CS3381 - OOPS Lab Manual

Object oriented programming (Anna University)

Ex.No: 1 a	**Java Program for Sequential Search (Linear Search)**
Date:	

Aim:

To create a Java program for Sequential search which is also called as Linear Search.

Algorithm:

Step 1 Start the process.
Step 2 Let the element to be search be **x**.
Step 3 Start from the leftmost element of **arr[]** and one by one compare **x** with each element of **arr[]**.
Step 4 If x matches with an element then return that **index**.
Step 5 If x doesn't match with any of elements then return -1.
Step 6 Exit from the process.
Step 7 Stop the process.

Coding:

```
class GFG {

        // Function for linear search
        public static int search(int arr[], int x)
        {
                int n = arr.length;

                for (int i = 0; i < n; i++) {


                                if (arr[i] == x)
                                return i;
                }
                return -1;
        }

        public static void main(String args[])
        {
                int arr[] = { 2, 3, 4, 10, 40 };

                int x = 10;

                // Function Call
                int result = search(arr, x);
                if (result == -1)
                        System.out.print(
                                "Element is not present in array");
                else



        }
}
```

```
System.out.print("
Element is
present"
                    +
                    index
                    +
```

Output:

Element is present at index 3

Result:

The java Program for Linear Search is developed and tested successfully.

Ex.No: 1 b	**Java Program for Binary Search**
Date:	

Aim:

To create a Java program for Binary search to find a given element.

Algorithm:

Step 1 Start the process.
Step 2 Compare x with the middle element.
Step 3 If x matches with middle element, we return the mid index.
Step 4 Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for right half.
Step 5 Else (x is smaller) recur for the left half.
Step 6 Exit from the process.
Step 7 Stop the process.

Coding:

```
class BinarySearch {
        int binarySearch(int arr[], int l,int r, int x)
        {
                if (r >= l) {
                        int mid = l + (r - l) /
                        2; if (arr[mid] == x)
                                return mid;

                        if (arr[mid] > x)
                                return binarySearch(arr, l,mid - 1, x);
                        // Else the element can only be
                        // present in right subarray
                        return binarySearch(arr, mid + 1,r, x);
                }

                return -1;
        }

        public static void main(String args[])
        {
                BinarySearch ob = new BinarySearch();
                // Given array arr[]
                int arr[] = { 2, 3, 4, 10, 40 };
                int n = arr.length;
                int x = 10;
                // Function Call
                int result = ob.binarySearch(arr, 0,n - 1, x);

                if (result == -1)
                        System.out.println("Element "+ "not present");

                else

        }
}
```

```
System.out.println("Ele
ment found"+ " at index
"+ result);
```

Output:

Element found at index 3

Result:

The java Program for Binary Search is developed and tested successfully.

Ex.No: 1 c	**Java Program for Selection Sort**
Date:	

Aim:

To create a Java program for Selection sort to sort the given elements.

Algorithm:

Step 1 Start the process.
Step 2 Initialize minimum value(min_idx) to location 0.
Step 3 Traverse the array to find the minimum element in the array.
Step 4 While traversing if any element smaller than min_idx is found then swap both the values.
Step 5 Then, increment min_idx to point to the next element.
Step 6 Repeat until the array is sorted.
Step 7 Stop the process.

Coding:

```
class SelectionSort
{
        void sort(int arr[])
        {
                int n = arr.length;
                for (int i = 0; i < n-1; i++)
                {
                        int min_idx = i;
                        for (int j = i+1; j < n; j++)
                                if (arr[j] < arr[min_idx])
                                        min_idx = j;
                        int temp = arr[min_idx];
                        arr[min_idx] = arr[i];
                        arr[i] = temp;
                }
        }

        void printArray(int arr[])
        {
                int n = arr.length;
                for (int i=0; i<n; ++i)
                        System.out.print(arr[i]+" ");
                System.out.println();
        }
        public static void main(String args[])
```

```
    {
        SelectionSort ob = new SelectionSort();
        int arr[] = {64,25,12,22,11};
        ob.sort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
```

Output:

Sorted array:

11 12 22 25 64

Result:

The java Program for Selection Sort is developed and tested successfully.

Ex.No: 1 d	**Java Program for Insertion Sort**
Date:	

Aim:

To create a Java program for Insertion sort to sort the given elements.

Algorithm:

Step 1 Start the process.

Step 2 Iterate from arr[1] to arr[N] over the array.

Step 3 Compare the current element (key) to its predecessor.

Step 4 If the key element is smaller than its predecessor, compare it to the elements before. **Step 5** Move the greater elements one position up to make space for the swapped element. **Step 6** Stop the process.

Coding:

```
class InsertionSort {   void sort(int arr[])
                        {
                                int n = arr.length;
                                for (int i = 1; i < n; ++i)
                                        { int key = arr[i];
                                        int j = i - 1;
                                        while (j >= 0 && arr[j] > key) {
                                                arr[j + 1] = arr[j];
                                                j = j - 1;
                                        }
                                        arr[j + 1] = key;
                                }
                        }

                        static void printArray(int arr[])
                        {
                                int n = arr.length;
                                for (int i = 0; i < n; ++i)
                                        System.out.print(arr[i] + " ");

                                System.out.println();
                        }

                        // Driver method
                        public static void main(String args[])
                        {
                                int arr[] = { 12, 11, 13, 5, 6 };

                                InsertionSort ob = new InsertionSort();
                                ob.sort(arr);

                                printArray(arr);
                        }
}
```

Output:

5 6 11 12 13

Result:

The java Program for Insertion Sort is developed and tested successfully.

Ex.No: 2a	**Java Application for Stack Data Structure**
Date:	

Aim:

To create a Java console application for stack. Stack operations must be controlled by exception handling techniques.

Algorithm:

Step 1: Start the program
Step 2: Initialize the necessary variables
Step 3: Create a stack, initialize the array and the stack variables
Step 4: Create the push function to insert the elements into the stack.
Step 5: The push function inserts element at the top and displays stack overflow, if the stack is full.
Step 6: Create the pop function to delete the elements from the stack.
Step 7: The pop function deletes the element from the top and displays stack empty, if the stack contains no element.
Step 8: Create isfull() and isempty() to check whether the stack is full or empty.
Step 9: In the main function, perform the operations to insert and delete the elements from the stack.
Step 10: Stop the program.

Coding

```
// Stack implementation in Java

class Stack {

  // store elements of stack
  private int arr[];
  // represent top of stack
  private int top;
  // total capacity of the stack
  private int capacity;

  // Creating a stack
  Stack(int size) {
    // initialize the array
    // initialize the stack variables
    arr = new int[size];
    capacity = size;
    top = -1;
  }
  // push elements to the top of stack
  public void push(int x) {
    if (isFull()) {
      System.out.println("Stack OverFlow");

      // terminates the program
      System.exit(1);
    }
```

```
    // insert element on top of stack
    System.out.println("Inserting " + x);
    arr[++top] = x;
  }
  // pop elements from top of stack
  public int pop() {
    // if stack is empty
    // no element to pop
    if (isEmpty()) {
      System.out.println("STACK EMPTY");
      // terminates the program
      System.exit(1);
    }

    // pop element from top of stack
    return arr[top--];
  }

  // return size of the stack
  public int getSize() {
  return top + 1;
  }

  // check if the stack is empty
  public Boolean isEmpty() {
  return top == -1;
  }
  // check if the stack is full
  public Boolean isFull() {
  return top == capacity - 1;
  }
  // display elements of stack
  public void printStack() {
    for (int i = 0; i <= top; i++) {
      System.out.print(arr[i] + ", ");
    }
  }
  public static void main(String[] args) {
    Stack stack = new Stack(5);
    stack.push(1);
    stack.push(2);
    stack.push(3);
    System.out.print("Stack: ");
    stack.printStack();
    // remove element from stack
    stack.pop();
    System.out.println("\nAfter popping out");
    stack.printStack();

  }
}
```

Output:

Inserting 1
Inserting 2
Inserting 3
Stack: 1, 2, 3,
After popping out
1, 2,

Result:

The java console application for stack data structure is developed and tested successfully.

