

A.V.C. COLLEGE OF ENGINEERING

Approved by AICTE & Affiliated to Anna University, Chennai

Re-accredited by NAAC with 'B++' Grade (2nd Cycle)

(An ISO 9001:2015 Certified Institution)

MAYILADUTHURAI, MANNAMPANDAL - 609 305



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Name	
Register Number	
Class / Semester	II CSE / III SEM
Subject Code & Title	CS3381 / OBJECT ORIENTED PROGRAMMING LABORATORY

CONTENTS

Name of the Experiment/Exercise	
1a	Sequential Search
1b	Binary Search
1c	Selection Sort
1d	Insertion Sort
2a	Stack Operations
2b	Queue Operations
3	Payroll Processing
4	Abstract class
5	Interface
6	Exception handling
7	Multithreading
8	File operations
9	Generic classes
10	Application using JavaFX layout, controls and menus
11	Mini project

Ex 1a- Sequential Search

Aim:

To write a java program to search an element in a given set of numbers using sequential searching method.

Procedure:

1. Start the program
2. Initialize an array, arr, with a set of numbers
3. Read the element to be searched, searchElement
4. Invoke the method linearSearch() by passing two arguments, the array and the element to be searched.
5. If the index returned by the method is other than -1, then print the element is found at the given index, otherwise, print the element is not found.
6. Stop the program

Method - linearSearch()

1. Loop through the array
2. Check whether the search element matches with any of the array elements. If a match found, then return the corresponding position of the element, otherwise return -1.
3. Return to the calling method.

Program

```
import java.util.*;
public class LinearSearch {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = {12, 23, 10, 34, 55, 4, 68, 3, 73, 99};
        System.out.println("Enter value to search: ");
        int searchElement = sc.nextInt();
        int index = linearSearch(arr, searchElement);
        if(index != -1){
            System.out.println("Searched item " + arr[index] + " found at index "+index);
        }else{
            System.out.println("Searched item " + searchElement + " not found in the array");
        }
    }

    private static int linearSearch(int[] arr, int searchElement){
        for(int i = 0; i < arr.length; i++){
            if(arr[i] == searchElement){
                return i;
            }
        }
        return -1;
    }
}
```

Sample output

```
D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java LinearSearch
Enter value to search:
22
Searched item 22 not found in the array

D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java LinearSearch
Enter value to search:
23
Searched item 23 found at index 1
```

Result:

Thus the sequential searching of a given element is successfully implemented.

Ex 1b- Binary Search

Aim:

To write a java program to search an element in a given set of numbers using binary searching method.

Procedure:

1. Start the program
2. Initialize an array, arr, with a set of numbers
3. Read the element to be searched, searchElement
4. Invoke the method binarySearch() by passing four arguments, the array and the element to be searched, starting index and ending index.
5. If the index returned by the method is other than -1, then print the element is found at the given index, otherwise, print the element is not found.
6. Stop the program

Method - binarySearch()

1. Find the middle position of the array, $mid = (low + high) / 2$.
2. Check if the search element is present in the middle of the array, if yes, then, return the middle index.
3. Check if the search element is less than the middle element, continue the search in the left half of the array.
4. If the search element is greater than the middle element, continue the search in the right half of the array.
5. If the element is found then return the index, otherwise, return -1.
6. Return to the calling method.

Program

```
import java.util.Scanner;
```

```
// Binary Search in Java
```

```
class Binarysearch {  
    int binarySearch(int array[], int element, int low, int high) {  
  
        // Repeat until the pointers low and high meet each other  
        while (low <= high) {  
  
            // get index of mid element  
            int mid = low + (high - low) / 2;
```

```

    // if element to be searched is the mid element
    if (array[mid] == element)
        return mid;

    // if element is less than mid element
    // search only the left side of mid
    if (array[mid] < element)
        low = mid + 1;

    // if element is greater than mid element
    // search only the right side of mid
    else
        high = mid - 1;
}

return -1;
}

public static void main(String args[]) {

    // create an object of Main class
    Binarysearch obj = new Binarysearch();

    // create a sorted array
    int[] array = { 3, 4, 5, 6, 7, 8, 9 };
    int n = array.length;

    // get input from user for element to be searched
    Scanner input = new Scanner(System.in);

    System.out.println("Enter element to be searched:");

    // element to be searched
    int element = input.nextInt();
    input.close();

    // call the binary search method
    // pass arguments: array, element, index of first and last element
    int result = obj.binarySearch(array, element, 0, n - 1);
    if (result == -1)
        System.out.println("Not found");
    else
        System.out.println("Element found at index " + result);
}}

