



SRM VALLIAMMAI ENGINEERING COLLEGE
(An Autonomous Institution)

SRM Nagar, Kattankulathur-603203



DEPARTMENT OF INFORMATION TECHNOLOGY

ACADEMIC YEAR: 2025-2026

ODD SEMESTER

LAB MANUAL

(REGULATION - 2023)

**CS3364- OBJECT ORIENTED
PROGRAMMING LAB**

THIRD SEMESTER

Prepared By

Ms. S. Shenbagavadivu, Assistant Professor (Sel.G)
Ms.G.Santhiya, Assistant Professor (Sr.G)

TABLE OF CONTENTS

E.NO	EXPERIMENT NAME	Pg. No.
a	PEO,PO,PSO	03-05
b	Syllabus with CO	06
c	Co, Co-Po Matrix, Co-PSO Matrix	07
d	Mode of Assessment	08
e	Introduction/ Description of major software used	09-11
1	Implementation of Search and Sort	12-15
2	Implementation of Stack and Queue using Classes and Objects	16-18
3	Implementation of Inheritance	19-20
4	Implementation of Abstract classes and Abstract methods	21-26
5	Implementation of Interfaces	27-28
6	Implementation of Exception Handling	29-37
7	Implementation of Multi Threading	38-45
8	Implementation of File Handling	46-47
9	Implementation of Generic Classes	48-53
10	Implementation of JavaFx Controls	54-55
11	Mini Project	56-67
12	TOPIC BEYOND SYLLABUS: Implementation of String Operation using Array List	68-70
13	TOPIC BEYOND SYLLABUS: Implementation of Packages	71-73
14	TOPIC BEYOND SYLLABUS : Implementation of JDBC	74-77

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. To afford the necessary background in the field of Information Technology to deal with engineering problems to excel as engineering professionals in industries.
2. To improve the qualities like creativity, leadership, teamwork and skill thus contributing towards the growth and development of society.
3. To develop ability among students towards innovation and entrepreneurship that caters to the needs of Industry and society.
4. To inculcate an attitude for life-long learning process through the use of information technology sources.
5. To prepare them to be innovative and ethical leaders, both in their chosen profession and in other activities.

PROGRAMME OUTCOMES (POs)

After going through the four years of study, Information Technology Graduates will exhibit ability to:

PO#	Graduate Attribute	Programme Outcome
1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2	Problem analysis	Identify, formulate, 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 public health and safety, and 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 the 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

PROGRAM SPECIFIC OUTCOMES (PSOs)

By the completion of Information Technology program the student will have following Program specific outcomes

1. Design secured database applications involving planning, development and maintenance using state of the art methodologies based on ethical values.
2. Design and develop solutions for modern business environments coherent with the advanced technologies and tools.
3. Design, plan and setting up the network that is helpful for contemporary business environments using latest hardware components.
4. Planning and defining test activities by preparing test cases that can predict and correct errors ensuring a socially transformed product catering all technological needs.



OBJECTIVES:

1. To build software development skills using java programming for real-world applications.
2. To understand and apply the concepts of objects and classes.
3. To Build packages, interfaces, inheritance.
4. To implement exception handling and file processing.
5. To develop applications using generic programming and event handling

LIST OF EXPERIMENTS:

1. Solve problems by using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
2. Develop stack and queue data structures using classes and objects.
3. Develop a java application with an Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club funds. Generate pay slips for the employees with their gross and net salary.
4. Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method printArea() that prints the area of the given shape.
5. Solve the above problem using an interface.
6. Implement exception handling and creation of user defined exceptions.
7. Write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.
8. Write a program to perform file operations.
9. Develop applications to demonstrate the features of generics classes.
10. Develop applications using Swing and JavaFX controls, layouts and menus.
11. Develop a mini project for any application using Java concepts.

Operating Systems: Linux / Windows**Front End Tools: Eclipse IDE / Netbeans IDE****TOTAL: 45 PERIODS**

OUTCOMES:

At the end of the course, the student should be able to:

1. Design and develop java programs using object oriented programming concepts
2. Develop simple applications using object oriented concepts such as package, exceptions
3. Implement multithreading, and generics concepts
4. Create GUIs and event driven programming applications for real world problems
5. Implement and deploy web applications using Java

CO – PO – PSO Mapping

CO	PO												PSO			
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4
1	3	-	-	-	2	-	-	-	-	-	-	-	3	2	-	-
2	2	2	-	-	-	-	-	-	-	-	-	-	3	2	-	-
3	2	2	-	-	-	-	-	-	-	-	-	-	3	-	-	-
4	3	-	-	-	-	-	-	-	-	-	-	-	3	2	-	-
5	3	3	3	-	2	-	-	-	-	-	-	-	3	2	-	-
Avg	2.6	2.3	3.0		2.0	-	-	-	-	-	-	-	3.0	2.0	-	-

MODE OF ASSESSMENT

EVALUATION PROCEDURE FOR EACH EXPERIMENT

S.No	Description	Mark
1.	Aim & Pre-Lab discussion	20
2.	Observation	20
3.	Conduction and Execution	30
4.	Output & Result	10
5.	Viva	20
Total		100

INTERNAL ASSESSMENT FOR LABORATORY

S.No	Description	Mark
1.	Conduction & Execution of Experiment	25
2.	Record	10
3.	Model Test	15
Total		50

INTRODUCTION / DESCRIPTION OF MAJOR SOFTWARE USED

JAVA:

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere. Java is free to download.

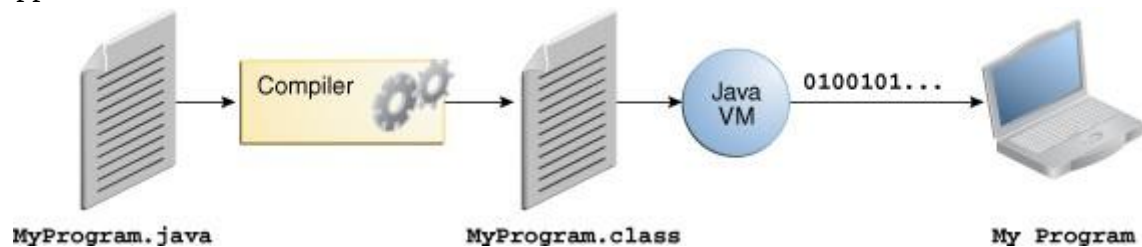
The latest Java version contains important enhancements to improve performance, stability and security of the Java applications that run on your machine. Installing this free update will ensure that your Java applications continue to run safely and efficiently.

The Java Runtime Environment (JRE) is what you get when you download Java software. The JRE consists of the Java Virtual Machine (JVM), Java platform core classes, and supporting Java platform libraries. The JRE is the runtime portion of Java software, which is all you need to run it in your Web browser.

The Java Plug-in software is a component of the Java Runtime Environment (JRE). The JRE allows applets written in the Java programming language to run inside various browsers. The Java Plug-in software is not a standalone program and cannot be installed separately.

The Java Virtual Machine is only one aspect of Java software that is involved in web interaction. The Java Virtual Machine is built right into your Java software download, and helps run Java applications.

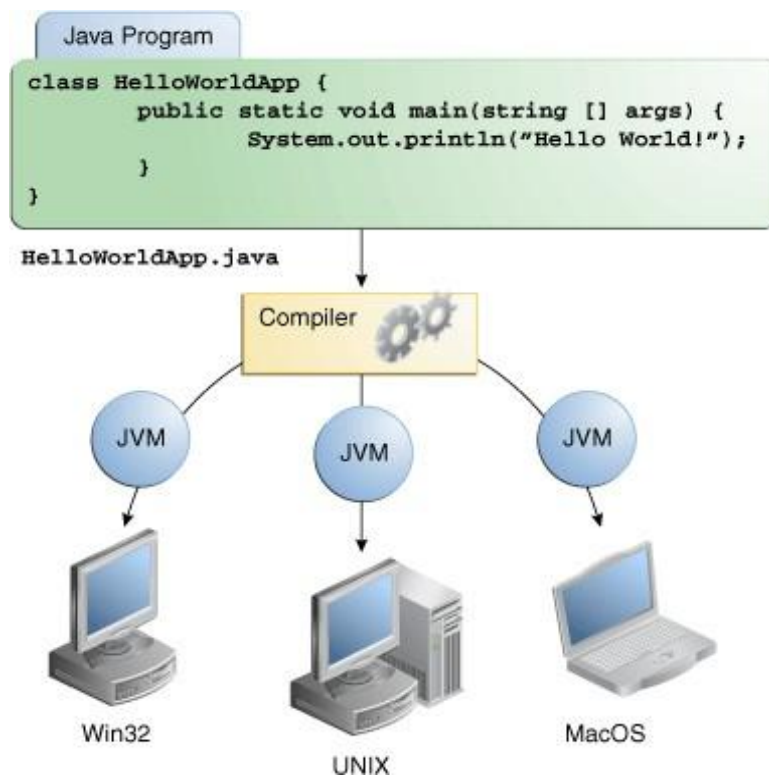
In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains *bytecodes* — the machine language of the Java Virtual Machine (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the [Java SE HotSpot at a Glance](#), perform additional steps at runtime to give your application a performance boost. This includes various tasks such as

finding performance bottlenecks and recompiling (to native code) frequently used sections of code.



Through the Java VM, the same application is capable of running on multiple platforms.