Ex.No: 2b	**Java Application for Queue Data Structure**
Date:	

Aim:

To create a Java console application for Queue. Queue operations must be controlled by exception handling techniques.

Algorithm:

Step 1: Start the program.
Step 2: Initialize the necessary variables.
Step 3: Initialize the front and rear variables.
Step 4: Create the enqueue function to insert the elements into the queue.
Step 5: The enqueue function inserts element at the rear and displays queue is full, if the queue is full.
Step 6: Create the dequeue function to delete the elements from the queue.
Step 7: The dequeue function deletes the element from the rear and displays queue empty, if the queue contains no element.
Step 8: Create isfull() and isempty() to check whether the queue is full or empty.
Step 9: In the main function, perform the operations to insert and delete the elements from the queue.
Step 10: Stop the program.

Coding

```
public class Queue {
  int SIZE = 5;
  int items[] = new int[SIZE];
  int front, rear;

  Queue() {
    front = -1;
    rear = -1;
  }

  // check if the queue is full
  boolean isFull() {
    if (front == 0 && rear == SIZE - 1) {
      return true;
    }
    return false;
  }

  // check if the queue is empty
  boolean isEmpty() {
    if (front == -1)
    return true;
    else
      return false;
  }
```

```
// insert elements to the queue
void enQueue(int element) {

  // if queue is full
  if (isFull()) {
    System.out.println("Queue is full");
  }
  else {
    if (front == -1) {
      // mark front denote first element of queue
      front = 0;
    }

    rear++;
    // insert element at the rear
    items[rear] = element;
    System.out.println("Insert " + element);
  }
}

// delete element from the queue
int deQueue() {
  int element;

  // if queue is empty
  if (isEmpty()) {
    System.out.println("Queue is empty");
    return (-1);
  }
  else {
    // remove element from the front of queue
    element = items[front];

    // if the queue has only one element
    if (front >= rear) {
      front = -1;
      rear = -1;
    }
    else {
      // mark next element as the front
      front++;
    }
    System.out.println( element + " Deleted");
    return (element);
  }
}
// display element of the queue
void display() {
  int i;
  if (isEmpty()) {
    System.out.println("Empty Queue");
  }
  else {
```

```
    // display the front of the queue
    System.out.println("\nFront index-> " + front);

    // display element of the queue
    System.out.println("Items -> ");
    for (i = front; i <= rear; i++)
    System.out.print(items[i] + "  ");

    // display the rear of the queue
    System.out.println("\nRear index-> " + rear);
  }
}


public static void main(String[] args) {

  // create an object of Queue class
  Queue q = new Queue();

  // try to delete element from the queue
  // currently queue is empty
  // so deletion is not possible
  q.deQueue();

  // insert elements to the queue
  for(int i = 1; i < 6; i ++) {
  q.enQueue(i);
  }

  // 6th element can't be added to queue because queue is full
  q.enQueue(6);

  q.display();

  // deQueue removes element entered first i.e. 1
  q.deQueue();

  // Now we have just 4 elements
  q.display();

  }
}
```

Output:

```
Queue is empty
Insert 1
Insert 2
Insert 3
Insert 4
Insert 5
Queue is full

Front index-> 0
Items ->
1 2  3  4  5
Rear index-> 4
1 Deleted

Front index-> 1
Items ->
2  3  4  5
Rear index-> 4
```

Result:

The java console application for Queue data structure is developed and tested successfully.

Ex.No: 3	**Employee Payroll System**
Date:	

Aim:

To create a Java console application for employee payroll process management. This project includes Employee and they further classified as Programmer, Assistant Professor, Associate Professor, Professor.

Algorithm:

Step 1 Start the process

Step 2 Prompt the user with converter choice 1. Programmer 2. Assistant Professor 3. Associate Professor 4. Professor 5. Exit and get the choice.

Step 3 If user selects a Programmer then proceed to step 4

Step 4 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 5

Step 5 Proceed to Programmers Payment Processing

- **Step 5.1** Get the basic pay of Programmer
- **Step 5.2** If user enters -1 assume basic pay as 30000 and goto step 15
- **Step 5.3** Else directly go to step 15

Step 6 If user selects Assistant Professor step 7

Step 7 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 8

Step 8 Proceed to Assistant Professor Payment Processing

- **Step 8.1** Get the basic pay of Assistant Professor
- **Step 8.2** If user enters -1 assume basic pay as 25000 and goto step 15
- **Step 8.3** Else directly go to step 15

Step 9 If user selects Associate Professor step 10

Step 10 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 11

Step 11 Proceed to Associate Professor Payment Processing

- **Step 11.1** Get the basic pay of Associate Professor
- **Step 11.2** If user enters -1 assume basic pay as 40000 and goto step 15
- **Step 11.3** Else directly go to step 15

Step 12 If user selects Professor step 13

Step 13 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 14

Step 14 Proceed to Professor Payment Processing

- **Step 14.1** Get the basic pay of Professor
- **Step 14.2** If user enters -1 assume basic pay as 70000 and goto step 15
- **Step 14.3** Else directly go to step 15

Step 15 Compute Per_Day_Pay = original_basic_pay / no_of_days_in_the_current_month

Step 16 Get the number of days worked from user that include Cl, WH, FH and exclude the LOP

Step 17 Check no_days_worked <= no_of_days_in_the_current_month. Else display “Error Message” and goto step 18

Step 18 Compute Current_Basic_Pay = Per_Day_Pay * no_days_worked

Step 19 Compute Following and Store

DA = (Current_Basic_Pay/100) * 97
HRA = (Current_Basic_Pay/100) * 12
PF = (Current_Basic_Pay/100) * 0.1
GROSS_PAY = Current_Basic_Pay + DA + HRA + PF
NET_PAY = GROSS_PAY - PF

Step 17 Display Payment Details [Name, ID, Address, Mail ID, Mobile Number, BASIC PAY, DA, HRA,PF,GROSS_PAY, NET_PAY].

