# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**Belgaum, Karnataka-590 014**



**Object Oriented Programming with JAVA**

**(**BCS306A**)**

**B.E - III Sem**

# Compiled and Reviewed By

**Prof. Jawahar Jonathan Prof. Bharathi K**

**Assistant Professor Assistant Professor**



## Approved By:

## Dr. Ajith Padyana

## Professor and HOD

## Computer Science Engineering, AIT, Acharya, Bangalore -107

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**(ACCREDITED BY NBA)**

## ACHARYA INSTITUTE OF TECHNOLOGY

**Soldevanahalli, Bengaluru-560107 2023-2024**

## Table of contents

Vision, Mission, Motto of Institute I

Vision, Mission of Department II

Laboratory Objectives III

[Program Specific Outcomes (PSOs) III](#_bookmark1)

[Program outcomes (POs) IV](#_bookmark2)

Course outcomes (COs) VI

**ACHARYA INSTITUTE OF TECHNOLOGY**

**MOTTO**

"Nurturing Aspirations Supporting Growth" VISION “Acharya Institute of Technology, committed to the cause of sustainable value-based education in all disciplines, envisions itself as a global fountainhead of innovative human enterprise, with inspirational initiatives for Academic Excellence”.

**VISION OF THE INSTITUTE**

Acharya Institute of Technology, committed to the cause of value-based education in all disciplines, envisions itself as fountainhead of innovative human enterprise, with inspirational initiatives for Academic Excellence.

**MISSION OF INSTITUTE**

“Acharya Institute of Technology strives to provide excellent academic ambiance to the students for achieving global standards of technical education, foster intellectual and personal development, meaningful research and ethical service to sustainable societal needs.”

**VISION OF THE DEPARTMENT**

Envisions to be recognized for quality education and research in the field of Computing, leading to creation of globally competent engineers, who are innovative and adaptable to the changing demands of industry and society.

**MISSION OF THE DEPARTMENT**

Act as a nurturing ground for young computing aspirants to attain the excellence by imparting quality education. Collaborate with industries and provide exposure to latest tools/ technologies. Create an environment conducive for research and continuous learning

**LABORATORY OBJECTIVES**

This laboratory will enable students to

* To introduce Java compiler and eclipse platform.
* Learn and acquire the art Java Programming
* To make the student learn an object-oriented way of solving problems using java.
* To write programs using abstract classes
* Create packages and interfaces

Descriptions (if any):

* Installation procedure of the Java software must be demonstrated carried out in groups.
* Students should use the latest version of Eclipse to execute these programs.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

PSO-1 Students shall apply the knowledge of hardware, system software, algorithms, computer networks and data bases for real world problems.

PSO-2 Students shall design, analyze and develop efficient and secure algorithms using appropriate data structures, databases for processing of data.

PSO-3 Students shall be capable of developing stand alone, embedded and web-based solutions having easy to operate interface using software engineering practices and contemporary computer programming languages.

**PROGRAM OUTCOMES (Pos)**

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**COURSE OUTCOMES**

After the completion of the course, the students will be able to

1. Demonstrate ability in writing simple programs using class, methods and involving branching and looping statements.
2. Apply the concepts of inheritance, polymorphism and interface in solving real world problems.
3. Analyze the concept of packages and exception handling in solving complex problem.

Implement concepts of multithreading, autoboxing and enumerations in program

development.

Programming Assignments

|  |  |  |
| --- | --- | --- |
| SI.No | Name of Program | Page No |
| 1 | Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments). |  |
| 2 | Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations. |  |
| 3 | A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration. |  |
| 4 | A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:  ● Two instance variables x (int) and y (int).  ● A default (or "no-arg") constructor that construct a point at the default location of (0, 0).  ● A overloaded constructor that constructs a point with the given x and y coordinates.  ● A method setXY() to set both x and y.  ● A method getXY() which returns the x and y in a 2-element int array.  ● A toString() method that returns a string description of the instance in the format "(x, y)".  ● A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates  ● An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another)  ● Another overloaded distance() method that returns the distance from this point to the origin (0,0) Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class. |  |
| 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. |  |
| 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. |  |
| 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 |  |
| 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. |  |
| 9 | Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally. |  |
| 10 | Develop a JAVA program to create a package named mypack and import & implement it in a suitable class. |  |
| 11 | Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds). |  |
| 12 | Develop a program to create a class MyThread in this class a constructor, call the base class constructor, using super and start the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently. |  |

### JAVA TERMINOLOGY

Before learning Java, one must be familiar with these common terms of Java.