The Java Platform

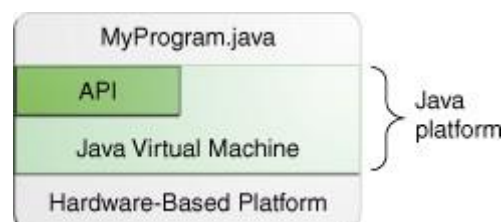
A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine
- The Java Application Programming Interface (API)

Java Virtual Machine; it's the base for the Java platform and is ported onto various hardware-based platforms.

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages.



The API and Java Virtual Machine insulate the program from the underlying hardware.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

Key advantages of Java Programming

Object Oriented – In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

Platform Independent – Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

Simple – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

Secure – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Architecture-neutral – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

Portable – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

Robust – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

AIM:

To develop a Java application to search an element in an array using sequential search algorithm.

ALGORITHM:

Step 1: Start the program

Step 2: Declare an array and search element as key.

Step 3: Traverse the array until the key is found.

Step 4: If the key is found, return the index position of the array element

Step 5: If the key element is not found, return -1.

Step 6: Stop the program.

PROGRAM:**LinearSearch.java**

```
public class LinearSearch
{
    static intsearch(intarr[], int n, int s)
    {
        for (inti = 0; i< n; i++)
        {
            if (arr[i] == s)
                return i;
        }
        return -1;
    }
    public static void main(String[] args)
    {
        int[] arr = { 3, 4, 1, 7, 5 };
        int n = arr.length;
        int s = 4;
        int index = search(arr, n, s);
        if (index == -1)
            System.out.println("Element is not present in the array");
        else
            System.out.println("Element found at index " + index);
    }
}
```

OUTPUT:

```
D:\Java>javac LinearSearch.java
```

```
D:\Java>java LinearSearch
```

```
Element found at index 1
```

RESULT:

Thus, the Java application to perform sequential search was implemented and executed successfully.

AIM:

To develop a Java application to search an element in an array using binary search algorithm.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare an array and search element as key.

Step 3: Compare search key with the middle element.

Step 4: If key matches with the middle element, return the mid index.

Step 5: Otherwise, if key is greater than the mid element recur for the right half.

Step 6: Otherwise (s is smaller) recur for the left half.

Step 7: Stop the program.

PROGRAM:**BinarySearch.java**

```
class BinarySearch
{
    public static int binarySearch(int arr[], int first, int last, int key)
    {
        if (last >= first)
        {
            int mid = (first + last) / 2;
            if (arr[mid] == key)
            {
                return mid;
            }
            else if (arr[mid] > key)
            {
                return binarySearch(arr, first, mid - 1, key);
            }
            else
            {
                return binarySearch(arr, mid + 1, last, key);
            }
        }
    }
}
```

```
        return -1;
    }
    public static void main(String args[])
    {
        intarr[] = {10,20,30,40,50};
        int key = 30;
        int last=arr.length-1;
        int result = binarySearch(arr,0,last,key);
        if (result == -1)
            System.out.println("Element is not found!");
        else
            System.out.println("Element is found at index: "+result);
    }
}
```

OUTPUT:

D:\Java>javac BinarySearch.java

D:\Java>java BinarySearch

Element is found at index: 2

D:\Java>

RESULT:

Thus, the Java application to perform binary search was implemented and executed successfully.

AIM:

To develop a Java application to sort an array of elements in ascending order using insertion sort.

ALGORITHM:

Step 1: Start the program.

Step 2: Define an array num to store N numbers for insertion sort.

Step 3: Run an outer loop i from 1 to N to repeat the process.

Step 4: Store the number num[i] to be inserted at proper place in variable x.

Step 5: Run a while loop j inside the body of the outer loop i from i-1 to 0.

Step 6: Check if the value of x is less than value of num[j] then shift the number num[j] towards right else break the inner loop j.

Step 7: Outside the body of inner loop j insert the value of x at num[j+1] position.

Step 8: Print the sorted array.

Step 9: Stop the program.

PROGRAM:**InsertionSort.java**

```
public class InsertionSort
{
    public static void main(String args[])
    {
        int num[] = {12,9,37,86,2,17,5};
        int i,j,x;
        System.out.println("Array before Insertion Sort");
        for(i=0; i<num.length; i++)
        {
            System.out.print(num[i]+" ");
        }
        for(i=1; i<num.length; i++)
        {
            x=num[i];
            j=i-1;
            while(j>=0)
            {
                if(x<num[j])
```

```

        {
            num[j+1]=num[j];
        }
        else
        {
            break;
        }
        j=j-1;
    }

    num[j+1]=x;
}
System.out.print("\n\nArray after Insertion Sort\n");
for(i=0; i<num.length; i++)
{
    System.out.print(num[i]+" ");
}
}
}

```

OUTPUT:

D:\Java>javac InsertionSort.java

D:\Java>java InsertionSort

Array before Insertion Sort

12 9 37 86 2 17 5

Array after Insertion Sort

2 5 9 12 17 37 86

D:\Java>

RESULT:

Thus, the Java application to sort an array of N elements using insertion sort was implemented and executed successfully.

EX.NO.:1(d)	SELECTION SORT
-------------	----------------

AIM:

To develop a Java application to sort an array of elements in ascending order using selection sort.

ALGORITHM:

Step 1: Start the program.

Step 2: Select the first unsorted element as the minimum.

Step 3: For each of the unsorted elements, if the element is <minimum, set element as new minimum.

Step 4: Swap minimum with first unsorted position.

Step 5: Repeat steps 2-4 for (n-1) elements until the list is sorted.

Step 6: Print the sorted array.

Step 7: Stop the program.

PROGRAM:**SelectionSort.java**

```
public class SelectionSort
{
    public static void selectionsort(int[] arr)
    {
        int n=arr.length;
        for(int i=0;i<n-1;i++)
        {
            int min=i;
            for(int j=i+1;j<n;j++)
            {
                if(arr[j]<arr[min])
                {
                    min=j;
                }
            }
        }
    }
}
```

```

        int temp=arr[i];
        arr[i]=arr[min];
        arr[min]=temp;
    }
}

public static void main(String[] args)
{
    int[] arr= {15,21,6,3,19,20};
    System.out.println("Elements in the array before Sorting");
    for(int i:arr)
        System.out.print(i+" ");
    selectionsort(arr);
    System.out.println("\nElements in the array after Sorting");
    for(int i:arr)
        System.out.print(i+" ");
    }
}

```

OUTPUT:

```

D:\Java>javac SelectionSort.java
D:\Java>java SelectionSort
Elements in the array before Sorting
15 21 6 3 19 20
Elements in the array after Sorting
3 6 15 19 20 21
D:\Java>

```

RESULT:

Thus, the Java application to sort an array of N elements using selection sort was implemented and executed successfully.

AIM:

To develop a Java program to implement stack data structure using classes and objects.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a class named Stack and declare the instance variables st[], top, maxsize as private.

Step3: Use constructor to allocate memory space for the array and initialize top as -1.

Step 4: Define methods such as isFull(), isEmpty(), push(), pop() and printStack();

Step 5: Push Operation

Step 5.1: Check whether stack has some space or stack is full.

Step 5.2: If the stack has no space then display “overflow” and exit.

Step 5.3: If the stack has space then increase top by 1 to point next empty space.

Step 5.4: Add element to the new stack location, where top is pointing.

Step 5.5: Push operation performed successfully.

Step 6: Pop operation

Step 6.1: Check whether stack has some element or stack is empty.

Step 6.2: If the stack has no element means it is empty then display “underflow”

Step 6.3: If the stack has some element, accesses the data element at which top is pointing.

Step 6.4: Decrease the value of top by 1.

Step 6.5: Pop operation performed successfully.

Step 7: PrintStack operation

Step 7.1: Check whether stack has some element or stack is empty.

Step 7.2: If the stack has no element means it is empty then display “underflow”.

Step 7.3: If the stack has some element, traverse the array from top to bottom and display the elements.

Step 8: Define the main() method

Step 9: Create object of Stack class.

Step 10: Display the menu and get the user choice and invoke appropriate method.

Step 11: Stop the program.

PROGRAM:

Stack.java

```
import java.util.Scanner;
public class Stack
{
    private int maxsize, top;
    private int[] st;
    public Stack(int size)
    {
        maxsize = size;
        st = new int[maxsize];
        top = -1;
    }
    boolean isEmpty()
    {
        return top == -1;
    }
    boolean isFull()
    {
        return top == maxsize - 1;
    }
    public void push(int element)
    {
        if (isFull())
            System.out.println("Overflow");
        else
            st[++top] = element;
    }

    public int pop()
    {
        if (isEmpty())
        {
            System.out.println("UnderFlow");
            return (-1);
        }
        return (st[top--]);
    }
    public void printStack()
    {
        System.out.println("Stack Elements:");
        for (int i = top; i >= 0; i--)
```

```

        System.out.println(st[i]);
    }
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter stack size");
        int size=sc.nextInt();
        Stack obj = new Stack(size);
        while (true)
        {
            System.out.println("\nSTACK\n*****\n1.PUSH\n2.POP\n3.Display\n4.EXIT\nEnter your choice");
            intch = sc.nextInt();
            switch (ch)
            {
                case 1:
                    System.out.println("Enter Element");
                    int n = sc.nextInt();
                    obj.push(n);
                    break;
                case 2:
                    System.out.printf("Poped element is %d", obj.pop());
                    break;
                case 3:
                    obj.printStack();
                    break;
                case 4:
                    System.exit(0);
                default:
                    System.out.println("Wrong option");
            }
        }
    }
}

```

OUTPUT:

```

D:\Java>javac Stack.java
D:\Java>java Stack
Enter stack size
5

```

```

STACK
*****

```

```

1. PUSH
1. POP

```

2.Display
4.EXIT
Enter your choice
1
Enter Element
12

STACK

1.PUSH
1. POP
2.Display
4.EXIT
Enter your choice
1
Enter Element
34

STACK

1.PUSH
1. POP
2.Display
4.EXIT
Enter your choice
1
Enter Element
56

STACK

1.PUSH
1. POP
2.Display
4.EXIT
Enter your choice
1
Enter Element
78

STACK

1.PUSH
1. POP
2.Display
4.EXIT
Enter your choice
3
Stack Elements:
78

56
34
12

STACK

1.PUSH

1. POP

2.Display

4.EXIT

Enter your choice

2

Poped element is 78

STACK

1.PUSH

1.POP

2.Display

4.EXIT

Enter your choice

3

Stack Elements:

56

34

12

STACK

1.PUSH

3.POP

4.Display

4.EXIT

Enter your choice

4

D:\Java>

RESULT:

Thus, the Java program to implement stack data structure using classes and objects has developed and executed successfully.

