SRM VALLIAMMAI ENGINEERINGCOLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur-603203

DEPARTMENT OF INFORMATION TECHNOLOGY

AND

COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR: 2025-2026

ODD SEMESTER

LAB MANUAL

(REGULATION - 2023)

CS3364- OBJECT ORIENTED PROGRAMMING LAB

THIRD SEMESTER

TABLE OF CONTENTS

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 and attitude for life-long learning process through the use of information technology sources.
5. To prepare then 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 theart methodologies based on ethical values.
2. Design and develop solutions for modern business environments coherent with the advanced technologiesand tools.
3. Design, plan and setting up the network that is helpful for contemporary business environments usinglatest hardware components.
4. Planning and defining test activities by preparing test cases that can predict and correct errors ensuring asocially transformed product catering all technological needs.

CS3364 OBJECT ORIENTED PROGRAMMING LABORATORY L T P C

0 0 3 1.5

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 Bulid 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.

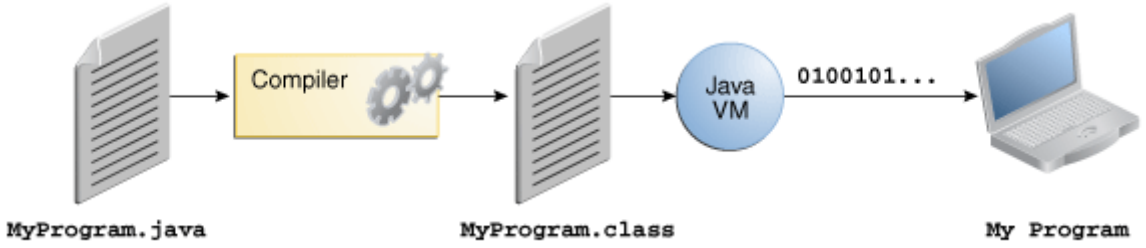
The latest Java version contains important enhancements to improve performance, stability and security of the Java applications that run on your machine. Installing this free update will ensure that your Java applications continue to run safely and efficiently.

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.

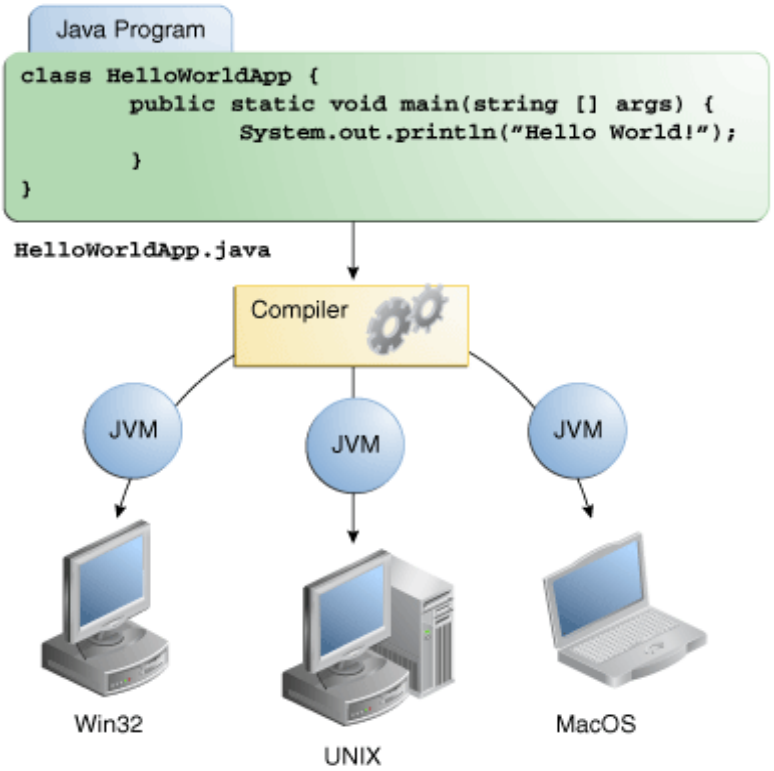
In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains *bytecodes* — 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

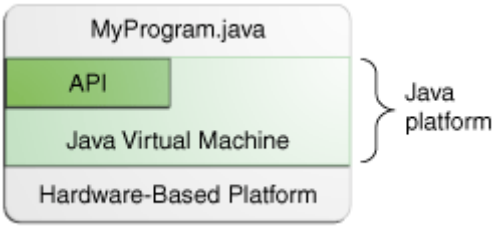
A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform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

EX.NO.:1(a)	SEQUENTIAL SEARCH
DATE:	

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.