**1.  Java Virtual Machine(JVM):** This is generally referred to as [JVM](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/#:~:text=JVM(Java%20Virtual%20Machine)%20acts,(Write%20Once%20Run%20Anywhere).). There are three execution phases of a program. They are written, compile and run the program.

* Writing a program is done by a java programmer like you and me.
* The compilation is done by the **JAVAC** compiler which is a primary Java compiler included in the Java development kit (JDK). It takes the Java program as input and generates bytecode as output.
* In the Runningphase of a program,**JVM** executes the bytecode generated by the compiler.

Now, we understood that the function of Java Virtual Machine is to execute the bytecode produced by the compiler. Every Operating System has a different JVM but the output they produce after the execution of bytecode is the same across all the operating systems. This is why Java is known as a**platform-independent language.**

**2. Bytecode in**the **Development Process:**As discussed, the Javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM. It is saved as **.class** file by the compiler. To view the bytecode, a disassembler like [javap](https://www.geeksforgeeks.org/javap-tool-in-java-with-examples/) can be used.

**3. Java Development Kit(JDK):**While we were using the term JDK when we learn about bytecode and JVM. So, as the name suggests, it is a complete Java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs, etc. For the program to execute in java, we need to install JDK on our computer in order to create, compile and run the java program.

**4. Java Runtime Environment (JRE):**JDK includes JRE. JRE installation on our computers allows the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet support, and plugins. For running the java program, a computer needs JRE.

**5. Garbage Collector:**In Java, programmers can’t delete the objects. To delete or recollect that memory JVM has a program called [Garbage Collector](https://www.geeksforgeeks.org/garbage-collection-java/). Garbage Collectors can recollect the objects that are not referenced. So Java makes the life of a programmer easy by handling memory management. However, programmers should be careful about their code whether they are using objects that have been used for a long time. Because Garbage cannot recover the memory of objects being referenced.

**6. ClassPath:**The [classpath](https://www.geeksforgeeks.org/classpath-in-java/) is the file path where the java runtime and Java compiler look for **.class** files to load. By default, JDK provides many libraries. If you want to include external libraries, they should be added to the classpath.

#### LABORATORY PROGRAMS

1. Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).

**import** java.util.Scanner;

**public** **class** Program1

{

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

{

System.***out***.println("Set Commanad line argument by: Run->Run configuration->arguments");

**int** N = Integer.*parseInt*(args[0]);

System.***out***.println("The order of the matrix is:"+N );

System.***out***.println("Read first matrix NXN elements");

**if** (N <= 0)

{

System.***out***.println("N should be a positive integer.");

**return**;

}

**int**[][] matrixA = *read*(N);

**int**[][] matrixB = *generateRandomMatrix*(N);//generateRandomMatrix(N);

System.***out***.println("Matrix A:");

*printMatrix*(matrixA);

System.***out***.println("\nMatrix B:");

*printMatrix*(matrixB);

**int**[][] resultMatrix = *addMatrices*(matrixA, matrixB);

System.***out***.println("\nMatrix A + Matrix B:");

*printMatrix*(resultMatrix);

}

**public** **static** **int**[][] read(**int** N)

{

**int**[][] matrix = **new** **int**[N][N];

**for** (**int** i = 0; i < N; i++) {

**for** (**int** j = 0; j < N; j++) {

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

matrix[i][j]=in.nextInt();}

}

**return** matrix;

}

**public** **static** **int**[][] generateRandomMatrix(**int** N)

{

**int**[][] matrix = **new** **int**[N][N];

**for** (**int** i = 0; i < N; i++) {

**for** (**int** j = 0; j < N; j++) {

matrix[i][j] = (**int**) (Math.*random*() \* 100); // Fill with random values between 0 and 99

}

}

**return** matrix;

}

**public** **static** **int**[][] addMatrices(**int**[][] matrixA, **int**[][] matrixB)

{

**int** N = matrixA.length;

**int**[][] resultMatrix = **new** **int**[N][N];

**for** (**int** i = 0; i < N; i++) {

**for** (**int** j = 0; j < N; j++) {

resultMatrix[i][j] = matrixA[i][j] + matrixB[i][j];