AIM:

To develop a Java program to implement queue data structure using classes and objects.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a class named Queue and declare the instance variables items[], front, rear, maxsize as private.

Step3: Use constructor to allocate memory space for the array and initialize front as -1 and rear as -1.

Step 4: Define methods such as isFull(), isEmpty(), enqueue(), dequeue() and display();

Step 5: Enqueue Operation

Step 5.1: Check whether queue has some space or queue is full.

Step 5.2: If the queue has no space then display “overflow” and exit.

Step 5.3: If the queue has space then increase rear by 1 to point next empty space.

Step 5.4: Add element to the new location, where rear is pointing.

Step 5.5: Enqueue operation performed successfully.

Step 6: Dequeue operation

Step 6.1: Check whether queue has some element or queue is empty.

Step 6.2: If the queue has no element means it is empty then display “underflow”

Step 6.3: If the queue has some element, accesses the data element at which front is pointing.

Step 6.4: Increment the value of front by 1.

Step 6.5: Dequeue operation performed successfully.

Step 7: Display operation

Step 7.1: Check whether queue has some element or queue is empty.

Step 7.2: If the stack has no element means it is empty then display “underflow”.

Step 7.3: If the stack has some element, traverse the array from front to rear and display the elements.

Step 8: Define the main() method

Step 9: Create object of Queue class.

Step 10: Display the menu and get the user choice and invoke appropriate method.

Step 11: Stop the program.

PROGRAM:

Queue.java

```
import java.util.Scanner;

public class Queue
{

    private int items[];
    private int maxsize, front, rear;
    Queue(int size)
    {
        maxsize=size;
        items = new int[size];
        front = -1;
        rear = -1;
    }
    boolean isFull()
    {
        if (front == 0 && rear ==maxsize-1)
        {
            return true;
        }
        return false;
    }
    boolean isEmpty()
    {
        if (front == -1)
            return true;
        else
            return false;
    }

    void enqueue(int element)
    {
        if (isFull())
```

```

        {
            System.out.println("Queue is full");
        }
        else
        {
            if (front == -1)
                front = 0;
            rear++;
            items[rear] = element;
            System.out.println("Inserted " + element);
        }
    }
    intdeQueue()
    {
        int element;
        if (isEmpty())
        {
            System.out.println("Queue is empty");
            return (-1);
        }
        else
        {
            element = items[front];
            if (front >= rear)
            {
                front = -1;
                rear = -1;
            }
            else
            {
                front++;
            }

            return (element);
        }
    }

```

```

    }
    void display()
    {
        inti;
        if (isEmpty())
        {
            System.out.println("Empty Queue");
        }
        else
        {
            System.out.println("\nFront index-> " + front);
            System.out.println("Items -> ");
            for (i = front; i<= rear; i++)
                System.out.print(items[i] + " ");
            System.out.println("\nRear index-> " + rear);
        }
    }

    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter queue size");
        int size=sc.nextInt();
        Queue obj = new Queue(size);
        while (true)
        {
            System.out.println("\nQUEUE\n*****\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.EXIT\nEnter your choice");
            intch = sc.nextInt();
            switch (ch)
            {
                case 1:
                    System.out.println("Enter Element");
                    int n = sc.nextInt();
                    obj.enqueue(n);

```

```

        break;
    case 2:
        System.out.printf("Dequeued element is %d",
            obj.dequeue());
        break;
    case 3:
        obj.display();
        break;
    case 4:
        System.exit(0);
    default:
        System.out.println("Wrong option");
    }
}
}
}
}

```

OUTPUT:

D:\Java>javac Queue.java

D:\Java>java Queue

Enter queue size

5

QUEUE

1.ENQUEUE

1. DEQUEUE

2. DISPLAY

3. EXIT

Enter your choice

1

Enter Element

12

Inserted 12

QUEUE

1.ENQUEUE

1. DEQUEUE

2. DISPLAY

3. EXIT

Enter your choice

Enter Element

34

Inserted 34

QUEUE

1.ENQUEUE

1. DEQUEUE
2. DISPLAY
3. EXIT

Enter your choice

1

Enter Element

56

Inserted 56

QUEUE

1.ENQUEUE

1. DEQUEUE
2. DISPLAY
3. EXIT

Enter your choice

1

Enter Element

78

Inserted 78

QUEUE

1.ENQUEUE

1. DEQUEUE
2. DISPLAY
3. EXIT

Enter your choice

3

Front index-> 0

Items ->

12 34 56 78

Rear index-> 3

QUEUE

1.ENQUEUE

1. DEQUEUE
2. DISPLAY
3. EXIT

Enter your choice

2

Poped element is 12

QUEUE

1.ENQUEUE

4. DEQUEUE

5. DISPLAY

6. EXIT

Enter your choice

3

Front index-> 1

Items ->

34 56 78

Rear index-> 3

QUEUE

1.ENQUEUE

7. DEQUEUE

8. DISPLAY

9. EXIT

Enter your choice

4

D:\Java>

RESULT:

Thus, the Java program to implement queue data structure using classes and objects has developed and executed successfully.

AIM:

To develop a java application to generate pay slip for different category of employees using the concept of inheritance.

ALGORITHM:

Step 1: Start the program.

Step 2: Create the class **Employee** with name, Empid, address, mailid, mobilenos as fields.

Step 3: Inherit the classes **Programmer**, **AssistantProfessor**, **AssociateProfessor** and **Professor** from employee class.

Step 4: Add Basic Pay (BP) as the member of all the inherited classes.

Step 5: Calculate DA as 97% of BP, HRA as 10% of BP, PF as 12% of BP, Staff club fund as 0.1% of BP.

Step 6: Calculate gross salary and net salary.

Grosssal=BP+HRA+DA

NetSal=GrossSal-(PF+Staff Club Fund)

Step 7: Create a test class **PaySlip**. Read employee details, choice and basic pay from user.

Step 8: Based on user choice and input create the object and invoke the necessary methods to display the Payslip.

Step 9: Stop the program.

PROGRAM:**PaySlip.java**

```
import java.util.Scanner;
class Employee
{
    String Emp_name, Mail_id, Address, Emp_id, Mobile_no;
    double BP, GP, NP, DA, HRA, PF, CF;
    Scanner get = new Scanner(System.in);
    Employee()
    {
        System.out.println("Enter Name of the Employee:");
        Emp_name = get.nextLine();
    }
}
```

```

        System.out.println("Enter Address of the Employee:");
        Address = get.nextLine();
        System.out.println("Enter ID of the Employee:");
        Emp_id = get.nextLine();
        System.out.println("Enter Mobile Number:");
        Mobile_no = get.nextLine();
        System.out.println("Enter E-Mail ID of the Employee :");
        Mail_id = get.nextLine();
    }
    void display()
    {
        System.out.println("Employee Name: "+Emp_name);
        System.out.println("Employee Address: "+Address);
        System.out.println("Employee ID: "+Emp_id);
        System.out.println("Employee Mobile Number: "+Mobile_no);
        System.out.println("Employee E-Mail ID"+Mail_id);
        DA=BP*0.97;
        HRA=BP*0.10;
        PF=BP*0.12;
        CF=BP*0.01;
        GP=BP+DA+HRA;
        NP=GP-PF-CF;
        System.out.println("Basic Pay :"+BP);
        System.out.println("Dearness Allowance : "+DA);
        System.out.println("House Rent Allowance :"+HRA);
        System.out.println("Provident Fund :"+PF);
        System.out.println("Club Fund :"+CF);

        System.out.println("Gross Pay :"+GP);
        System.out.println("Net Pay :"+NP);
    }
}
class Programmer extends Employee
{
    Programmer()
    {
        System.out.println("Enter Basic pay of the Programmer:");
        BP = get.nextFloat();
    }
    void display()
    {
        System.out.println("======"+"\\n"+"Programmar  
Pay Slip"+"\\n"+"======"+"\\n");
        super.display();
    }
}
class AssistantProfessor extends Employee
{
    AssistantProfessor()
    {

```

```

        System.out.println("Enter Basic pay of the Assistant Professor:");
        BP = get.nextFloat();
    }
    void display()
    {
        System.out.println("======"+"\\n"+"Assistant
Professor Pay Slip"+"\\n"+"======"+"\\n");
        super.display();
    }
}
class AssociateProfessor extends Employee
{
    AssociateProfessor()
    {
        System.out.println("Enter Basic pay of the Professor:");
        BP = get.nextFloat();
    }
    void display()
    {
        System.out.println("======"+"\\n"+"Associate
Professor Pay Slip"+"\\n"+"======"+"\\n");
        super.display();
    }
}
class Professor extends Employee
{
    Professor()
    {
        System.out.println("Enter Basic pay of the Professor:");
        BP = get.nextFloat();
    }
    void display()
    {
        System.out.println("======"+"\\n"+"Professor Pay
Slip"+"\\n"+"======"+"\\n");
        super.display();
    }
}
class Payslip
{
    public static void main(String[] args)
    {
        char ans;
        Scanner sc = new Scanner(System.in);
        do
        {
            System.out.println("Main Menu");
            System.out.println("1. Programmer \\n2. Assistant Professor \\n3. Associate
Professor \\n4. Professor");
            System.out.println("Enter your choice: ");