Step 18 Stop Processing

Coding

Employee.java

```
package payscale;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Scanner;

class Employee {
	String emp_name;
	String emp_id;
	String emp_address;
	String emp_mail_id;
	String emp_mobile_no;
	int basic_pay;
	int per_day_pay;
	int	current_basic_pay;
	int	da,	hra,	pf,
	gross_pay; int net_pay;
	int no_of_days_in_current_month;
	int no_of_days_worked;
	Calendar cal;
	Scanner input;

	Employee() {
		input = new Scanner(System.in);
		cal = new GregorianCalendar();
		no_of_days_in_current_month =
cal.getActualMaximum(Calendar.DAY_OF_MONTH);
		getUserBasicDetails();
	}

	public void generatePaySlip() {
		this.da = (this.current_basic_pay / 100) * 97;
		this.hra = (this.current_basic_pay / 100) * 12;
		this.pf = (int) ((this.current_basic_pay / 100) * 0.1);
		this.gross_pay = this.current_basic_pay + da + hra + pf;
```

```
		this.net_pay = gross_pay - pf;
	}

	public void displayPaySlip() {
		System.out.println("Name: " + this.emp_name);
		System.out.println("ID: " + this.emp_id);
		System.out.println("Address: " + this.emp_address);
		System.out.println("MailID: " + this.emp_mail_id);
		System.out.println("Mobile No: " + this.emp_mobile_no);
		System.out.println("\nEarnings");
		System.out.println("-------");
		System.out.println("BASIC Pay: " + current_basic_pay + "
		Rs"); System.out.println("DA : " + da + " Rs");
		System.out.println("HRA : " + hra + " Rs");
		System.out.println("\nDeductions");
		System.out.println("---------");
		System.out.println("PF : " + pf + " Rs");
		System.out.println("GROSS Pay: " + gross_pay + "
		Rs"); System.out.println("NET Pay: " + net_pay + "
		Rs");
	}

	public void getUserBasicDetails() {
		System.out.println("Enter Details");
		System.out.println("Name: ");
		this.emp_name = input.next();
		System.out.println("ID: ");
		this.emp_id = input.next();
		System.out.println("Address: ");
		this.emp_address = input.next();
		System.out.println("MailID: ");
		this.emp_mail_id = input.next();
		System.out.println("Mobile No:");
		this.emp_mobile_no =
		input.next();
	}

	public void computeCurrentBasicPay(String empType) { this.per_day_pay
		= this.basic_pay / no_of_days_in_current_month;
		System.out.println("\nBasic Pay of " + empType + " " + this.basic_pay + " for "
				+ this.no_of_days_in_current_month + " days");
		System.out.println("This month this " + empType + " gets " + this.per_day_pay + "
INR as basic pay per day");
		System.out.println("Enter no.of days worked by " + empType + " including CL,
WH, FH and excluding LWP:");
		this.no_of_days_worked = input.nextInt();
		if (no_of_days_worked <= no_of_days_in_current_month) {
			this.current_basic_pay = this.per_day_pay * no_of_days_worked;
			System.out.println("Programmer");
			System.out.println("---------");
			generatePaySlip();
		} else {
			System.out.println("Sorry Please Enter Valid Days");
		}
	}

	protected void finalize()
		{ input.close();
		System.exit(0);
	}
```

```
}
```

Programmer.java

```
package payscale;

public class Programmer extends Employee {

	public Programmer() {
		super();
		computeProgrammerPay();

	}

	public void computeProgrammerPay() {
		System.out.println("Enter Basic pay of Programmer [enter -1 for Default [BP =
30000]]:");	this.basic_pay = input.nextInt();
		if (this.basic_pay == -1) {
			this.basic_pay = 30000;
			System.out.println("Default Pay Taken");
		}
		computeCurrentBasicPay("Programmer");
		generatePaySlip();
		displayPaySlip();
	}

}
```

Assistant Professor.java

```
package payscale;

public class AssistantProfessor extends Employee {
	public AssistantProfessor() {
		super();
		computeAssistantProfessorPay();
	}

	public void computeAssistantProfessorPay() {
		System.out.println("Enter Basic pay of AssistantProfessor [enter -1 for Default [BP
= 25000]]:");	this.basic_pay = input.nextInt();
		if (this.basic_pay == -1) {
			this.basic_pay = 25000;
			System.out.println("Default Pay Taken");
		}
		computeCurrentBasicPay("AssistantProfessor");
		generatePaySlip();
		displayPaySlip();
	}
}
```

Associate Professor

```
package payscale;

public class AssociateProfessor extends Employee {
	public AssociateProfessor() {
		super();
		computeAssociateProfessorPay();
	}

	public void computeAssociateProfessorPay() {
		System.out.println("Enter Basic pay of AssociateProfessor [enter -1 for Default [BP
= 40000]]:");	this.basic_pay = input.nextInt();
		if (this.basic_pay == -1) {
			this.basic_pay = 40000;
			System.out.println("Default Pay Taken");
		}

		computeCurrentBasicPay("AssociateProfessor");
		generatePaySlip();
		displayPaySlip();
	}
}
```

Professor

```
package payscale;

public class Professor extends Employee
	{ public Professor() {
		super();
		computeProfessorPay();
	}

	public void computeProfessorPay() {
		 System.out.println("Enter Basic pay of Professor [enter -1 for Default [BP
=
70000]]:");	this.basic_pay = input.nextInt();
		if (this.basic_pay == -1) {
			this.basic_pay = 70000;
			System.out.println("Default Pay Taken");
		}
		computeCurrentBasicPay("Professor");
		generatePaySlip();
		displayPaySlip();
	}
}
```

Main.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

import payscale.AssistantProfessor;
import payscale.AssociateProfessor;
import payscale.Programmer;
import payscale.Professor;

public class Main {

	public static void main(String[] args) throws IOException {
		Programmer aProgrammer;
		AssistantProfessor aAssistantProfessor;
		AssociateProfessor aAssociateProfessor;
		Professor aProfessor;
		String  choice;
		int n_choice = 0;
		Scanner userInput = new Scanner("System.in");

		while (n_choice != 5) {
			System.out.println("\n\nEmployee Payroll System");
			System.out.println("**********************\n");
			System.out.println("1. Programmer\n2. Assistant Professor\n" + "3. Associate
Professor\n4. Professor\n"
						+ "5. Exit\n\nEnter Your Choice");
			choice = new BufferedReader(new
InputStreamReader(System.in)).readLine();
			n_choice = Integer.parseInt(choice);
			switch (n_choice) {
			case 1:
				System.out.println("Programmer Selected");
				aProgrammer = new Programmer();
				break;
			case 2:
				System.out.println("AssistantProfessor Selected");
				aAssistantProfessor = new AssistantProfessor();
				break;
			case 3:
				System.out.println("AssociateProfessor Selected");
				aAssociateProfessor = new AssociateProfessor();
				break;
			case 4:
				System.out.println("Professor Selected");
				aProfessor = new Professor();
			case 5:
				System.out.println("Thank You !!!");
				userInput.close();
				break;
			default:
				System.out.println("Enter valid choice !!!");
				break;
			}
		}
	}}
```

Output:

Choices

```
Employee Payroll System
***********************

1. Programmer
2. Assistant Professor
3. Associate Professor
4. Professor
5. Exit

Enter Your Choice
```

Basic Details

```
Enter Your Choice
1
Programmer Selected
Enter Details
Name:
Raja
ID:
987654
Address:
Coimbatore
MailID:
raja@gmail.com
Mobile No:
74874928921
Enter Basic pay of Programmer [enter -1  for Default [BP = 30000]]:
40000
```

Programmer

```
Basic Pay of Programmer 40000 for 30 days
This month this Programmer gets 1333 INR as basic pay per day
Enter no.of days worked by Programmer including CL, WH, FH and excluding LWP:
28
Programmer
----------
Name: Raja
ID: 987654
Address: Coimbatore
MailID: raja@gmail.com
Mobile No: 74874928921

Earnings
--------
BASIC Pay: 37324 Rs
DA : 36181 Rs
HRA : 4476 Rs

Deductions
----------
PF : 37 Rs
GROSS Pay: 78018 Rs
NET Pay: 77981 Rs
```

Assistant Professor

```
Enter Basic pay of AssistantProfessor [enter -1  for Default [BP = 25000]]:
-1
Default Pay Taken