EX.NO.:1(b)	BINARY SEARCH
DATE:	

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 intbinarySearch(intarr[], 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.

EX.NO.:1(c)	INSERTION SORT
DATE:	

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[])
	{
		intnum[]= {12,9,37,86,2,17,5};
		inti,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
DATE:	

AIM:

To develop a Java application to sort an array of elements in ascending order using selectin 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(inti=0;i<n-1;i++)
		{
			intmin=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.

EX.NO.: 2(a)	STACK DATA STRUCTURE
DATE:	

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 whetherstack 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 intmaxsize, top;
	private int[] st;
	public Stack(int size)
	{
		maxsize = size;
		st = new int[maxsize];
		top = -1;
	}
	booleanisEmpty()
	{
		return top==-1;
	}
	booleanisFull()
	{
		return top==maxsize-1;
	}
	public void push(int element)
	{
		 if(isFull())
			System.out.println("Overflow");
		else
			st[++top] = element;
	}

	public intpop()
	{
		 if(isEmpty())
		{
			System.out.println("UnderFlow");
		return (-1);
		}
		 return (st[top--]);
	}
	public void printStack()
	{
		System.out.println("Stack Elements:");
		for (inti = 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.

EX.NO.: 2(b)	QUEUE DATA STRUCTURE
DATE:	

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 whetherqueue 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: Incrementthe value of front by 1.

Step 6.5: Dequeue operation performed successfully.

Step 7: Displayoperation

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 intitems[];
	private intmaxsize, front, rear;
	Queue(int size)
	{
		maxsize=size;
		items = new int[size];
		front = -1;
		rear = -1;
	}
	booleanisFull()
	{
		if (front == 0 && rear ==maxsize-1)
		{
			return true;
		}
		return false;
	}
	booleanisEmpty()
	{
		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.

EX.NO.: 3	PAYSLIP GENERATION USING INHERITANCE
DATE:	

AIM:

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

ALGORITHM:

Step 1: Start the program.

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

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

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.** Reademployee 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
ProfessorPay 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 implementedusing inheritance and the program was executed successfully.

EX.NO.:4	ABSTRACT CLASS IMPLEMENTATION
DATE:	

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 methodprintArea().

Step 5: In printAree() 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
{
	inta,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("\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 wasdeveloped and executed successfully.

EX.NO.: 5	INTERFACE
DATE:	

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 methodnamed 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 printAree() method get the input from user and calculate the area of rectangle, circle and triangle.

Step 6: In the TestAreaclass , create the objects for the three classes and invoke the MethodprintArea() 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 interfacewasdeveloped and executed successfully.

EX.NO.:6	USER DEFINED EXCEPTION HANDLING
DATE:	

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 intaccountNumber;
	public BankAccount(intaccountNumber,doubleinitialBalance)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 intgetAccountNumber()
	{
		return accountNumber;
	}
	public String toString ()
	{
```

```
                return "Account Number :" + accountNumber + " Balance :" + balance;
        }
}
public class UserDefinedException
{
        public static void main(String[] args)
        {
                intch,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 implementedand executed successfully.

EX.NO.: 7	MULTITHREADING IMPLEMENTATION
DATE:	

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: Therun() 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 displaysthe square of the number. If the random number generated is odd, it displays the cube of thegiven 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 classMultiThread
{
        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.

EX.NO.: 8	FILE OPERATIONS
DATE:	

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;
		FileOutputStreamfout = 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.

EX.NO.: 9	GENERIC CLASS
DATE:	

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 displayedbased 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(inti=1; i<vals.length; i++)
			if(vals[i].compareTo(v) < 0)
				v = vals[i];
		return v;
	}
	public T max()
	{
		T v = vals[0];
		for(inti=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.