```

```

        int choice=sc.nextInt();
        switch(choice)
        {
            case 1:
                Programmer p=new Programmer();
                p.display();
                break;
            case 2:
                AssistantProfessorap=new AssistantProfessor();
                ap.display();
                break;
            case 3:
                AssociateProfessor asp=new AssociateProfessor();
                asp.display();
                break;
            case 4:
                Professor PR=new Professor();
                PR.display();
                break;
        }

        System.out.println("Do you want to go to Main Menu?(y/n): ");
        ans=sc.next().charAt(0);
        }while(ans=='y'||ans=='Y');
        sc.close();
    }
}

```

OUTPUT:

Main Menu

10. Programmer
11. Assistant Professor
12. Associate Professor
13. Professor

Enter your choice:

1

Enter Name of the Employee:

Raja

Enter Address of the Employee:

Chennai

Enter ID of the Employee:

12345

Enter Mobile Number:

9876543210

Enter E-Mail ID of the Employee :

abc@xyz.com

Enter Basic pay of the Programmer:

56000

=====

Programmar Pay Slip

=====

Employee Name: Raja

Employee Address: Chennai

Employee ID: 12345

Employee Mobile Number: 9876543210

Employee E-Mail IDabc@xyz.com

Basic Pay :56000.0

Dearness Allowance : 54320.0

House Rent Allowance :5600.0

Provident Fund :6720.0

Club Fund :560.0

Gross Pay :115920.0

Net Pay :108640.0

Do you want to go to Main Menu?(y/n):

n

RESULT:

Thus, the Java application to generate pay slip for different category of employees was implemented using inheritance and the program was executed successfully.

AIM:

To write a Java program to calculate the area of rectangle, circle and triangle using the concept of abstract class.

ALGORITHM:

Step 1: Start the program.

Step 2: Create an abstract class named shape that contains two integers and an empty method named printArea().

Step 3: Create three classes named rectangle, triangle and circle such that each one of the classes extends the class Shape.

Step 4: Each of the inherited class from shape class should provide the implementation for the method printArea().

Step 5: In printArea() method get the input from user and calculate the area of rectangle, circle and triangle.

Step 6: In the AbstractArea, create the objects for the three inherited classes and invoke the methods and display the area values of the different shapes.

Step 7: Stop the program.

PROGRAM:**AbstractArea.java**

```
import java.util.*;
abstract class Shape
{
    int a,b;
    abstract void printArea();
}
class Rectangle extends Shape
{
    void printArea()
    {
        System.out.println("\t\tCalculating Area of Rectangle");
        Scanner input=new Scanner(System.in);
        System.out.print("Enter length: ");
        a=input.nextInt();
        System.out.print("\n\nEnter breadth: ");
```

```

        b=input.nextInt();
        int area=a*b;
        System.out.println("Area of Rectangle: "+area);
    }
}
class Triangle extends Shape
{
    void printArea()
    {
        System.out.println("\t\tCalculating Area of Triangle");
        Scanner input=new Scanner(System.in);
        System.out.print("Enter height: ");
        a=input.nextInt();
        System.out.println("\nEnter breadth: ");
        b=input.nextInt();
        double area=0.5*a*b;
        System.out.println("Area of Triangle: "+area);
    }
}
class Circle extends Shape
{
    void printArea()
    {
        System.out.println("\t\tCalculating Area of Circle");
        Scanner input=new Scanner(System.in);
        System.out.print("Enter radius: ");
        a=input.nextInt();
        double area=3.14*a*a;
        System.out.println("Area of Circle: "+area);
    }
}
class AbstractArea
{
    public static void main(String[] args)
    {
        Shape obj;
        obj=new Rectangle();
        obj.printArea();
        obj=new Triangle();
        obj.printArea();
        obj=new Circle();
        obj.printArea();
    }
}

```

OUTPUT:

Calculating Area of Rectangle

Enter length: 10

Enter breadth: 20

Area of Rectangle: 200

Calculating Area of Triangle

Enter height: 34

Enter breadth: 56

Area of Triangle: 952.0

Calculating Area of Circle

Enter radius: 23

Area of Circle: 1661.06

RESULT:

Thus, the Java program to calculate the area of rectangle, circle and triangle using the concept of abstract class was developed and executed successfully.

AIM:

To write a Java program to calculate the area of rectangle, circle and triangle by implementing the interface shape.

ALGORITHM:

Step 1: Start the program.

Step 2: Create an interface named shape that contains an empty method named printArea().

Step 3: Create three classes named rectangle, triangle and circle such that each one of the classes implements the class Shape.

Step 4: Each of the class should provide the implementation for the method printArea().

Step 5: In printArea() method get the input from user and calculate the area of rectangle, circle and triangle.

Step 6: In the TestArea class, create the objects for the three classes and invoke the Method printArea() and display the area values of the different shapes.

Step 7: Stop the program.

PROGRAM:**TestArea.java**

```
import java.util.Scanner;
interface Shape
{
    public void printArea();
}
class Rectangle implements Shape
{
    public void printArea()
    {
        System.out.println("\t\tCalculating Area of Rectangle");
        Scanner input=new Scanner(System.in);
        System.out.print("Enter length: ");
        int a=input.nextInt();
        System.out.print("\nEnter breadth: ");
        int b=input.nextInt();
        int area=a*b;
        System.out.println("Area of Rectangle: "+area);
    }
}
class Triangle implements Shape
{
    public void printArea()
```

```

        {
            System.out.println("\t\tCalculating Area of Triangle");
            Scanner input=new Scanner(System.in);
            System.out.print("Enter height: ");
            int a=input.nextInt();
            System.out.print("\nEnter breadth: ");
            int b=input.nextInt();
            double area=0.5*a*b;
            System.out.println("Area of Triangle: "+area);
        }
    }
    class Circle implements Shape
    {
        public void printArea()
        {
            System.out.println("\t\tCalculating Area of Circle");
            Scanner input=new Scanner(System.in);
            System.out.print("Enter radius: ");
            int a=input.nextInt();
            double area=3.14*a*a;
            System.out.println("Area of Circle: "+area);
        }
    }
    public class TestArea
    {
        public static void main(String[] args)
        {
            Shape obj;
            obj=new Rectangle();
            obj.printArea();
            obj=new Triangle();
            obj.printArea();
            obj=new Circle();
            obj.printArea();
        }
    }
}

```

OUTPUT:

D:\Java>javac TestArea.java

D:\Java>java TestArea

Calculating Area of Rectangle

Enter length: 14

Enter breadth: 24

Area of Rectangle: 336

Calculating Area of Triangle

Enter height: 45

Enter breadth: 32

Area of Triangle: 720.0

Calculating Area of Circle

Enter radius: 5

Area of Circle: 78.5

D:\Java>

RESULT:

Thus, the Java program to calculate the area of rectangle, circle and triangle using the concept of interface was developed and executed successfully.

AIM:

To write a Java program to implement user defined exception handling.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a class NegativeAmtException which extends Exception class.

Step 3: Create a constructor which receives the string as argument.

Step 4: Create a class named BankAccount with a constructor and methods such as deposit(), withdraw(), getBalance() and toString().

Step 5: Inside the constructor, deposit() and withdraw() methods check for negative amount.

If the amount is negative, then exception will be generated and thrown. Make these methods to specify that NegativeAmtException will be thrown.

Step 6: Create a test class UserDefinedException . Read the account number and initial balance from user and create object of BankAccount class.

Step 7: Display the menu and get the user choice and execute the required operation. Write the code that generates exception, in try block.

Step 8: Use catch block to catch and handle NegativeAmtException. Display the caught exception.

Step 9: Stop the program.