Basic Pay of AssistantProfessor 25000 for 30 days
This month this AssistantProfessor gets 833 INR as basic pay per day
Enter no.of days worked by AssistantProfessor including CL, WH, FH and excluding LWP:
29
Pay Slip
--------
Name: Selva
ID: 099893
Address: Saravanapatti
MailID: selva@gmail.com
Mobile No: 989823892

Earnings
--------
BASIC Pay: 24157 Rs
DA : 23377 Rs
HRA : 2892 Rs

Deductions
----------
PF : 24 Rs
GROSS Pay: 50450 Rs
NET Pay: 50426 Rs
```

Associate Professor

```
Enter Basic pay of AssociateProfessor [enter -1  for Default [BP = 40000]]:
-1
Default Pay Taken

Basic Pay of AssociateProfessor 40000 for 30 days
This month this AssociateProfessor gets 1333 INR as basic pay per day
Enter no.of days worked by AssociateProfessor including CL, WH, FH and excluding LWP:
30
Pay Slip
--------
Name: Mani
ID: 983982
Address: Sulur
MailID: mani@gmail.com
Mobile No: 9389892208

Earnings
--------
BASIC Pay: 39990 Rs
DA : 38703 Rs
HRA : 4788 Rs

Deductions
----------
PF : 39 Rs
GROSS Pay: 83520 Rs
NET Pay: 83481 Rs
```

Professor

```
Enter Basic pay of Professor [enter -1  for Default [BP = 70000]]:
-1
Default Pay Taken

Basic Pay of Professor 70000 for 30 days
This month this Professor gets 2333 INR as basic pay per day
Enter no.of days worked by Professor including CL, WH, FH and excluding LWP:
27
Pay Slip
--------
Name: Anvar
ID: 847479
Address: Erode
MailID: anvar@gmail.com
Mobile No: 9379212080

Earnings
--------
BASIC Pay: 62991 Rs
DA : 61013 Rs
HRA : 7548 Rs

Deductions
----------
PF : 62 Rs
GROSS Pay: 131614 Rs
NET Pay: 131552 Rs
Thank You !!!
```

Result:

The java console application for employee payroll system was developed and tested successfully.

Ex.No: 4	**Java Application to Find the Area of different Shapes**
Date:	

Aim:

To create a Java console application to find the area of different shapes using abstract class concept in java.

Algorithm:

Step 1 Start the Process

Step 2 Prompt user with List Operation Choices
1. Rectangle 2. Triangle 3. Circle 4,
Exit Get the choice from user.

Step 3 If user selects Rectangle

Step 3.1 Get the Length and Breath from the

user Step 3.2 Compute Area = Length * Breath

Step 3.3 Display

Area Step 4 If user

selects Triangle

Step 3.1 Get the Length and Height from the

user Step 3.2 Compute Area = Length * Height

* 0.5 Step 3.3 Display Area

Step 5 If user selects Circle

Step 5.1 Get the Radius of the Circle

Step 5.2 Compute Area = 3.14 * Radius *

Radius Step 5.3 Display Area

Step 6 If user selects exit end the

process Step 7 Stop the Process

Coding:

Shape.java

```
package com.raja.oopslanb.shapes;

public abstract class Shape {
	double length = 0.0;
	double hight = 0.0;
	public abstract void printArea();
}
```

Rectangle.java

```
package com.raja.oopslanb.shapes;

import java.util.Scanner;

public class Rectangle extends Shape
	{ double area = 0.0;

	@Override
	public void printArea() {
		System.out.println("\nRectangle");
		System.out.println("--------\n");
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Length of Rectangle : ");
		this.length = input.nextDouble();
		System.out.println("Enter Breadth of Rectangle :
		"); this.hight = input.nextDouble();
		this.area = this.length * this.hight;
		System.out.println("Area of the Rectangle is : " + this.area);
	}

}
```

Triangle.java

```
package com.raja.oopslanb.shapes;

import java.util.Scanner;

public class Triangle extends Shape
	{ double area = 0.0;

	@Override
	public void printArea() {
		System.out.println("\nTriangle");
		System.out.println("--------\n");
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Length of Triangle : ");
		this.length = input.nextDouble();
		System.out.println("Enter Hight of Triangle : ");
		this.hight = input.nextDouble();
		this.area = 0.5 * this.length * this.hight;
		System.out.println("Area of the Triangle is : " + this.area);
	}
```

```
}
```

Circle.java

```
package com.raja.oopslanb.shapes;

import java.util.Scanner;

public class Circle extends Shape {
	double area = 0.0;

	@Override
	public void printArea() {
		System.out.println("\nCircle");
		System.out.println("------\n");
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Radius of Circle : ");
		this.length = input.nextDouble();
		this.area = Math.PI * this.length * this.length;
		System.out.println("Area of the Circle is : " + this.area);
	}

}
```

Main.java

```
import com.raja.oopslanb.shapes.Rectangle;
import com.raja.oopslanb.shapes.Shape;
import com.raja.oopslanb.shapes.Triangle;

import java.util.Scanner;

import com.raja.oopslanb.shapes.Circle;

public class Main {
	public static void main(String[] args) {
		Scanner userInput = new Scanner(System.in);
		int choice = 0;
		do {
			System.out.println("Finding Area");
			System.out.println("************");
			System.out.println(
				"\n1. Rectangle" + "\n2. Triangle" + "\n3. Circle" + "\n4. Exit"
+ "\n\nEnter your choice: ");
			choice = userInput.nextInt();
			switch (choice) {
			case 1:
				Shape rt = new
				Rectangle(); rt.printArea();
				break;
			case 2:
				Shape tr = new Triangle();
				tr.printArea();
				break;
			case 3:
				Shape cr = new Circle();
```

case 4:

```
c                                   intArea();
r                                   break;
.
p                                   System.out.println("\n\nThank You !!!");
r                                   userInput.close();
                                    break;
                            default:
                                    System.out.println("Please enter valid input");
                                    break;
                            }
                    } while (choice != 4);
            }
    }
```

Output:

Choice

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)
Finding Area
************

1. Rectangle
2. Triangle
3. Circle
4. Exit

Enter your choice:
```

Rectangle

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)

Enter your choice:
1

Rectangle
---------

Enter Length of Rectangle :
67
Enter Breadth of Rectangle :
78
Area of the Rectangle is : 5226.0
```

Triangle

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)

Triangle
---------

Enter Length of Triangle :
67
Enter Hight of Triangle :
78
Area of the Triangle is : 2613.0
```

Circle

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)

Circle
-------

Enter Radius of Circle :
89
Area of the Circle is : 24884.555409084755
```

Result:

The Java console application to find the area of different shapes using abstract class concept in java was developed and tested successfully.

Ex.No: 5	**Java Application to Find the Area of different Shapes - Interface**
Date:	

Aim:

To create a Java console application to find the area of different shapes using interface concept in java.

Algorithm:

Step 1 Start the Process

Step 2 Prompt user with List Operation Choices
1. Rectangle 2. Triangle 3. Circle 4, Exit
Get the choice from user.

Step 3 If user selects Rectangle

Step 3.1 Get the Length and Breath from the

user Step 3.2 Compute Area = Length * Breath

Step 3.3 Display

Area Step 4 If user

selects Triangle

Step 3.1 Get the Length and Height from the

user Step 3.2 Compute Area = Length * Height

* 0.5 Step 3.3 Display Area

Step 5 If user selects Circle

Step 5.1 Get the Radius of the Circle

Step 5.2 Compute Area = 3.14 * Radius *

Radius Step 5.3 Display Area

Step 6 If user selects exit end the

process Step 7 Stop the Process

Coding:

Shape.java