```

Sample output

```
D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java Binarysearch
Enter element to be searched:
1
Not found

D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java Binarysearch
Enter element to be searched:
6
Element found at index 3
```

Result:

Thus the binary search of a given element is successfully implemented.

Ex 1c- Selection sort

Aim:

To write a java program to sort a given set of numbers using selection sort method.

Procedure:

1. Start the program
2. Initialize an array, arr1, with a set of numbers
3. Print the elements of the array before sorting
4. Invoke the method selectionSort() by passing the array as its argument.
5. Print the elements of the array after sorting.
6. Stop the program

Method - selectionSort()

1. Set smallerNumber to element in the lowest index
2. Search the minimum element in the array.
3. Swap the first location with the minimum value in the array.
4. Assign the second element as min.
5. Repeat the process until we get a sorted array.
6. Return to the calling method.

Program

```
public class SelectionSort {
    public static void selectionSort(int[] arr){
        for (int i = 0; i < arr.length - 1; i++)
        {
            int index = i;
            for (int j = i + 1; j < arr.length; j++){
                if (arr[j] < arr[index]){
                    index = j;//searching for lowest index
                }
            }
            int smallerNumber = arr[index];
            arr[index] = arr[i];
            arr[i] = smallerNumber;
        }
    }

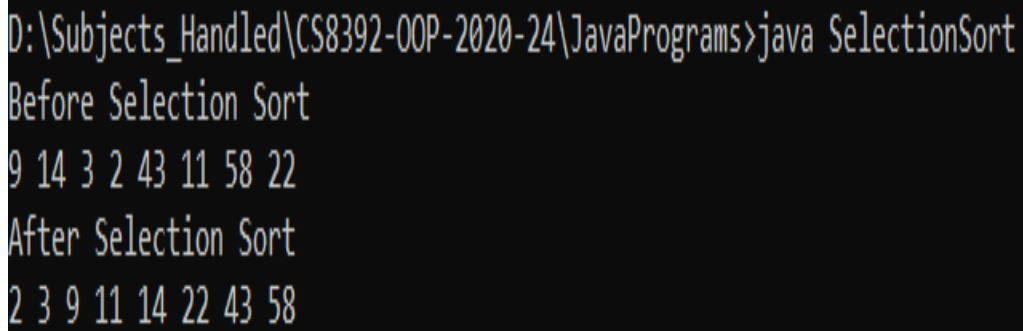
    public static void main(String a[]){
        int[] arr1 = {9,14,3,2,43,11,58,22};
        System.out.println("Before Selection Sort");
        for(int i:arr1){
            System.out.print(i+" ");
        }
        System.out.println();

        selectionSort(arr1);//sorting array using selection sort
    }
}
```



```
        System.out.println("After Selection Sort");
        for(int i:arr1){
            System.out.print(i+" ");
        }
    }
}
```

Sample output



```
D:\Subjects_Handled\CS8392-OOP-2020-24\JavaPrograms>java SelectionSort
Before Selection Sort
9 14 3 2 43 11 58 22
After Selection Sort
2 3 9 11 14 22 43 58
```

Result

Thus the selection sort program to sort a given set of numbers was executed successfully.

Ex 1d- Insertion sort

Aim:

To write a java program to sort a given set of numbers using insertion sort method.