PROGRAM:**UserDefinedException.java**

```
import java.util.*;
class NegativeAmtException extends Exception
{
    String msg;
    NegativeAmtException(String msg)
    {
        this.msg=msg;
    }
    public String toString()
    {
        return msg;
    }
}
```

```

class BankAccount
{
    private double balance;
    private int accountNumber;
    public BankAccount(int accountNumber, double initialBalance) throws
    NegativeAmtException
    {
        if(initialBalance<0)
            throw new NegativeAmtException("Initial amount should not be
            negative!");
        balance = initialBalance;
        this.accountNumber = accountNumber;
    }
    public void deposit(double amount) throws NegativeAmtException
    {
        if (amount < 0)
        {
            throw new NegativeAmtException("Don't deposit negative amount!");
        }
        balance = balance + amount;
        System.out.println("Amount deposited");
        System.out.println("Balance amount : "+getBalance());
    }
    public void withdraw(double amount) throws NegativeAmtException
    {
        if (amount < 0)
        {
            throw new NegativeAmtException("Don't withdraw a negative
            amount!");
        }
        else if(amount<=balance)
        {
            balance = balance – amount;
        }
        else
        {
            System.out.println("Insufficient amount");
        }

        System.out.println("Balance amount : "+getBalance());
    }
    public double getBalance()
    {
        return balance;
    }
    public int getAccountNumber()
    {
        return accountNumber;
    }
    public String toString ()
    {

```

```

        return "Account Number :" + accountNumber + " Balance :" + balance;
    }
}
public class UserDefinedException
{
    public static void main(String[] args)
    {
        int ch, amt;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Account Number:");
        int a = sc.nextInt();
        System.out.print("Enter the initial Amount:");
        int b = sc.nextInt();
        BankAccount ac;
        try
        {
            ac = new BankAccount(a, b);
            while(true)
            {
                System.out.println("Main Menu");
                System.out.println("1. Deposit \n2. Withdraw \n3. Check Balance \n4. Display \n5. Exit");
                System.out.print("Enter your Choice: ");
                ch = sc.nextInt();
                switch(ch)
                {
                    case 1:
                        System.out.print("Enter the amount to deposit:");
                        amt = sc.nextInt();
                        ac.deposit(amt);
                        break;
                    case 2:
                        System.out.print("Enter the amount to Withdraw:");
                        amt = sc.nextInt();
                        ac.withdraw(amt);
                        break;
                    case 3:
                        System.out.println("Balance amount : "+ac.getBalance());
                        break;
                    case 4:
                        System.out.println("Your account details\n"+ac);
                        break;
                    case 5:
                        sc.close();
                        System.exit(0);
                }
            }
        }
    }
}

```

```

                                default:
                                    System.out.println("Invalid Choice");
                                }
                            }
                        }
                    catch(NegativeAmtException e)
                    {
                        System.out.println("Exception Caught : "+e);
                    }
                }
            }
        }
    }
}

```

OUTPUT:

```

Enter Account Number:1234
Enter the initial Amount:500
Main Menu
1.Deposit
2.Withdraw
3.Check Balance
4.Display
5.Exit
Enter your Choice: 1
Enter the amount to deposit:-456
Exception Caught : Don't deposit negative amount!

```

RESULT:

Thus the Java program to implement user defined exception handling was implemented and executed successfully.

AIM:

To write a java program to implement a multi-threaded application.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a class **even** which implements first thread that computes the square of the number.

Step 3: The `run()` method implements the code to be executed when thread gets executed.

Step 4: Create a class **odd** which implements second thread that computes the cube of the number.

Step 5: Create a third thread that generates random number. If the random number is even, it displays the square of the number. If the random number generated is odd, it displays the cube of the given number.

Step 6: The Multithreading is performed and the task switched between multiple threads.

Step 7: The `sleep()` method makes the thread to suspend for the specified time.

Step 8: Stop the program.

PROGRAM:**MultiThread.java**

```
import java.util.*;
class even implements Runnable
{
    public int x;
    public even(int x)
    {
        this.x = x;
    }
    public void run()
    {
        System.out.println("New Thread "+ x +" is EVEN and Square of " + x + " is: "
            + x * x);
    }
}
class odd implements Runnable
{
    public int x;
    public odd(int x)
    {
```

```

        this.x = x;
    }
    public void run()
    {
        System.out.println("New Thread "+ x +" is ODD and Cube of " + x + " is: " +
            x * x * x);
    }
}
class A extends Thread
{
    public void run()
    {
        intnum = 0;
        Random r = new Random();
        try
        {
            for (inti = 0; i < 5; i++)
            {
                num = r.nextInt(100);
                System.out.println("Main Thread and Generated Number is " +
                    num);
                if (num % 2 == 0)
                {
                    Thread t1 = new Thread(new even(num));
                    t1.start();
                }
                else
                {
                    Thread t2 = new Thread(new odd(num));
                    t2.start();
                }

                Thread.sleep(1000);
                System.out.println(".....");
            }
        }
        catch (Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
}
public class MultiThread
{
    public static void main(String[] args)

```

```
    {  
        A a = new A();  
        a.start();  
    }  
}
```

OUTPUT:

Main Thread and Generated Number is 33
New Thread 33 is ODD and Cube of 33 is: 35937

Main Thread and Generated Number is 31
New Thread 31 is ODD and Cube of 31 is: 29791

Main Thread and Generated Number is 25
New Thread 25 is ODD and Cube of 25 is: 15625

Main Thread and Generated Number is 43
New Thread 43 is ODD and Cube of 43 is: 79507

Main Thread and Generated Number is 14
New Thread 14 is EVEN and Square of 14 is: 196

RESULT:

Thus, the java program to implement multithreaded application was executed successfully.

AIM:

To write a java program to copy the contents of one file to another file using file operations.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a class FileCopy. Get the source and destination file names from the user.

Step 3: Create object of FileInputStream by passing the source file name.

Step 4: Using read() method read the contents of the file till end of the file.

Step 5: Create object of FileOutputStream by passing the second file to the constructor.

Step 6: Use write() method to write the contents to the destination file.

Step 7: Close the file input and output stream using close() method.

Step 8: Display the contents of both files.

Step 9: Stop the program.

PROGRAM:**FileCopy.java**

```
import java.io.*;
class CopyFile
{
    public static void main(String args[]) throws IOException
    {
        inti;
        FileInputStream fin = null;
        FileOutputStream fout = null;
        if(args.length != 2)
        {
            System.out.println("Usage: CopyFile from to");
            return;
        }
        System.out.println("Displaying contents of "+ args[0]+"\\n");
        try
        {
            fin = new FileInputStream(args[0]);
```

```

        do
        {
            i = fin.read();
            if(i != -1)
                System.out.print((char) i);
        } while(i != -1);
    }
    catch(IOException e)
    {
        System.out.println("Error Reading File");
    }
    finally
    {
        try
        {
            fin.close();
        }
        catch(IOException e)
        {
            System.out.println("Error Closing File");
        }
    }
    System.out.println("\nCopying contents of " + args[0] + "to " + args[1] + "\n");
    try
    {
        fin = new FileInputStream(args[0]);
        fout = new FileOutputStream(args[1]);
        do
        {
            i = fin.read();
            if(i != -1) fout.write(i);
        } while(i != -1);
    }
    catch(IOException e)
    {
        System.out.println("I/O Error: " + e);
    }
    finally
    {
        try
        {
            if(fin != null)
                fin.close();
        }
    }

```

```

        catch(IOException e2)
        {
            System.out.println("Error Closing Input File");
        }
        try
        {
            if(fout != null)
                fout.close();
        }

        catch(IOException e2)
        {
            System.out.println("Error Closing Output File");
        }
    }
    System.out.println("\nFile Copied\n");
    System.out.println("\nDisplaying contents of "+ args[1]+" \n");
    try
    {
        fin = new FileInputStream(args[1]);
        do
        {
            i = fin.read();
            if(i != -1)
                System.out.print((char) i);
        } while(i != -1);
    }
    catch(IOException e)
    {
        System.out.println("Error Reading File");
    }
    finally
    {
        try
        {
            fin.close();
        }
        catch(IOException e)
        {
            System.out.println("Error Closing File");
        }
    }
}

```

OUTPUT:

```
R:\oop>javac CopyFile.java  
R:\oop>java CopyFile FIRST.txt SECOND.txt
```

Displaying contents of FIRST.txt

To use this program, specify the name of the source file and the destination file.
For example, to copy a file called FIRST.TXT to a file called SECOND.TXT, use the following command line.
java CopyFile FIRST.TXT SECOND.TXT

Copying contents of FIRST.txt to SECOND.txt

File Copied

Displaying contents of SECOND.txt

To use this program, specify the name of the source file and the destination file.
For example, to copy a file called FIRST.TXT to a file called SECOND.TXT, use the following command line.
java CopyFile FIRST.TXT SECOND.TXT

```
R:\oop>
```

RESULT:

Thus, the java program to copy the contents of one file to another file using file was written, executed and verified.

AIM:

To write a java program to find the maximum and minimum value from the given type of elements using ageneric function.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a class **Myclass** to implement generic class and generic methods.

Step 3: Get the set of the values belonging to specific data type.

Step 4: Create the objects of the class to hold integer, character and double values.

Step 5: Create the method to compare the values and find the maximum value stored in the array.

Step 6: Invoke the method with integer, character, double and string values. The output will be displayed based on the data type passed to the method.

Step 7: Stop the program.

PROGRAM:**GenericsDemo.java**

```
class MyClass<T extends Comparable<T>>
{
    T[] vals;
    MyClass(T[] obj)
    {
        vals = obj;
    }
    public T min()
    {
        T v = vals[0];
        for(int i=1; i<vals.length; i++)
            if(vals[i].compareTo(v) < 0)
                v = vals[i];
        return v;
    }
    public T max()
    {
        T v = vals[0];
        for(int i=1; i<vals.length; i++)
            if(vals[i].compareTo(v) > 0)
                v = vals[i];
        return v;
    }
}
```

```

    }
}
class GenericsDemo
{
    public static void main(String args[])
    {
        Integer num[]={10,2,5,4,6,1};
        Character ch[]={ 'v','p','s','a','n','h' };
        Double d[]={20.2,45.4,71.6,88.3,54.6,10.4};
        String str[]= {"hai","how","are","you"};
        MyClass<Integer>iob = new MyClass<Integer>(num);
        MyClass<Character> cob = new MyClass<Character>(ch);
        MyClass<Double>dob = new MyClass<Double>(d);
        MyClass<String>sob=new MyClass<String>(str);
        System.out.println("Max value in num: " + iob.max());
        System.out.println("Min value in num: " + iob.min());
        System.out.println("Max value in ch: " + cob.max());
        System.out.println("Min value in ch: " + cob.min());
        System.out.println("Max value in d: " + dob.max());
        System.out.println("Min value in d: " + dob.min());
        System.out.println("Max value in str: " + sob.max());
        System.out.println("Min value in str: " + sob.min());
    }
}

```

OUTPUT:

```

Max value in num: 10
Min value in num: 1
Max value in ch: v
Min value in ch: a
Max value in d: 88.3
Min value in d: 10.4
Max value in str: you
Min value in str: are

```

RESULT:

Thus, the Java program to find the maximum and minimum value from the given type of elements was implemented using generics and executed successfully.