```
package com.raja.oopslanb.shapes;

interface Shape {
	double length = 0.0;
	double hight = 0.0;
	public void printArea();
}
```

Rectangle.java

```
package com.raja.oopslanb.shapes;

import java.util.Scanner;

public class Rectangle implements Shape
	{ double area = 0.0;

	@Override
	public void printArea() {
		System.out.println("\nRectangle");
		System.out.println("--------\n");
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Length of Rectangle : ");
		this.length = input.nextDouble();
		System.out.println("Enter Breadth of Rectangle :
		"); this.hight = input.nextDouble();
		this.area = this.length * this.hight;
		System.out.println("Area of the Rectangle is : " + this.area);
	}

}
```

Triangle.java

```
package com.raja.oopslanb.shapes;

import java.util.Scanner;

public class Triangle implements Shape {
	double area = 0.0;

	@Override
	public void printArea() {
		System.out.println("\nTriangle");
		System.out.println("--------\n");
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Length of Triangle : ");
		this.length = input.nextDouble();
		System.out.println("Enter Hight of Triangle : ");
		this.hight = input.nextDouble();
		this.area = 0.5 * this.length * this.hight;
		System.out.println("Area of the Triangle is : " + this.area);
	}
```

```
}
```

Circle.java

```
package com.raja.oopslanb.shapes;

import java.util.Scanner;

public class Circle implements Shape
	{ double area = 0.0;

	@Override
	public void printArea() {
		System.out.println("\nCircle");
		System.out.println("------\n");
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Radius of Circle : ");
		this.length = input.nextDouble();
		this.area = Math.PI * this.length * this.length;
		System.out.println("Area of the Circle is : " + this.area);
	}

}
```

Main.java

```
import com.raja.oopslanb.shapes.Rectangle;
import com.raja.oopslanb.shapes.Shape;
import com.raja.oopslanb.shapes.Triangle;

import java.util.Scanner;

import com.raja.oopslanb.shapes.Circle;

public class Main {
	public static void main(String[] args) {
		Scanner userInput = new Scanner(System.in);
		int choice = 0;
		do {
			System.out.println("Finding Area");
			System.out.println("************");
			System.out.println(
				"\n1. Rectangle" + "\n2. Triangle" + "\n3. Circle" + "\n4. Exit"
+ "\n\nEnter your choice: ");
			choice = userInput.nextInt();
			switch (choice) {
			case 1:
				Shape rt = new
				Rectangle(); rt.printArea();
				break;
			case 2:
				Shape tr = new Triangle();
				tr.printArea();
				break;
			case 3:
				Shape cr = new Circle();
```

case 4:

```
c                                   intArea();
r                                   break;
.
p                                   System.out.println("\n\nThank You !!!");
r                                   userInput.close();
                                    break;
                            default:
                                    System.out.println("Please enter valid input");
                                    break;
                            }
                    } while (choice != 4);
            }
    }
```

Output:

Choice

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)
Finding Area
************

1. Rectangle
2. Triangle
3. Circle
4. Exit

Enter your choice:
```

Rectangle

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)

Enter your choice:
1

Rectangle
---------

Enter Length of Rectangle :
67
Enter Breadth of Rectangle :
78
Area of the Rectangle is : 5226.0
```

Triangle

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)

Triangle
---------

Enter Length of Triangle :
67
Enter Hight of Triangle :
78
Area of the Triangle is : 2613.0
```

Circle

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (12) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 11:06:49 AM)

Circle
-------

Enter Radius of Circle :
89
Area of the Circle is : 24884.555409084755
```

Result:

The Java console application to find the area of different shapes using interface concept in java was developed and tested successfully.

Ex.No: 6	**Bank Transaction System with User Exceptions**
Date:	

Aim:

To create a Java console application for Banking transaction system that helps the users to do their credit and debit transactions and it rises user defined exception while encountering errors in transaction and also it should be solved using exception handling techniques.

Algorithm:

Step 1 Start the Process

Step 2 Prompt user with List Operation Choices
1. Add Money 2. Get Money 3. Details 4. Exit
Get the choice from user.

Step 3 If user selects Add Money

- **Step 3.1** Get the amount to be added to balance from the user
- **Step 3.2** If the amount is less than 0 then throw Invalid Credit Exception and goto step 3.4
- **Step 3.3** Else add amount to balance and goto step 2
- **Step 3.4** Prompt user with "Enter valid credit amount" and goto step 3.1

Step 4 If user selects Get Money

- **Step 4.1** Get the amount to be debited to balance from the user
- **Step 4.2** If the amount is greater than existing balance then throw Invalid Debit Exception and goto step 4.4
- **Step 4.3** Else deduct amount from balance and goto step 2
- **Step 4.4** Prompt user with "Enter valid debit amount" and goto step 4.1

Step 5 If user selects Details then display Customer Details [Name, Account Number, Current Balance]

Step 6 If user wants to exit display "Thank You !!!" and end process

Step 7 Stop the Process

Coding:

Customer.java

```
package com.raja.oopslab.exception.bank;

import java.util.Scanner;

public class Customer {
	String name;
	int accNo;
	int balance;

	public Customer(String name, int accNo) {
		this.name = name;
		this.accNo = accNo;
		this.balance = 0;
	}

	public void creditTransaction(int amount) {
		Scanner input = new
		Scanner(System.in); try {
			if (amount < 0)
				throw new InvalidCredit();

			else
				balance = balance + amount;
		} catch (InvalidCredit e) {
			amount =
			input.nextInt();
			creditTransaction(amount);
		}

	}

	public void debitTransaction(int amount) {
		Scanner input = new
		Scanner(System.in); try {
			if (amount > balance)
				throw new InvalidDebit();

			else
				balance = balance - amount;
		} catch (InvalidDebit e) {
			amount =
			input.nextInt();
			debitTransaction(amount);
		}

	}

	public void displayDetails(){ System.out.println("Customer
		Details");
		System.out.println("***************");
		System.out.println("Customer Name :
		"+this.name);
		System.out.println("Customer AccNo : "+this.accNo);
		System.out.println("Customer Current Balance : "+this.balance);
	}
```

}

InvalidCredit.java

```
package com.raja.oopslab.exception.bank;

public class InvalidCredit extends Exception

{
	public InvalidCredit() {
		System.out.print("Please enter valid credit amount");
	}
}
```

InvalidDebit.java

```
package com.raja.oopslab.exception.bank;

public class InvalidDebit extends Exception {
	public InvalidDebit() {
		System.out.print("Please enter valid debit amount");
	}
}
```

Main.java

```
import java.util.Scanner;
import com.raja.oopslab.exception.bank.*;
public class Main {
	public static void main(String[] args) {
		Scanner input = new
		Scanner(System.in); String name;
		int acc_no;
		System.out.println("Enter Customer Name");
		name = input.next();
		System.out.println("Enter account number");
		acc_no = input.nextInt();
		Customer aCustomer = new Customer(name, acc_no);
		int choice = 0;
		while(choice != 4){
			System.out.println("\n1. Add Money\n2. Get Money\n3. Details\n4. Exit");
			choice = input.nextInt();
			switch(choice){
			case 1:
				System.out.println("Enter the amount");
				aCustomer.creditTransaction(input.nextInt());
				break;
			case 2:
				System.out.println("Enter the amount");
				aCustomer.debitTransaction(input.nextInt());
				break;
			case 3:
				aCustomer.displayDetails();
				break;
			case 4:
				System.out.println("Thank You !!!");
				break;
			}
		}
	}
}
```

Output:

Choice:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (3) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 12:45:12 PM)
Enter Customer Name
Raja
Enter account number
98932891

1. Add Money
2. Get Money
3. Details
4. Exit
```