EX.NO.: 10(a)	MULTIPLE CHOICE TEST QUESTION IN JAVAFX
DATE:	

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

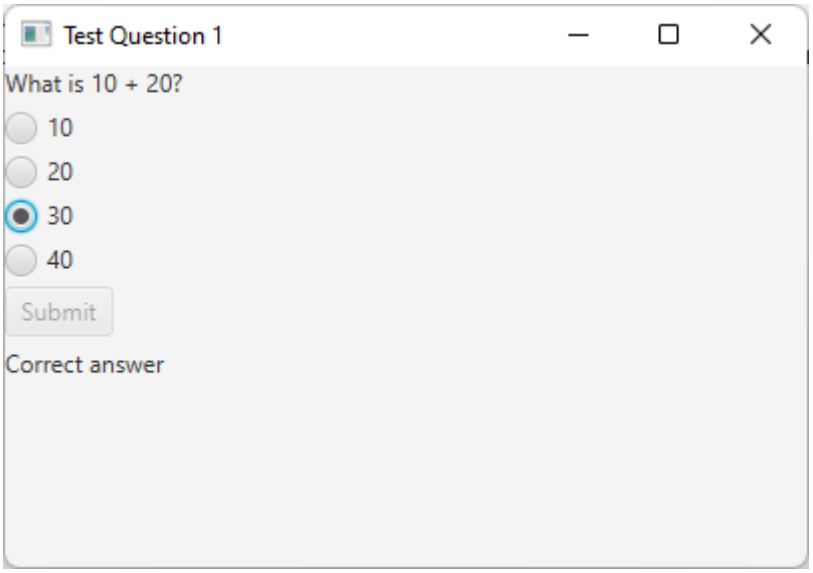
Step 9: Stop the program.

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.

EX.NO.: 10(b)	SIMPLE EDITOR USING JAVAFX
DATE:	

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 menubar 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.

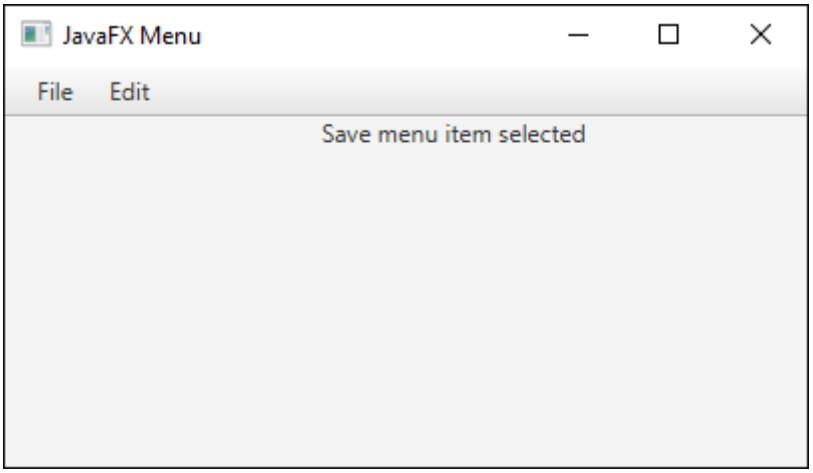
PROGRAM:

```
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\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\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.

EX.NO.:11	MINI PROJECT LIBRARY MANAGEMENT SYSTEM
DATE:	

AIM:

To write a program to create simple library management system that allows users to add, view, and borrow books.

ALGORITHM

Step 1: Start

Step 2: Initialize Library

Create an empty list to store books.

Initialize a counter for book IDs.

Step 3: Main Menu

Display options: Add Book, View Books, Borrow Book, Exit.

Step 4: Get User Choice

Read the user's menu choice.

Step 5: Perform Action Based on Choice

If Add Book:

Prompt for book title and author.

Create a new book with a unique ID and add it to the library.

If View Books:

Display all books in the library with their details.

If Borrow Book:

Prompt for book ID.

Mark the book as borrowed if it's available.

If Exit:

Exit the program.

Step 6: Stop

PROGRAM:

```
import java.util.ArrayList;
import java.util.Scanner;
class Book {
    int id;
    String title;
    String author;
    boolean isBorrowed;
    Book(int id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.isBorrowed = false;
    }
}
public class LibraryManagementSystem {
    public static void main(String[] args) {
        ArrayList<Book> library = new ArrayList<>();
```

```
        Scanner scanner = new Scanner(System.in);
        int bookIdCounter = 1;
        while (true) {
            System.out.println("Library Management System Menu:");
            System.out.println("1. Add Book");
            System.out.println("2. View Books");
            System.out.println("3. Borrow Book");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();  // Consume the newline character
            if (choice == 4) {
                System.out.println("Exiting the program...");
                break;
            }
            switch (choice) {
                case 1:
                    System.out.print("Enter the book title: ");
                    String title = scanner.nextLine();
                    System.out.print("Enter the book author: ");
                    String author = scanner.nextLine();
                    library.add(new Book(bookIdCounter++, title, author));
                    System.out.println("Book added.");
                    break;
                case 2:
                    System.out.println("Library Books:");
                    for (Book book : library) {
                        System.out.println("ID: " + book.id + ", Title: " + book.title + ", Author: " +
book.author + ", Borrowed: " + book.isBorrowed);
                    }
                    break;
                case 3:
                    System.out.print("Enter the book ID to borrow: ");
                    int bookId = scanner.nextInt();
                    boolean found = false;
                    for (Book book : library) {
                        if (book.id == bookId) {
                            found = true;
                            if (!book.isBorrowed) {
                                book.isBorrowed = true;
                                System.out.println("You have borrowed \"" + book.title + "\".");
                            } else {
                                System.out.println("Sorry, the book is already borrowed.");
                            }
                            break;
                        }
                    }
                    if (!found) {
                        System.out.println("Book ID not found.");
                    }
                    break;
```

```
                default:
                    System.out.println("Invalid choice. Please try again.");
                    break;
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

BUILDING AND RUNNING THE APPLICATION

1. Set Up the Environment:

Ensure you have the Java Development Kit (JDK) installed on your system.

2. Compile the Code:

Open a terminal or command prompt and navigate to the `src` directory of your project. Compile the Java code using the following command:

javac LibraryManagementSystem.java

3. Run the Application:

After compilation, run the application using the following command:

java LibraryManagementSystem

PACKAGING AS A JAR FILE

To package the application as an executable JAR file, follow these steps:

1. Compile the Code:

From the root directory of your project, compile the Java code and place the compiled classes in the `build` directory:

javac -d build src/LibraryManagementSystem.java

2. Create a Manifest File:

Create a file named `MANIFEST.MF` in the root directory with the following content:

Manifest-Version: 1.0

Main-Class: LibraryManagementSystem

3. Package the JAR:

Create the JAR file using the following command:

jar cfm LibraryManagementSystem.jar MANIFEST.MF -C build .

4. Run the JAR File:

Run the JAR file using the following command:

java -jar LibraryManagementSystem.jar

OUTPUT:

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 1
Enter the book title: Effective Java
Enter the book author: Joshua Bloch
Book added.

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 1
Enter the book title: Java: The Complete Reference
Enter the book author: Herbert Schildt
Book added.

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 1
Enter the book title: Head First Java
Enter the book author: Kathy Sierra, Bert Bates
Book added.

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 2
Library Books:
ID: 1, Title: Effective Java, Author: Joshua Bloch, Borrowed: false

ID: 2, Title: Java: The Complete Reference, Author: Herbert Schildt, Borrowed: false
ID: 3, Title: Head First Java, Author: Kathy Sierra, Bert Bates, Borrowed: false

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 3
Enter the book ID to borrow: 1
You have borrowed "Effective Java".

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 2
Library Books:
ID: 1, Title: Effective Java, Author: Joshua Bloch, Borrowed: true
ID: 2, Title: Java: The Complete Reference, Author: Herbert Schildt, Borrowed: false
ID: 3, Title: Head First Java, Author: Kathy Sierra, Bert Bates, Borrowed: false

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 3
Enter the book ID to borrow: 1
Sorry, the book is already borrowed.

Library Management System Menu:
1. Add Book
2. View Books
3. Borrow Book
4. Exit
Enter your choice: 4
Exiting the program...

RESULT:

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