AIM:

To write a program to display multiple choice test question using JavaFX.

ALGORITHM:

Step 1: Start the program

Step 2: Import the necessary packages

Step 3: Create a public class that extends the Application class.

Step 4: Override the start() method, which is found in the Application class.

Step 5: Create the button and name "Submit" on the button.

Step 6: Create radio buttons and link them all to the group question1.

Step 7: Disable the submit button by default initially.

Step 8: Add event handlers to all the radio buttons

Step 9: Stop the program.

PROGRAM:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
public class MCTest extends Application
{
    @Override
    public void start(Stage primaryStage)
    {
        primaryStage.setTitle("Test Question 1");
        Label labelfirst= new Label("What is 10 + 20?");
        Label labelresponse= new Label();
        Button button= new Button("Submit");
        RadioButton radio1, radio2, radio3, radio4;
        radio1=new RadioButton("10");
```

```

radio2= new RadioButton("20");
radio3= new RadioButton("30");
radio4= new RadioButton("40");

ToggleGroup question1= new ToggleGroup();
radio1.setToggleGroup(question1);
radio2.setToggleGroup(question1);
radio3.setToggleGroup(question1);
radio4.setToggleGroup(question1);

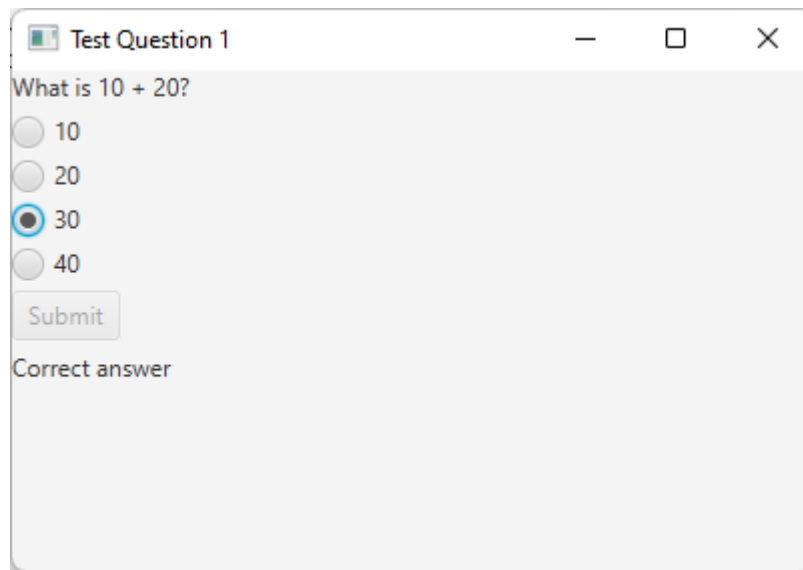
button.setDisable(true);
radio1.setOnAction(e ->button.setDisable(false) );
radio2.setOnAction(e ->button.setDisable(false) );
radio3.setOnAction(e ->button.setDisable(false) );
radio4.setOnAction(e ->button.setDisable(false) );

button.setOnAction(e ->
{
    if (radio3.isSelected())
    {
        labelresponse.setText("Correct answer");
        button.setDisable(true);
    }
    else
    {
        labelresponse.setText("Wrong answer");
        button.setDisable(true);
    }
});

VBox layout= new VBox(5);
layout.getChildren().addAll(labelfirst, radio1, radio2, radio3, radio4, button,
labelresponse);
Scene scene1= new Scene(layout, 400, 250);
primaryStage.setScene(scene1);
primaryStage.show();
}
public static void main(String[] args)
{
    launch(args);
}
}

```

OUTPUT:



RESULT:

Thus, the Java program for multiple choice test questions was implemented and executed successfully.

AIM:

To write a program to create simple editor with menu using JavaFX.

ALGORITHM:

Step 1: Start the program

Step 2: Import the necessary packages

Step 3: Create a public class that extends the Application class.

Step 4: Override the start() method, which is found in the Application class.

Step 5: Create the menu bar and add menu like File, Edit etc..

Step 6: In File menu add the menu items open, save and exit.

Step 7: In Edit menu add menu items like cut, copy and paste.

Step 8: Add event handlers to all menu items

Step 9: Create scene and add it to primary stage.

Step 10: Launch the application.

Step 11: Stop the program.

PROGRAM:

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.layout.VBox;
public class MenuUI extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception
    {
```

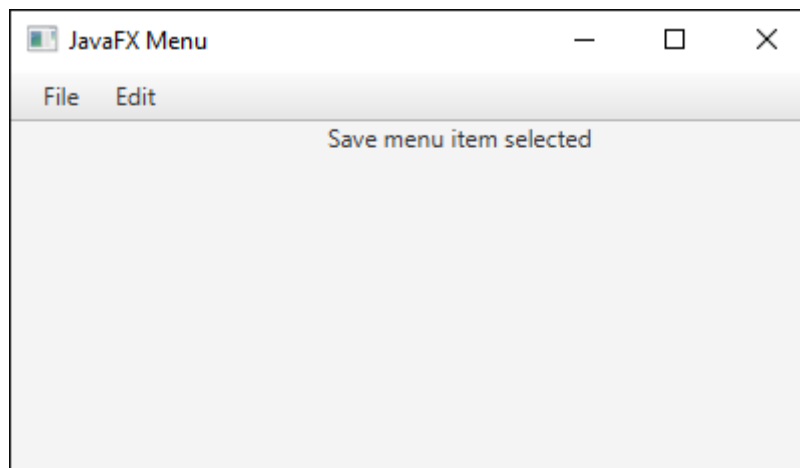
```

Menu newmenu = new Menu("File");
Menu newmenu2 = new Menu("Edit");
MenuItem m1 = new MenuItem("Open");
MenuItem m2 = new MenuItem("Save");
MenuItem m3 = new MenuItem("Exit");
MenuItem m4 = new MenuItem("Cut");
MenuItem m5 = new MenuItem("Copy");
MenuItem m6 = new MenuItem("Paste");
newmenu.getItems().add(m1);
newmenu.getItems().add(m2);
newmenu.getItems().add(m3);
newmenu2.getItems().add(m4);
newmenu2.getItems().add(m5);
newmenu2.getItems().add(m6);
MenuBar newmb = new MenuBar();
newmb.getMenus().add(newmenu);
newmb.getMenus().add(newmenu2);
Label l = new Label("\t\t\t\t\t" + "You have selected no menu items");
EventHandler<ActionEvent> event = new EventHandler<ActionEvent>() {
    public void handle(ActionEvent e)
    {
        l.setText("\t\t\t\t\t" + ((MenuItem)e.getSource()).getText() +
            " menu item selected");
    }
};
m1.setOnAction(event);
m2.setOnAction(event);
m3.setOnAction(event);
m4.setOnAction(event);
m5.setOnAction(event);
m6.setOnAction(event);
VBox box = new VBox(newmb,l);
Scene scene = new Scene(box,400,200);
primaryStage.setScene(scene);

```

```
primaryStage.setTitle("JavaFX Menu");  
primaryStage.show();  
}  
public static void main(String[] args) {  
    Application.launch(args);  
}  
}
```

OUTPUT:



RESULT:

Thus, the Java program for simple editor was implemented and executed successfully.

AIM

To perform string operations using Array List with the following functions for Append - add at end, Insert – add at particular index, Search and list all string starts with given letter.

ALGORITHM:

1. Create an ArrayList named obj.
2. Add elements to the existing ArrayList at a specific position.
3. Insert the String elements at last one by one and display the Arraylist obj elements.
4. To search a specific element, get the input of the element to search.
5. Get the element starting letter from the user and find the position then store in the new ArrayList
6. Display the new Arraylist.
7. Stop the program execution.

SOURCE CODE:

```
import java.util.*;
import java.util.ArrayList;
public class stringoperation
{
    public static void main(String args[])
    {
        int choice, i, size;
        Scanner sc=new Scanner(System.in);
        ArrayList<String> obj = new ArrayList<String>();
        System.out.println("Enter the total number of strings to be entered into the arraylist");
        size=sc.nextInt();
        for (int j=0;j<size;j++)
        {
            System.out.println("Enter the string"+j);
            obj.add(sc.nextLine());
        }
        System.out.println("Currently the array list has following elements:"+obj);
    }
}
```

```

System.out.println("Type of String Operation");

System.out.println("1. Append - add at end \n2. Insert – add at particular index \n3.
Search \n4. List all
string starts with given letter");

System.out.println("Select the Operation");

choice=sc.nextInt();

switch(choice)
{
    case 1 :
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter the String to Append at the end:");
        String append=scan.nextLine();
        obj.add(append);
        System.out.println("Current array list is:"+obj);
        break;
    case 2 :
        Scanner scan1=new Scanner(System.in);
        System.out.println("Enter the String to Insert at a particular index:");
        String b =scan1.nextLine();
        System.out.println("Enter the index:");
        i =sc.nextInt();
        obj.add(i, b);
        System.out.println("Current array list is:"+obj);
        break;
    case 3 :
        Scanner scan2=new Scanner(System.in);
        System.out.println("Enter the String to search:");
        String st1 =scan2.nextLine();
        if (obj.contains(st1))
        {
            System.out.println("Found the String");
        }
    }
}

```

```

    }
    else
    {
        System.out.println("There is no such String");
    }
    break;
case 4 :
    Scanner scan3=new Scanner(System.in);
    System.out.println("Enter the character to List all string starts with given letter");
    String startString =scan3.nextLine();
    for (String str : obj)
    {
        if(str.startsWith(startString))
            System.out.println(str);
    }
    System.out.println("Current array list is:"+obj);
}
}
}

```

INPUT AND OUTPUT:

Enter the total number of strings to be entered into the arraylist : 5

Enter the string1: Hari

Enter the string2: ram

Enter the string3: prakash

Enter the string4: sanjeve

Enter the string5: harray

Currently the array list has following elements:[, Hari, ram, prakash, sanjeve, harray]

Type of String Operation

1. Append - add at end

2. Insert – add at particular index

3. Search

4. List all string starts with given letter

Select the Operation:: 1

Enter the String to Append at the end:raj

Current array list is:[, Hari, ram, prakash, sanjeve, harray, raj]

RESULT

Thus the string operations using Array List with the following functions for Append - add at end, Insert – add at particular index, Search and list all string starts with given letter.