Procedure:

1. Start the program
2. Initialize an array, arr1, with a set of numbers
3. Print the elements of the array before sorting
4. Invoke the method insertionSort() by passing the array as its argument.
5. Print the elements of the array after sorting.
6. Stop the program

Method - insertionSort()

1. If the element is the first one, it is already sorted.
2. Move to the next element
3. Compare the current element with all elements in the sorted array.
4. If the element in the sorted array is smaller than the current element, iterate to the next element. Otherwise, shift all the greater element in the array by one position towards the right
5. Insert the value at the correct position
6. Repeat until the complete list is sorted.
7. Return to the calling method.

Program

```
public class InsertionSort {
    public static void insertionSort(int array[]) {
        int n = array.length;
        for (int j = 1; j < n; j++) {
            int key = array[j];
            int i = j-1;
            while ( (i > -1) && ( array [i] > key ) ) {
                array [i+1] = array [i];
                i--;
            }
            array[i+1] = key;
        }
    }

    public static void main(String a[]){
        int[] arr1 = {12,11,10,15,14,13};
        System.out.println("Before Insertion Sort");
        for(int i:arr1){
            System.out.print(i+" ");
        }
    }
}
```

```

    }
    System.out.println();

    insertionSort(arr1);//sorting array using insertion sort

    System.out.println("After Insertion Sort");
    for(int i:arr1){
        System.out.print(i+" ");
    }
}
}

```

Sample output

```

D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java InsertionSort
Before Insertion Sort
12 11 10 15 14 13
After Insertion Sort
10 11 12 13 14 15

```

Result

Thus the insertion sort program to sort a given set of numbers was executed successfully.

Ex 2a- Stack using classes

Aim:

To write a java program to implement the operations of stack data structure.

Procedure:

1. Start the program
2. Create an instance of Stack class
3. Invoke the method pushelmnt() to push the elements into the stack one by one.
4. Invoke the method popelmnt() to remove an element from the stack
5. Print the elements of the stack after every push and pop operation
6. Stop the program

Program

```
import java.util.*;
public class StackDemo
{
    public static void main(String args[])
    {
        //creating an object of Stack class
        Stack <Integer> stk = new Stack<Integer>();
        System.out.println("stack: " + stk);
        //pushing elements into the stack
        pushelmnt(stk, 20);
        pushelmnt(stk, 13);
        pushelmnt(stk, 89);
        pushelmnt(stk, 90);
        pushelmnt(stk, 11);
        pushelmnt(stk, 45);
        pushelmnt(stk, 18);
        //popping elements from the stack
        popelmnt(stk);
        popelmnt(stk);
        //throws exception if the stack is empty
        try
        {
            popelmnt(stk);
        }
        catch (EmptyStackException e)
        {
            //exception handling
        }
    }
}
```

```

System.out.println("empty stack");
}
}
//performing push operation
static void pushelmnt(Stack stk, int x)
{
//invoking push() method
stk.push(new Integer(x));
System.out.println("push -> " + x);
//prints modified stack
System.out.println("stack: " + stk);
}
//performing pop operation
static void popelmnt(Stack stk)
{
System.out.print("pop -> ");
//invoking pop() method
Integer x = (Integer) stk.pop();
System.out.println(x);
//prints modified stack
System.out.println("stack: " + stk);
}
}

```

Sample output

```

D:\Subjects_Handled\CS8392-OOP-2020-24\JavaPrograms>java StackDemo
stack: []
push -> 20
stack: [20]
push -> 13
stack: [20, 13]
push -> 89
stack: [20, 13, 89]
push -> 90
stack: [20, 13, 89, 90]
push -> 11
stack: [20, 13, 89, 90, 11]
push -> 45
stack: [20, 13, 89, 90, 11, 45]
push -> 18
stack: [20, 13, 89, 90, 11, 45, 18]
pop -> 18
stack: [20, 13, 89, 90, 11, 45]
pop -> 45
stack: [20, 13, 89, 90, 11]
pop -> 11
stack: [20, 13, 89, 90]

```

Result

Thus the java program to implement stack operations was successfully executed.

Ex 2b- Queue using classes

Aim:

To write a java program to implement the operations of queue data structure.