}

}

**return** resultMatrix;

}

**public** **static** **void** printMatrix(**int**[][] matrix)

{

**int** N = matrix.length;

**for** (**int** i = 0; i < N; i++) {

**for** (**int** j = 0; j < N; j++) {

System.***out***.print(matrix[i][j] + "\t");

}

System.***out***.println();

}

}

}

OUTPUT:

Set Commanad line argument by: Run->Run configuration->arguments

The order of the matrix is:3

Read first matrix NXN elements

1 2 1

2 3 4

5 6 4

Matrix A:

1 2 1

2 3 4

5 6 4

Matrix B:

83 44 20

37 98 34

34 60 22

Matrix A + Matrix B:

84 46 21

39 101 38

39 66 26

1. Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.

**public** **class** Stack

{

**private** **int**[] stackArray;

**private** **int** top;

**private** **int** maxSize;

**public** Stack(**int** size)

{

maxSize = size;

stackArray = **new** **int**[maxSize];

top = -1;

}

**public** **boolean** isEmpty()

{

**return** top == -1;

}

**public** **boolean** isFull()

{

**return** top == maxSize - 1;

}

**public** **void** push(**int** value)

{

**if** (isFull())

{

System.***out***.println("Stack is full. Cannot push " + value);

}

**else**

{

stackArray[++top] = value;

System.***out***.println("Pushed: " + value);

}

}

**public** **int** pop()

{

**if** (isEmpty())

{

System.***out***.println("Stack is empty. Cannot pop.");

**return** -1; // Return a sentinel value indicating an error

}

**else**

{

**int** poppedValue = stackArray[top--];

System.***out***.println("Popped: " + poppedValue);

**return** poppedValue;

}

}

**public** **int** top()

{

**if** (isEmpty())

{

System.***out***.println("Stack is empty");

**return** -1; // Return a sentinel value indicating an error

}

**else**

{

**int** topValue = stackArray[top];

System.***out***.println("Top Element: " + topValue);

**return** topValue;

}

}

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

{

Stack stack = **new** Stack(10);

// Push some values onto the stack

stack.push(5);

stack.push(10);

stack.push(20);

// top element

stack.top();

// Pop elements from the stack

stack.pop();

stack.pop();

stack.pop();

stack.pop(); // Attempt to pop when the stack is empty

// Peek when the stack is empty

stack.top();

} }

OUTPUT:

Pushed: 5

Pushed: 10

Pushed: 20

Top Element: 20

Popped: 20

Popped: 10

Popped: 5

Stack is empty. Cannot pop.

Stack is empty

1. A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.

**public** **class** Employee

{

**private** **int** id;

**private** String name;

**private** **double** salary;

**public** Employee(**int** id, String name, **double** salary)

{

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

}

/\* public int getId() {

return id;

}

public String getName() {

return name;

}

public double getSalary() {

return salary;

}\*/

**public** **void** raiseSalary(**double** percent)

{

**if** (percent > 0)

{

**double** increaseAmount = salary \* (percent / 100);

salary += increaseAmount;

System.***out***.println(name + "'s salary increased by " + percent + "%.");

}

**else**

{

System.***out***.println("Invalid percentage. Salary remains unchanged.");

}

}

// @Override

**public** String toString()

{

**return** "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";

}

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

{

// Create Employee objects

Employee emp1 = **new** Employee(1, "Raam", 50000.0);

Employee emp2 = **new** Employee(2, "Krishn", 60000.0);

// Display initial employee details

System.***out***.println("The Employee Details:");

System.***out***.println(emp1);

System.***out***.println(emp2);

// Raise salary for employees

emp1.raiseSalary(10.0);

emp2.raiseSalary(5.0);

// Display updated employee details

System.***out***.println("\nUpdated Employee Details:");

System.***out***.println(emp1);

System.***out***.println(emp2);

}

}

OUTPUT:

The Employee Details:

Employee [id=1, name=Raam, salary=50000.0]

Employee [id=2, name=Krishn, salary=60000.0]

Raam's salary increased by 10.0%.

Krishn's salary increased by 5.0%.