EX.NO.: 13	TOPIC BEYOND SYLLABUS Implementation of Packages
-------------------	---

AIM:

To implement packages by writing a java programme for currency converter (Dollar to INR, EURO to INR, Yen to INR and vice versa), distance converter (meter to KM, miles to KM and vice versa) , time converter (hours to minutes, seconds and vice versa) .

ALGORITHM:

1. Start the program
2. Create a class with the name Conversion
3. Create a Package with the name mypack1 and import the mypack1 in the conversion program
4. Create a class currencyconverter, distanceconverter, timeconverter in the package calledmypack1.
5. Call the currencyconverter, distanceconverter, timeconverter from the conversion class.
6. Get the details of the user for the corresponding conversion and print it.

SOURCE CODE:**Main Program**

```
import mypack1.*;
import java.util.Scanner;

class converter
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);
        System.out.println("1 Currency Converter");
        System.out.println("2 Distance Converter");
        System.out.println("3 Time Converter");
        System.out.println("3 Exit");
        System.out.print("Enter your choice: ");
        int choice=in.nextInt();
        switch(choice)
        {
```

```

case 1:
    currency obj1 = new currency();
    obj1.convertcurrency();
break;
case 2:
    distanceconverter obj2=new distanceconverter();
    obj2.convertDistance();
break;
case 3:
    timeconverter obj3=new timeconverter();
    obj3.converttime();
break;
case 4:
    System.exit(0);
break;
}
}
}

```

Currency Conversion

```

package mypack1;
import java.util.Scanner;
public class currency
{
    public void convertcurrency()
    {
        char us_dollar_sym = 36;
        char pound_sym = 163;
        char yen_sym = 165;
        char euro_sym = 8364;
        char inr_sym = 8377;
    }
}

```

```

String us_dollar = "Dollars";

String pound = "Pounds";

String yen = "Yen";

String euro = "Euros";

String rupee = "Rupees";

double rate = 0;


// Interface

System.out.println("Welcome to the A Cool Currency Converter \n");

System.out.println("Use the following codes to input your currency choices: \n 1 - US
dollars \n 2 -Euros \n 3 - British Pounds \n 4 - Japanese Yen \n 5 - Indian Rupee \n");

System.out.println("Please choose the input currency:");

Scanner in = new Scanner(System.in);

int choice = in.nextInt();

String inType = null;

switch (choice) {

    case 1:

        inType = "US Dollars >> " + us_dollar_sym;

        break;

    case 2:

        inType = "Euros >> " + euro_sym;

        break;

    case 3:

        inType = "British Pounds >> " + pound_sym;

        break;

    case 4:

        inType = "Japanese Yen >> " + yen_sym;

        break;

    case 5:

        inType = "Indian Rupees >> " + inr_sym;

        break;

```

```

default:

    System.out.println("Please restart the program & enter a number from the list.");

    return;

}

System.out.println("Please choose the output currency");

int output = in.nextInt();

System.out.printf("Now enter the input in " + inType);

double input = in.nextDouble();

if (choice == output)

    System.out.println("Same currency no need to convert");

if (choice == 1 && output == 2) {

    double dollar_euro_rate = 0.78391;

    rate = input * dollar_euro_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + dollar_euro_rate + "
Dollars to %s = %.2f\n", (char) us_dollar_sym, euro, rate);

} else if (choice == 1 && output == 3) {

    double dollar_pound_rate = 0.621484;

    rate = input * dollar_pound_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + dollar_pound_rate + "
Dollars to %s = %.2f\n", (char) us_dollar_sym, pound, rate);

} else if (choice == 1 && output == 4) {

    double dollar_yen_rate = 107.174;

    rate = input * dollar_yen_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + dollar_yen_rate + "
Dollars to %s = %.2f\n", (char) us_dollar_sym, yen, rate);

}

else if (choice == 1 && output == 5) {

    double dollar_rupee_rate = 67.061;

    rate = input * dollar_rupee_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + dollar_rupee_rate + "
Dollars to %s = %.2f\n", (char) us_dollar_sym, rupee, rate);

}

```

```

if (choice == 2 && output == 1) {
    double euro_dollar_rate = 1.27579;
    rate = input * euro_dollar_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + euro_dollar_rate + " Euros
to %s = %.2f\n", (char) euro_sym, us_dollar, rate);
} else if (choice == 2 && output == 3) {
    double euro_pound_rate = 0.792648;
    rate = input * euro_pound_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + euro_pound_rate + "
Euros to %s = %.2f\n", (char) euro_sym, pound, rate);
} else if (choice == 2 && output == 4) {
    double euro_yen_rate = 136.708;
    rate = input * euro_yen_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + euro_yen_rate + " Euros
to %s = %.2f\n", (char) euro_sym, yen, rate);
}
else if (choice == 2 && output == 5) {
    double euro_rupee_rate = 79.26;
    rate = input * euro_rupee_rate;

    System.out.printf("%s" + input + " at a conversion rate of " + euro_rupee_rate + "
Euros to %s = %.2f\n", (char) euro_sym, rupee, rate);
}

if (choice == 3 && output == 1) {
    double pound_dollar_rate = 1.60972;

    System.out.printf("%s" + input + " at a conversion rate of " + pound_dollar_rate + "
Pounds to %s = %.2f\n", (char) pound_sym, us_dollar, rate);
} else if (choice == 3 && output == 2) {
    double pound_euro_rate = 1.26161;

    System.out.printf("%s" + input + " at a conversion rate of " + pound_euro_rate + "
Pounds to %s = %.2f\n", (char) pound_sym, euro, rate);
} else if (choice == 3 && output == 4) {

```

```

        double pound_yen_rate = 172.511;

        System.out.printf("%s" + input + " at a conversion rate of " + pound_yen_rate + "
Pounds to %s = %.2f\n", (char) pound_sym, yen, rate);
    }

    else if (choice == 3 && output == 5) {

        double pound_rupee_rate = 90.231;

        System.out.printf("%s" + input + " at a conversion rate of " + pound_rupee_rate + "
Pounds to %s = %.2f\n", (char) pound_sym, rupee, rate);
    }

    if (choice == 4 && output == 1) {

        double yen_dollar_rate = 0.00932574;

        System.out.printf("%s" + input + " at a conversion rate of " + yen_dollar_rate + " Yen
to %s = %.2f\n", (char) yen_sym, us_dollar, rate);
    } else if (choice == 4 && output == 2) {

        double yen_euro_rate = 0.00730615;

        System.out.printf("%s" + input + " at a conversion rate of " + yen_euro_rate + " Yen to
%s = %.2f\n", (char) yen_sym, euro, rate);
    } else if (choice == 4 && output == 3) {

        double yen_pound_rate = 0.00579135;

        System.out.printf("%s" + input + " at a conversion rate of " + yen_pound_rate + " Yen
to %s = %.2f\n", (char) yen_sym, pound, rate);
    }

    else if (choice == 4 && output == 5) {

        double yen_rupee_rate = 0.61135;

        System.out.printf("%s" + input + " at a conversion rate of " + yen_rupee_rate + " Yen
to %s = %.2f\n", (char) yen_sym, rupee, rate);
    }

    if (choice == 5 && output == 1) {

        double inr_dollar_rate = 0.015555555555555;

        System.out.printf("%s" + input + " at a conversion rate of " + inr_dollar_rate + " Rupees
to %s = %.2f\n", (char) inr_sym, us_dollar, rate);
    } else if (choice == 5 && output == 2) {

        double inr_euro_rate = 0.0139386;

```

```

    System.out.printf("%s" + input + " at a conversion rate of "+ inr_euro_rate + " Rupees to %s
= %.2f\n", (char) inr_sym, euro, rate);

    } else if (choice == 5 && output == 3) {

        double inr_pound_rate = 0.0119942;

        System.out.printf("%s" + input + " at a conversion rate of "+ inr_pound_rate + " Rupees
to %s = %.2f\n", (char) inr_sym, pound, rate);

    } else if (choice == 5 && output == 4) {

        double inr_yen_rate = 1.72887;

        rate = input * inr_yen_rate;

        System.out.printf("%s" + input + " at a conversion rate of "+ inr_yen_rate + " Rupees
to %s = %.2f\n", (char) inr_sym, yen, rate);

    }

    System.out.println("Thanks for using the currency converter!");

}}

