 4, 5);

System.***out***.println("Triangle Area: " + triangle.calculateArea());

System.***out***.println("Triangle Perimeter: " + triangle.calculatePerimeter());

}

}

Output:

Circle Area: 78.53981633974483

Circle Perimeter: 31.41592653589793

Triangle Area: 6.0

Triangle Perimeter: 12.0

7. Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods .

// Resizable interface

**interface** Resizable {

**void** resizeWidth(**int** width);

**void** resizeHeight(**int** height);

}

// Rectangle class implementing Resizable interface

**class** Rectangle **implements** Resizable {

**private** **int** width;

**private** **int** height;

**public** Rectangle(**int** width, **int** height) {

**this**.width = width;

**this**.height = height;

}

**public** **void** displayDimensions() {

System.***out***.println("Rectangle Dimensions: Width = " + width + ", Height = " + height);

}

@Override

**public** **void** resizeWidth(**int** width) {

**this**.width = width;

}

@Override

**public** **void** resizeHeight(**int** height) {

**this**.height = height;

}

}

**public** **class** Program7 {

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

Rectangle rectangle = **new** Rectangle(5, 10);

// Displaying initial dimensions

System.***out***.println("Before resizing:");

rectangle.displayDimensions();

// Resizing the rectangle

rectangle.resizeWidth(8);

rectangle.resizeHeight(12);

// Displaying updated dimensions after resizing

System.***out***.println("\nAfter resizing:");

rectangle.displayDimensions();

}

}

Output:

Before resizing:

Rectangle Dimensions: Width = 5, Height = 10

After resizing:

Rectangle Dimensions: Width = 8, Height = 12

8. Develop a JAVA program to create an outer class with a function display. Create another class inside the outer class named inner with a function called display and call the two functions in the main class. //

//Outer class

**class** Outer {

**void** display() {

System.***out***.println("Outer display method");

}

// Inner class inside Outer class

**class** Inner {

**void** display() {

System.***out***.println("Inner display method");

}

}

}

**public** **class** Program8 {

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

Outer outer = **new** Outer();

Outer.Inner inner = outer.**new** Inner();

// Calling display methods from outer and inner classes

outer.display();

inner.display();

}

}

Output:

Outer display method

Inner display method

9. Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.

// Custom exception class for DivisionByZero

**class** DivisionByZeroException **extends** Exception {

**public** DivisionByZeroException(String message) {

**super**(message);

}

}

**public** **class** Program9 {

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

**int** dividend = 10;

**int** divisor = 0;

**try** {

**if** (divisor == 0) {

**throw** **new** DivisionByZeroException("Cannot divide by zero");

}

**int** result = dividend / divisor;

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

} **catch** (DivisionByZeroException e) {

System.***out***.println("DivisionByZeroException caught: " + e.getMessage());

} **finally** {

System.***out***.println("Finally block executed");

}

}

}

Output:

DivisionByZeroException caught: Cannot divide by zero

Finally block executed