Display Details:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (3) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 12:46:52 PM)
Enter Customer Name
Raja
Enter account number
98932891

1. Add Money
2. Get Money
3. Details
4. Exit

Enter Your Choice
3
Customer Details
****************
Customer Name : Raja
Customer AccNo : 98932891
Customer Current Balance : 0

1. Add Money
2. Get Money
3. Details
4. Exit

Enter Your Choice
```

Add Money:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  Git Staging
Main (3) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 12:46:52 PM)

Enter Your Choice
1
Enter the amount
2000

1. Add Money
2. Get Money
3. Details
4. Exit

Enter Your Choice
3
Customer Details
****************
Customer Name : Raja
Customer AccNo : 98932891
Customer Current Balance : 2000
```

Get Money:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  G
Main (3) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 12:46:52 PM)

Enter Your Choice
2
Enter the amount
500

1. Add Money
2. Get Money
3. Details
4. Exit

Enter Your Choice
3
Customer Details
****************
Customer Name : Raja
Customer AccNo : 98932891
Customer Current Balance : 1500
```

Exception in Add Money:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  Git
Main (3) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 12:46:52 PM)

Enter Your Choice
1
Enter the amount
-500
Please enter valid credit amount100

1. Add Money
2. Get Money
3. Details
4. Exit

Enter Your Choice
3
Customer Details
****************
Customer Name : Raja
Customer AccNo : 98932891
Customer Current Balance : 1600
```

Exception in Get Money:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
Main (3) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 12:46:52 PM)

Enter Your Choice
2
Enter the amount
1800
Please enter valid debit amount1400

1. Add Money
2. Get Money
3. Details
4. Exit

Enter Your Choice
3
Customer Details
****************
Customer Name : Raja
Customer AccNo : 98932891
Customer Current Balance : 200
```

Result:

The java console application that uses user defined exception handling techniques was developed and tested successfully.

Ex.No: 7	**Java Application for Multi threading**
Date:	

Aim:

To create a Java console application the uses the multi threading concepts in java. The Application has 3 threads one creates random number, one thread computes square of that number and another one computes the cube of that number.

Algorithm:

Step 1 Start the Process

Step 2 Create a thread that generates random number

Step 3 Obtain one random number and check is odd or even

- **Step 3.1** If number is even then create and start thread that computes square of a number
- **Step 3.2** Compute number * number and display the answer
- **Step 3.3** Notify to Random number thread and goto step 4
- **Step 3.4** If number is odd then create and start thread that computes cube of a number
- **Step 3.4** Compute number * number * number and display the answer
- **Step 3.5** Notify to Random number thread and goto step 4

Step 4 Wait for 1 Second and Continue to Step 3 until user wants to exits.

Step 5 Stop the Process

Coding:

RandomNumberThread.java

```
package com.raja.oopslab.threading;

import java.util.Random;

public class RandomNumberThread extends Thread{
	Random num = new Random();
	int value;
	@Override
	public void run(){
		while(true){
			try {
				this.sleep(1000);
			} catch (InterruptedException e) {

			}
			value = num.nextInt(1000);
			System.out.println("RandomNumberThread generated a number "+value);
			if(value % 2 == 0)
				new SquareGenThread(value).start();
			else
				new CubeGenThread(value).start();
		}

	}
}
```

SquareGenThread.java

```
package com.raja.oopslab.threading;

public class SquareGenThread extends
	Thread{ int number;
	int squre;
	public SquareGenThread(int number) {
		this.number = number;
	}
	@Override
	public void run(){
		try {
			this.sleep(3000);
		} catch (InterruptedException e) {

		}
		this.squre = this.number * this.number;
		System.out.println("SquareGenThread--> Square of "+number+" is
		"+squre);
	 }
}
```

CubeGenThread.java

```
package com.raja.oopslab.threading;

public class CubeGenThread extends Thread{
	int number;
```

```
        int squre;
        public CubeGenThread(int number) {
                this.number = number;
        }
        @Override
        public void run(){
                try {
                        this.sleep(2000);
                } catch (InterruptedException e) {

                }
                this.squre = this.number * this.number * this.number;
                System.out.println("CubeGenThread--> Square of "+number+" is "+squre);
        }
}
```

Main.java

```
import java.util.Random;
import com.raja.oopslab.threading.RandomNumberThread;
public class Main {
        public static void main(String[] args) {
                new RandomNumberThread().start();
        }
}
```

Output:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  G
Main (5) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 3:15:18 PM)
RandomNumberThread generated a number 327
RandomNumberThread generated a number 63
RandomNumberThread generated a number 851
CubeGenThread--> Square of 327 is 34965783
CubeGenThread--> Square of 63 is 250047
RandomNumberThread generated a number 384
CubeGenThread--> Square of 851 is 616295051
RandomNumberThread generated a number 500
RandomNumberThread generated a number 903
SquareGenThread--> Square of 384 is 147456
RandomNumberThread generated a number 766
SquareGenThread--> Square of 500 is 250000
CubeGenThread--> Square of 903 is 736314327
RandomNumberThread generated a number 561
RandomNumberThread generated a number 854
SquareGenThread--> Square of 766 is 586756
CubeGenThread--> Square of 561 is 176558481
RandomNumberThread generated a number 801
RandomNumberThread generated a number 87
SquareGenThread--> Square of 854 is 729316
CubeGenThread--> Square of 801 is 513922401
```

Result:

The java console application for multithreading was developed and tested successfully.

Ex.No: 8	**Java Application for File Handling**
Date:	

Aim:

To create a Java console application to handle the files and find the file properties [Availability, Readable or Writeable or Both, Length of the File].

Algorithm:

Step 1 Start the Process

Step 2 Prompt the user to enter the file name with path

Step 3 Get the file name

- **Step 3.1** Check the file is exits
- **Step 3.2** If file exists then proceed to step 3.3 else proceed to step 3.8
- **Step 3.3** Check the File is Both Readable and Writeable
- **Step 3.4** If yes display file is “Read and Writeable”
- **Step 3.5** Else check is readable if yes display “Read Only” else move to step 3.6
- **Step 3.6** Else check is writeable if yes display “Write Only”
- **Step 3.7** Compute file size and display
- **Step 3.8** If file not existing then display “File Not Found”

Step 4 Stop the Process

Coding:

UserFileHandler.java

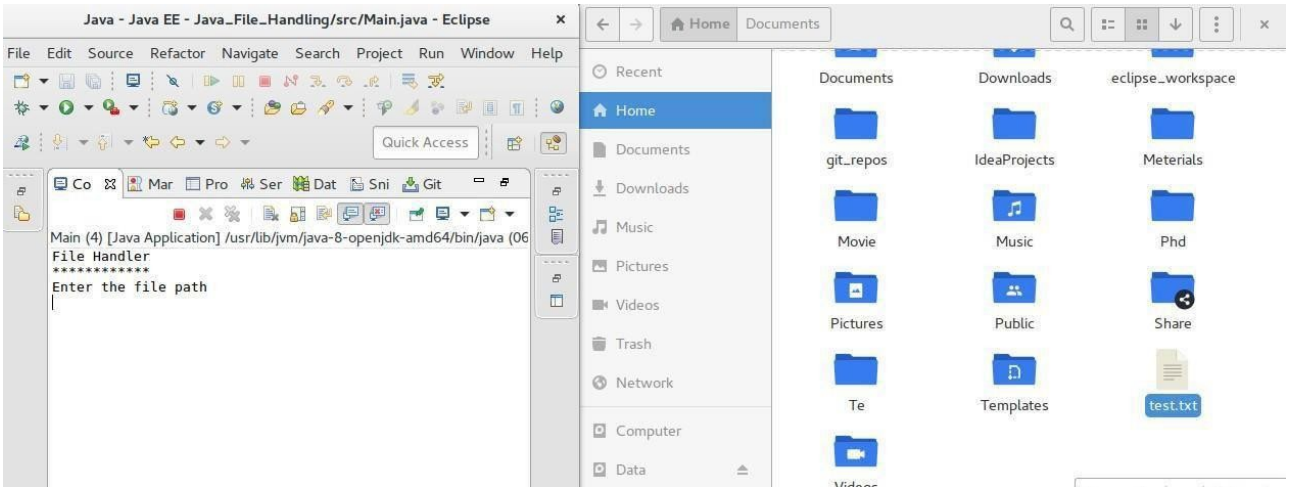
```
package com.raja.oopslab.files;

import java.io.File;
public class UserFileHandler{
	File aFile;
	boolean isReadable = false;
	boolean isWriteable = false;
	boolean isExists = false;
	int length = 0;
public UserFileHandler(String path) {
	aFile = new File(path);
	this.isExists = aFile.exists();
	this.isReadable = aFile.canRead();
	this.isWriteable =aFile.canWrite();
	this.length = (int) aFile.length();
}
public void fileDetails(){
	if(isExists){
		System.out.println("The File "+aFile.getName()+" is Available
at:"+aFile.getParent());
		if(isReadable && isWriteable)
			System.out.println("File is Readable and Writeable");
		else if(isReadable)
			System.out.println("File is Only Readable");
		else if(isWriteable)
			System.out.println("File is Only Writeable");
		System.out.println("Total length of the file is :"+this.length+" bytes");
	}
	else	System.out.println("File does not exists ");

}
}
```

Main.java

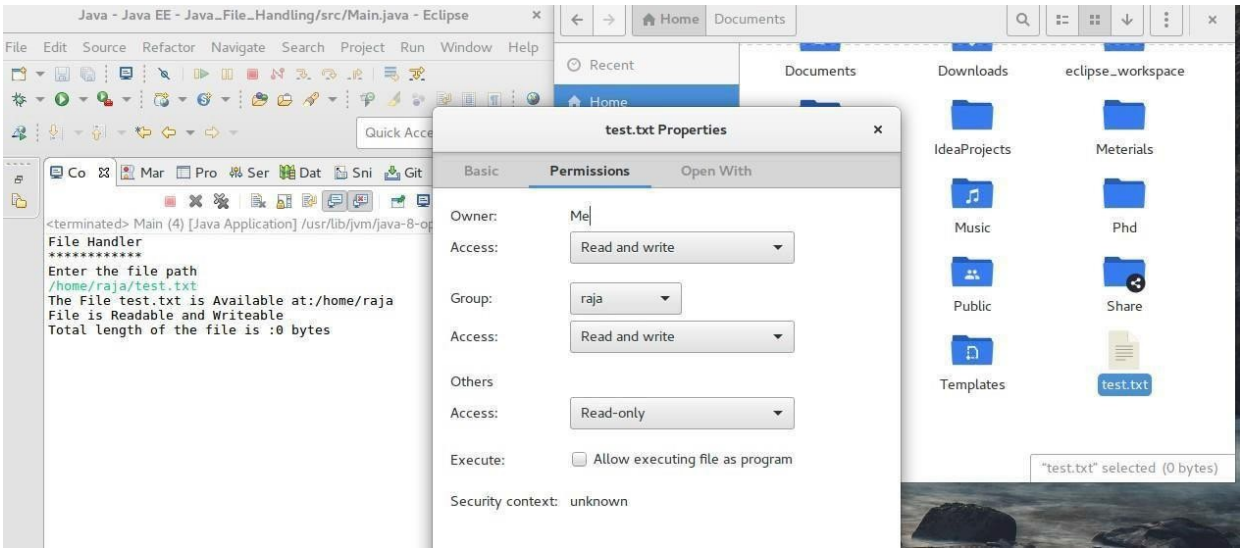
```
import java.io.File;
import java.util.Scanner;
import com.raja.oopslab.files.*;
public class Main {

	public static void main(String[] args)
		{ String file_path = null;
		Scanner input = new
		Scanner(System.in);
		System.out.println("File Handler");
		System.out.println("************");
		System.out.println("Enter the file path");
		file_path = input.next();
		new UserFileHandler(file_path).fileDetails();
	}
}
```

Output:

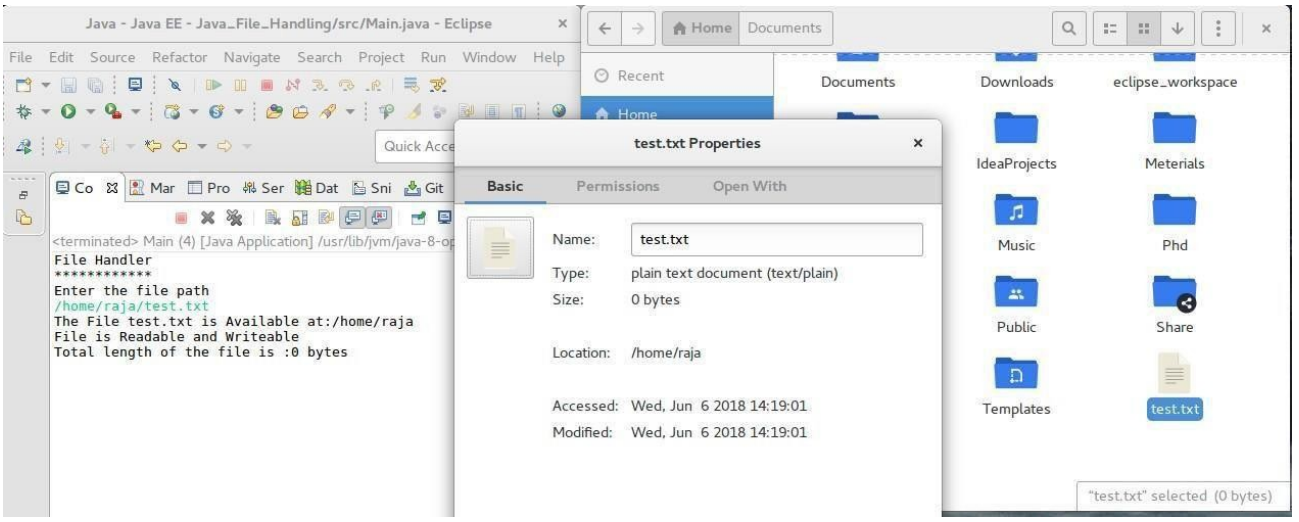
Availability of File:



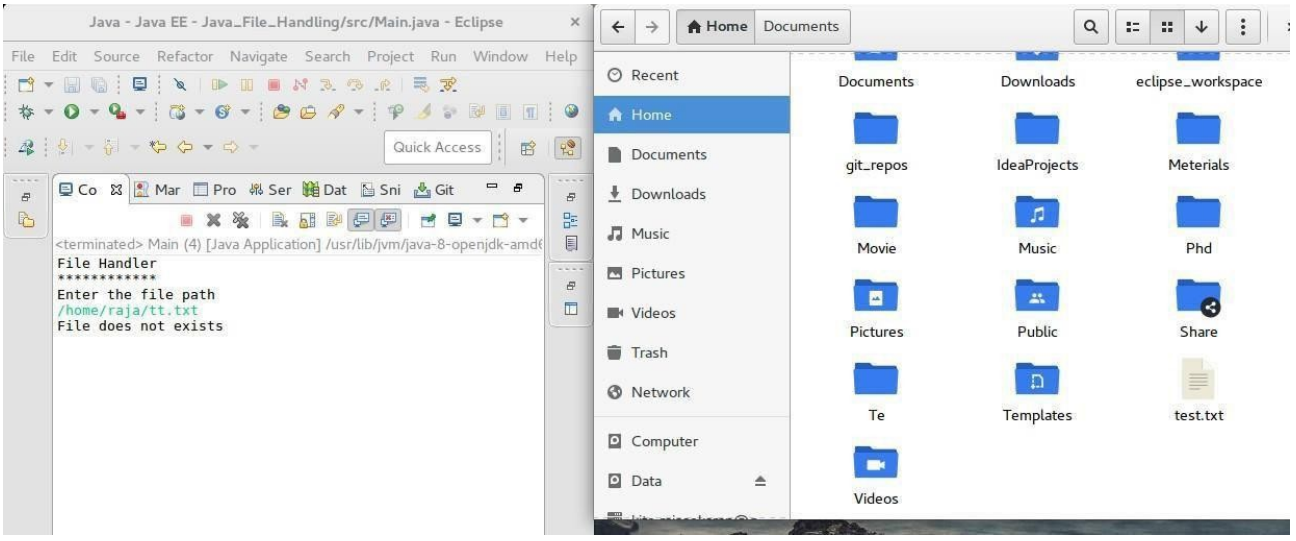
File Read and Writeable:



File Size:



File Not Exists:



Result:

The java console application for handling files was developed and tested successfully.

Ex.No: 9	**Java Application for Generic Max Finder**
Date:	

Aim:

To create a Java console application that finds the maximum in a array based on the type of the elements using generic functions in java.

Algorithm:

Step 1 Start the Process

Step 2 Create a array of number and array of strings and pass it to generic function.

Step 3 If the array is Integer type

- **Step 3.1** Assume first element as MAX
- **Step 3.2** Compare [Numeric Perspetive] this element with MAX
- **Step 3.3** If it is greater than MAX then store current element as MAX
- **Step 3.4** Else do nothing
- **Step 3.5** Goto step 3.1 until all the elements has been processed.

Step 4 If the array is String type

- **Step 4.1** Assume first element as MAX
- **Step 4.2** Compare [Dictionary Perspective] this element with MAX
- **Step 4.3** If it is greater than MAX then store current element as MAX **Step 4.4** Else do nothing
- **Step 4.5** Goto step 3.1 until all the elements has been processed.

Step 5 Stop the Process

Coding:

```
class GenericMax {
	public <T extends Comparable<T>> void maxFinder (T[] array){
		T max = array[0];
		for(T element: array){
			System.out.println(element);
			if(element.compareTo(max) > 0)
				max = element;
		}
		System.out.println("Max is : "+max);
	}
}

public class Main {

	public static void main(String[] args) {
		GenericMax max = new
		GenericMax(); Integer[] numbers =
		{14,3,42,5,6,10}; String[] strings =
		{"R","Ra","Raj"};
		max.maxFinder(numbers);
		max.maxFinder(strings);
	}

}
```

Output:

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets
<terminated> Main (6) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (06-Jun-2018, 3:41:49 PM)
14
3
42
5
6
10
Max is : 42
R
Ra
Raj
Max is : Raj
```

Result:

The java console application for finding generic max of given elements was developed and tested successfully.

Ex.No: 10	**Java applications using JavaFX controls, layouts and menus**
Date:	

Aim:

To create a Java application using JavaFX layouts and Menus to create a menu bar and add a menu to it and also add menu items to menu and also add an event listener to handle the events.

Algorithm:

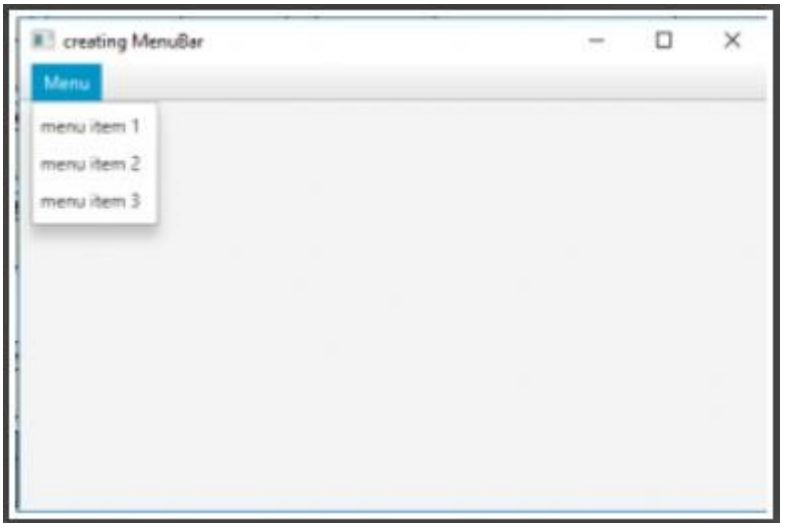
Step 1: Start the program
Step 2: Import the necessary javafx files.
Step 3: Create the class menubar and set the title for it.
Step 4: Create object for the class and create the necessary menu items in it.
Step 5: Add the menu items using the getitem() and add() functions.
Step 6: Create the label and events for the menu.
Step 7: Create the objects for the necessary classes.
Step 8: In the main function, launch the menu.
Step 9: Stop the program.

Coding:

```
// Java program to create a menu bar and add menu to
// it and also add menuitems to menu and also add
// an event listener to handle the events
import  javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.*;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.*;
import javafx.stage.Stage;
import javafx.scene.control.Alert.AlertType;
import java.time.LocalDate;
public class MenuBar_2 extends Application {

    // launch the application
    public void start(Stage s)
    {
        // set title for the stage
        s.setTitle("creating MenuBar");

        // create a menu
        Menu m = new Menu("Menu");
        // create menuitems
        MenuItem m1 = new MenuItem("menu item 1");
        MenuItem m2 = new MenuItem("menu item 2");
        MenuItem m3 = new MenuItem("menu item 3");
        // add menu items to menu
        m.getItems().add(m1);
        m.getItems().add(m2);
        m.getItems().add(m3);
```

```
                // label to display events
                Label l = new Label("\t\t\t\t"+ "no menu item selected");

                // create events for menu items
                // action event
                EventHandler<ActionEvent> event = new
EventHandler<ActionEvent>() {
                        public void handle(ActionEvent e)
                        {
                                l.setText("\t\t\t\t" +
((MenuItem)e.getSource()).getText() +" selected");
                        }
                };

                // add event
                m1.setOnAction(event);
                m2.setOnAction(event);
                m3.setOnAction(event);

                // create a menubar
                MenuBar mb = new MenuBar();

                // add menu to menubar
                mb.getMenus().add(m);

                // create a VBox
                VBox vb = new VBox(mb, l);

                // create a scene
                Scene sc = new Scene(vb, 500, 300);

                // set the scene
                s.setScene(sc);

                s.show();
        }

        public static void main(String args[])
        {
                // launch the application
                launch(args);
        }
}
```

Output:



Result:

The Java application using JavaFX layouts and Menus to create a menu bar and add a menu to it and also add menu items to menu was developed and tested successfully.