```

Distance Conversion

```

package mypack1;

import java.util.*;

public class distanceconverter
{
    public void convertDistance()
    {
        System.out.println("1 KM to Miles");
        System.out.println("2 Miles to KM");
        System.out.println("3 Meter to KM");
        System.out.println("4 KM to Meter");
        System.out.println("5 Exit");

        float meter, kilometer;

        Scanner input=new Scanner(System.in);

        System.out.print("Enter your choice: ");

        int choice=input.nextInt();
    }
}

```

```

switch(choice)
{
    case 1:
        int km[]=new int[1];
        double miles[]=new double[1];
        System.out.println("Enter distance in KM: ");
        for(int i=0;i<km.length;i++){
            km[i]=input.nextInt();
            miles[i] = km[i] * 0.621;
        }
        System.out.println("KMS \t Miles");
        for(int i=0;i<km.length;i++){
            System.out.println(km[i]+" \t "+miles[i]);
        }
        break;

    case 2:
        int mile[]=new int[1];
        double kms[]=new double[1];
        System.out.println("Enter distance in Miles: ");
        for(int i=0;i<mile.length;i++){
            mile[i]=input.nextInt();
            kms[i] = mile[i] / 0.621;
        }
        System.out.println("Miles \t KMS");
        for(int i=0;i<mile.length;i++){
            System.out.println(mile[i]+" \t "+kms[i]);
        }
        break;

    case 3:

```

```

        System.out.println("Enter meter:");

        Scanner input1 = new Scanner(System.in);

        meter = input1.nextFloat();

        kilometer = meter/1000;

        System.out.println("Length in kilometer = "+kilometer);

        break;

case 4:

    meter = 1000;

    System.out.println("Enter Kilometer: ");

    Scanner reader = new Scanner(System.in);

    kilometer = reader.nextFloat();

    double kmtometer=kilometer*meter;

    System.out.println( kilometer+ " kilometers is " +kmtometer+ " in meters");

    break;


case 5:

    System.exit(0);

    break;

}

}

}

```

Time Conversion

```

package mypack1;

import java.util.Scanner;

public class timeconverter
{
    public void convertime()
    {
        Scanner in = new Scanner(System.in);

        System.out.println("1 Seconds to Hours, Minutes, Seconds");
    }
}

```

```

System.out.println("2 Hours, Minutes, Seconds to seconds");
System.out.println("3 Exit");
System.out.print("Enter your choice: ");
int choice=in.nextInt();
switch(choice)
{
case 1:
    System.out.print("Enter seconds : ");
    int seconds = in.nextInt();
    int p1 = seconds % 60;
    int p2 = seconds / 60;
    int p3 = p2 % 60;
    p2 = p2 / 60;
    System.out.print("HH:MM:SS - " +p2 + ":" + p3 + ":" + p1);
    System.out.print("\n");
break;
case 2:
    System.out.print(" Enter the Hours Minutes and Seconds(HH:MM:SS) - " );
    String h=in.nextLine();
    String[] h1=h.split(":");
    int hour=Integer.parseInt(h1[0]);
    int minute=Integer.parseInt(h1[1]);
    int second=Integer.parseInt(h1[2]);
    int temp;
    temp = second + (60 * minute) + (3600 * hour);
    System.out.println("secondsss"+temp);
break;
case 3:
    System.exit(0);
break;

```

```
}  
}  
}
```

INPUT AND OUTPUT:

1 Currency Converter

2 Distance Converter

3 Time Converter

3 Exit

Enter your choice: 1

Welcome to the A Cool Currency Converter

Use the following codes to input your currency choices:

1 - US dollars

2 - Euros

3 - British Pounds

4 - Japanese Yen

5 - Indian Rupee

Please choose the input currency:

1

Please choose the output currency

5

Now enter the input in US Dollars >> \$1

\$1.0 at a conversion rate of 67.06 Dollars to Rupees = 67.06

Thanks for using the currency converter!

1 Currency Converter

2 Distance Converter

3 Time Converter

3 Exit

Enter your choice: 2

1 KM to Miles

2 Miles to KM

3 Meter to KM

4 KM to Meter

5 Exit

Enter your choice: 1

Enter distance in KM:

12

KMS	Miles
12	7.452

1 Currency Converter

2 Distance Converter

3 Time Converter

4 Exit

Enter your choice: 3

1 Seconds to Hours, Minutes, Seconds

2 Hours, Minutes, Seconds to seconds

3 Exit

Enter your choice: 1

Enter seconds : 12345

HH:MM:SS - 3:25:45

RESULT

Thus the packages were implemented in currency, distance, and time conversion java Program.

AIM

To develop a simple OPAC system for library using event-driven and concurrent programming paradigms of Java. Use JDBC to connect to a back-end database.

ALGORITHM:

1. Create a Database
2. Create a table employee in Database created.
3. Insert the fields into the employee table with the following fields: name, rollno, course, subject, marks.
4. Create another panel consisting of buttons UserID, BookID, Return/Renewal, Fine.
5. Associate these buttons with listeners(with Transaction Database).

SOURCE CODE:

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class JdbcOdbcExample
{
    public static void main(String args[])
    {
        Connection con = null;
        PreparedStatement stmt = null;
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:swing");

            String sql = "INSERT INTO employee (name, rollNo, course, subject, marks) VAL
UES" +
                "('Deepak', 10, 'MCA', 'Computer Science', 85)";
            stmt = con.prepareStatement(sql);
```

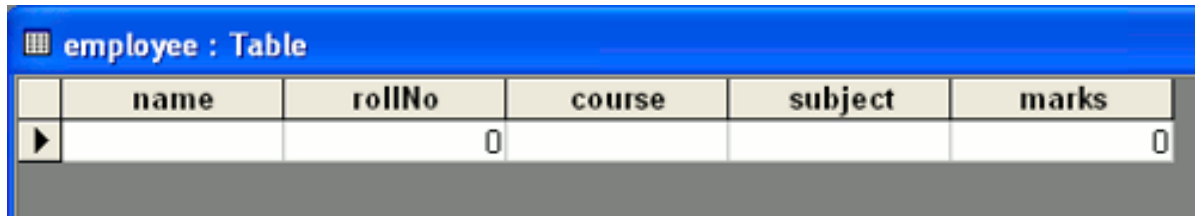
```

int i = stmt.executeUpdate();
if(i > 0 )
{
    System.out.println("Record Added Into Table Successfully.");
}
} catch (SQLException sqle) {
    System.out.println(sqle.getNextException());
} catch (ClassNotFoundException e) {
    System.out.println(e.getException());
} finally {
    try {
        if (stmt != null) {
            stmt.close();
            stmt = null;
        }
        if (con != null) {
            con.close();
            con = null;
        }
    } catch (Exception e) {
        System.out.println(e);
    }
}
}
}

```

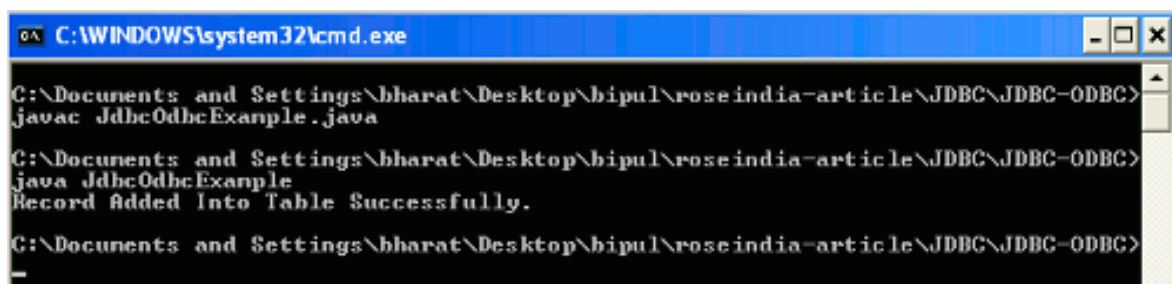
INPUT AND OUTPUT:

Table Before Inserting the record into it.



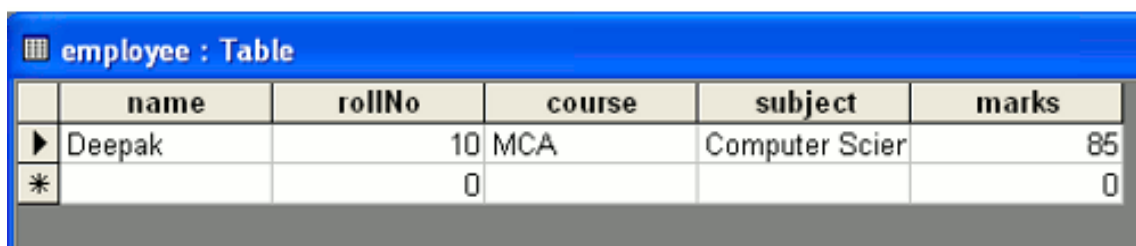
	name	rollNo	course	subject	marks
▶		0			0

When you will execute the above example you will get the output at console as follows :



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\bharat\Desktop\bipul\roseindia-article\JDBC\JDBC-ODBC>
javac JdbcOdbcExample.java
C:\Documents and Settings\bharat\Desktop\bipul\roseindia-article\JDBC\JDBC-ODBC>
java JdbcOdbcExample
Record Added Into Table Successfully.
C:\Documents and Settings\bharat\Desktop\bipul\roseindia-article\JDBC\JDBC-ODBC>
```

And then the Table after inserting record will be seen as follows :



	name	rollNo	course	subject	marks
▶	Deepak	10	MCA	Computer Scier	85
*		0			0

RESULT:

A simple OPAC system for library using event-driven and concurrent programming paradigms of Java with the use JDBC to connect to a back-end database was implemented.