Updated Employee Details:

Employee [id=1, name=Raam, salary=55000.0]

Employee [id=2, name=Krishn, salary=63000.0]

4. A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:

● Two instance variables x (int) and y (int).

● A default (or "no-arg") constructor that construct a point at the default location of (0, 0).

● A overloaded constructor that constructs a point with the given x and y coordinates.

● A method setXY() to set both x and y.

● A method getXY() which returns the x and y in a 2-element int array.

● A toString() method that returns a string description of the instance in the format "(x, y)".

● A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates

● An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another)

● Another overloaded distance() method that returns the distance from this point to the origin (0,0) Develop the

code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined

in the class.

**class** MyPoint

{

**private** **int** x;

**private** **int** y;

**public** MyPoint()

{

// Default constructor initializes the point at (0, 0)

**this**.x = 0;

**this**.y = 0;

}

**public** MyPoint(**int** x, **int** y)

{

// Overloaded constructor to set x and y coordinates

**this**.x = x;

**this**.y = y;

}

**public** **void** setXY(**int** x, **int** y)

{

// Set both x and y coordinates

**this**.x = x;

**this**.y = y;

}

**public** **int**[] getXY()

{

// Return x and y in a 2-element int array

**return** **new** **int**[]{x, y};

}

@Override

**public** String toString()

{

// Return a string description of the point in the format "(x, y)"

**return** "(" + x + ", " + y + ")";

}

**public** **double** distance(**int** x, **int** y)

{

// Calculate the distance from this point to another point at (x, y) coordinates

**int** dx = **this**.x - x;

**int** dy = **this**.y - y;

**return** Math.*sqrt*(dx \* dx + dy \* dy);

}

**public** **double** distance(MyPoint another)

{

// Calculate the distance from this point to another MyPoint instance

**return** distance(another.x, another.y);

}

**public** **double** distance()

{

// Calculate the distance from this point to the origin (0,0)

**return** distance(0, 0);

}

}

**public** **class** Point {

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

// Create MyPoint objects using both constructors

MyPoint point1 = **new** MyPoint();

MyPoint point2 = **new** MyPoint(3, 4);

// Display the points

System.***out***.println("Point 1: " + point1);

System.***out***.println("Point 2: " + point2);

// Set new coordinates for point1

point1.setXY(1, 2);

System.***out***.println("New coordinates for Point 1: " + point1);

// Get and display x and y coordinates of point2

**int**[] coordinates = point2.getXY();

System.***out***.println("Point 2 Coordinates: x = " + coordinates[0] + ", y = " + coordinates[1]);

// Calculate and display distances

**double** distance1 = point1.distance(point2);

**double** distance2 = point1.distance(5, 6);

**double** distance3 = point1.distance();

System.***out***.println("Distance from Point 1 to Point 2: " + distance1);

System.***out***.println("Distance from Point 1 to (5, 6): " + distance2);

System.***out***.println("Distance from Point 1 to the origin (0,0): " + distance3);

}

}

OUTPUT:

Point 1: (0, 0)

Point 2: (3, 4)

New coordinates for Point 1: (1, 2)

Point 2 Coordinates: x = 3, y = 4

Distance from Point 1 to Point 2: 2.8284271247461903

Distance from Point 1 to (5, 6): 5.656854249492381

Distance from Point 1 to the origin (0,0): 2.23606797749979

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.

**class** Shape {

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

{

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

}

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

{

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

}

}

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

{

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

{

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

}

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

{

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

}

}

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

{

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

{

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

}

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

{

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

}

}

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

{

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

{

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

}

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

{

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

}

}

**public** **class** Main

{

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

{

Shape[] shapes = **new** Shape[3];

shapes[0] = **new** Circle();

shapes[1] = **new** Triangle();

shapes[2] = **new** Square();

**for** (Shape shape : shapes)

{

shape.draw();

shape.erase();

System.***out***.println();

}

}

}

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 class Shape

{

abstract double calculateArea();

abstract double calculatePerimeter();

}

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;

}

}

class Triangle extends Shape

{

private double side1;

private double side2;

private double side3;

public Triangle(double side1, double side2, double side3)

{

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

@Override

double calculateArea()

{

double s = (side1 + side2 + side3) / 2; // Semi-perimeter

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

}

@Override

double calculatePerimeter()

{

return side1 + side2 + side3;

}

}

public class Main

{

public static void main(String[] args)

{

double r = 4.0;

Circle circle = new Circle(r);

double ts1 = 3.0, ts2 = 4.0, ts3 = 5.0;

Triangle triangle = new Triangle(ts1, ts2, ts3);

System.out.println("Radius of the Circle"+r);

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

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

System.out.println("\nSides of the Traiangel are: "+ts1+','+ts2+','+ts3);

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

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

}

}

OUTPUT:

Radius of the Circle4.0

Area of the Circle: 50.26548245743669

Perimeter of the Circle: 25.132741228718345

Sides of the Traiangel are: 3.0,4.0,5.0

Area of the Triangle: 6.0

Perimeter of the Triangle: 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

**interface** Resizable

{

// Declare the abstract method "resizeWidth" to resize the width

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

// Declare the abstract method "resizeHeight" to resize the height

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

}

// Rectangle.java

// Declare the Rectangle class, which implements the Resizable interface

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

{

// Declare private instance variables to store width and height

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

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

// Constructor for initializing the width and height

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

{

**this**.width = width;

**this**.height = height;

}

// Implement the "resizeWidth" method to resize the width

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

{

**this**.width = width;

}

// Implement the "resizeHeight" method to resize the height

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

{

**this**.height = height;

}

// Method to print the current width and height of the rectangle

**public** **void** printSize()

{

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

}

}

// Main.java

// Declare the Main class

**public** **class** Program7

{

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

{

// Create an instance of the Rectangle class with an initial size

Rectangle rectangle = **new** Rectangle(100, 150);

// Print the initial size of the rectangle

rectangle.printSize();

// Resize the rectangle by changing its width and height

rectangle.resizeWidth(150);

rectangle.resizeHeight(200);

// Print the updated size of the rectangle

rectangle.printSize();

}

}

OUTPUT:

Width: 100, Height: 150

Width: 150, Height: 200

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.

**class** Outer

{

**public** **void** display()

{

System.***out***.println("outer class");

}

// this class is nested class declared inside the outer class

**class** inner

{

**public** **void** display()

{

System.***out***.println("inner class");

}

}

}

**public** **class** Nested

{

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

{

Outer obj = **new** Outer();

obj.display();

Outer.inner obj2 = obj.**new** inner();

obj2.display();

}

}

OUTPUT

outer class

inner class

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

**import** java.util.Scanner;

**class** Division

{

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

{

**int** a,b,result;

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

System.***out***.println("Input two integers");

a=input.nextInt();

b=input.nextInt();

**try**

{

result=a/b; System.***out***.println("Result="+result);

}

**catch**(ArithmeticException e)

{

System.***out***.println("exception caught: Divide by zero error"+e);

}

}

}

OUTPUT

Input two integers

10

0

exception caught: Divide by zero errorjava.lang.ArithmeticException: / by zero

Input two integers

10

5

Result=2

10. Develop a JAVA program to create a package named mypack and import & implement it in a suitable class.

**package** myPack;

**import** java.util.\*;

**public** **class** myPack

{

**double** inr,usd; Scanner in=**new** Scanner(System.***in***);

**public** **void** dollartorupee()

{

System.***out***.println("Enter dollars to convert into Rupees:");

usd=in.nextInt();

inr=usd\*67;

System.***out***.println("Dollar ="+usd+"equal to INR="+inr);

}

**public** **void** rupeetodollar()

{

System.***out***.println("Enter Rupee to convert into Dollars:");

inr=in.nextInt();

usd=inr/67;

System.***out***.println("Rupee ="+inr+"equal to Dollars="+usd);

}

}

**import** java.util.\*;

**import** java.io.\*;

**import** myPack.\*;

**class** Converter

{

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

{

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

**int** choice,ch;

myPack c=**new** myPack();

**do**

{

System.***out***.println("1.dollar to rupee ");

System.***out***.println("2.rupee to dollar ");

System.***out***.println("Enter ur choice"); choice=s.nextInt();

**switch**(choice)

{

**case** 1:

{

c.dollartorupee();

**break**;

}

**case** 2:

{

c.rupeetodollar();

**break**;

}

}

System.***out***.println("Enter 0 to quit and 1 to continue ");

ch=s.nextInt();

}**while**(ch==1);

}

}

OUTPUT

1.dollar to rupee

2.rupee to dollar

Enter ur choice

1

Enter dollars to convert into Rupees:

10

Dollar =10.0equal to INR=670.0

Enter 0 to quit and 1 to continue

1

1.dollar to rupee

2.rupee to dollar

Enter ur choice

2

Enter Rupee to convert into Dollars:

670

Rupee =670.0equal to Dollars=10.0

Enter 0 to quit and 1 to continue

0

11. Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds).

**import** java.util.Random;

**class** SquareThread **implements** Runnable

{

**int** x;

SquareThread(**int** x)

{

**this**.x = x;

}

**public** **void** run()

{

System.***out***.println("Thread Name:Square Thread and Square of " + x + " is: " + x \* x);

}

}

**class** CubeThread **implements** Runnable

{

**int** x;

CubeThread(**int** x)

{

**this**.x = x;

}

**public** **void** run()

{

System.***out***.println("Thread Name:Cube Thread and Cube of " + x + " is: " + x \* x \* x);

}

}

**class** RandomThread **implements** Runnable

{

Random r;

Thread t2, t3; **public** **void** run()

{

**int** num;

r = **new** Random();

**try**

{

**while** (**true**)

{

num = r.nextInt(100);

System.***out***.println("Main Thread and Generated Number is " + num);

t2 = **new** Thread(**new** SquareThread(num));

t2.start();

t3 = **new** Thread(**new** CubeThread(num));

t3.start();

Thread.*sleep*(500);

System.***out***.println(" ");

}

}

**catch** (Exception ex)

{

System.***out***.println("Interrupted Exception");

}

}

}

**public** **class** MainThread

{

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

{

RandomThread thread\_obj = **new** RandomThread();

Thread t1 = **new** Thread(thread\_obj);

t1.start();

}

}

OUTPUT

Main Thread and Generated Number is 91

Thread Name:Square Thread and Square of 91 is: 8281

Thread Name:Cube Thread and Cube of 91 is: 753571

Main Thread and Generated Number is 98

Thread Name:Square Thread and Square of 98 is: 9604

Thread Name:Cube Thread and Cube of 98 is: 941192

Main Thread and Generated Number is 65

Thread Name:Square Thread and Square of 65 is: 4225

Thread Name:Cube Thread and Cube of 65 is: 274625

Main Thread and Generated Number is 6

Thread Name:Square Thread and Square of 6 is: 36

Thread Name:Cube Thread and Cube of 6 is: 216

Main Thread and Generated Number is 55

Thread Name:Square Thread and Square of 55 is: 3025

Thread Name:Cube Thread and Cube of 55 is: 166375

Main Thread and Generated Number is 42

Thread Name:Cube Thread and Cube of 42 is: 74088

Thread Name:Square Thread and Square of 42 is: 1764

Main Thread and Generated Number is 12

Thread Name:Square Thread and Square of 12 is: 144

Thread Name:Cube Thread and Cube of 12 is: 1728

……..

12.Write a program to create a class MyThread in this class a constructor, call the base class constructor, using super, and starts the thread. The run method of the class starts after this. It can be observed that both the main thread and the created child thread are executed concurrently**.**

**class** MyThread **extends** Thread

{

MyThread()

{

**super** ("Using Thread class");

System.***out***.println ("Child thread:" + **this**);

start();

}

**public** **void** run()

{

**try**

{

**for** ( **int** i =5; i > 0; i--)

{

System.***out***.println ("Child thread" + i);

Thread.*sleep* (500);

}

}

**catch** (InterruptedException e) { }

System.***out***.println ("exiting child thread …");

}

}

**class** TestMyThread

{

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

{

**new** MyThread();

**try**

{

**for** ( **int** k = 5; k < 0; k--)

{

System.***out***.println ("Running main thread :" + k);

Thread.*sleep*(1000);

}

}

**catch** (InterruptedException e) { }

System.***out***.println ("Exiting main thread . . .");

}

}

**OUTPUT**

Child thread:Thread[Using Thread class,5,main]

Exiting main thread . . .

Child thread5

Child thread4

Child thread3

Child thread2

Child thread1

exiting child thread …