Procedure:

1. Start the program
2. Create an instance of Queue class
3. Invoke the method add() to insert the elements into the queue one by one.
4. Print the elements of the queue after inserting all the elements
5. Invoke the method remove() to delete an element from the queue
6. Print the elements of the queue after the remove operation.
7. Print the head element of the queue by calling the element() method.
8. Invoke the poll() method to remove and return the head of the queue.
9. Invoke the peek() method to display the head of the queue.
10. Stop the program

Program

```
import java.util.*;
public class QueueTest {
    public static void main(String[] args) {
        Queue<Integer> q1 = new LinkedList<Integer>();
        //Add elements to the Queue
        q1.add(10);
        q1.add(20);
        q1.add(30);
        q1.add(40);
        q1.add(50);
        System.out.println("Elements in Queue:"+q1);
        //remove () method =>removes first element from the queue
        System.out.println("Element removed from the queue: "+q1.remove());
        //element() => returns head of the queue
        System.out.println("Head of the queue: "+q1.element());
        //poll () => removes and returns the head
        System.out.println("Poll():Returned Head of the queue: "+q1.poll());
        //returns head of the queue
        System.out.println("peek():Head of the queue: "+q1.peek());
        //print the contents of the Queue
        System.out.println("Final Queue:"+q1);
    }
}
```

Sample output

```
D:\Subjects_Handled\CS8392-OOP-2020-24\JavaPrograms>java QueueTest
Elements in Queue:[10, 20, 30, 40, 50]
Element removed from the queue: 10
Head of the queue: 20
Poll():Returned Head of the queue: 20
peek():Head of the queue: 30
Final Queue:[30, 40, 50]
```

Result

Thus the java program to implement queue operations was successfully executed.

Ex. No.:3

Date:

PAYROLL PROCESSING

Aim:

To develop a Java application with employee class and generate pay slips for the employees with their gross and net salary.

Procedure:

1. Start the program
2. Create Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members.
3. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class.
4. 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 fund.
5. Generate pay slips for the employees with their gross and net salary.
6. Stop the program

Program

```
import java.util.Scanner;
class Emp
{
String ename,Address,email,mobile;
int eid;
double BP,DA,HRA,PF,Sfund,GP,NP;
void getEmployeeedetails()
{
Scanner in = new Scanner(System.in);
System.out.println("Enter the Emp_id. :");
eid=in.nextInt();
System.out.println("Enter the Employee Name:");
ename=in.next();
System.out.println("Enter the Employee Address:");
Address=in.next();
System.out.println("Enter the Employee Email id :");
email=in.next();
System.out.println("Enter the Mobile No:");
mobile=in.next();
}
}
```



```

void pay_calulation(double BasicPay)
{
    BP=BasicPay;
    DA=BP*0.97;
    HRA=BP*0.10;
    PF=BP*0.12;
    Sfund=BP*0.1;
    GP=BP+DA+HRA;
    NP=GP-(PF+Sfund);
    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("Gross Pay:"+GP);
    System.out.println("Net Pay:"+NP);
}
void display()
{
    System.out.println("***** Employee Salary Bill Details *****");
    System.out.println("Emp_ID:"+eid);
    System.out.println("Name:"+ename);
    System.out.println("Address:"+Address);
    System.out.println("Email Id :"+email);
    System.out.println("Mobile No:"+mobile);
}
}
class Programmer extends Emp
{
    void Programmerdetails()
    {
        double BasicPay;
        getEmployeeedetails();
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the Basic Pay of the Programmer:");
        BasicPay=in.nextDouble();
        display();
        pay_calulation(BasicPay);
    }
}
class AssistantProfessor extends Emp
{
    void APDetails()
    {
        double BasicPay;
        getEmployeeedetails();
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the Basic Pay of the AssistantProfessor:");
        BasicPay=in.nextDouble();
    }
}

```

```

display();
pay_calulation(BasicPay);
}
}
class AssociateProfessor extends Emp
{
void ASPDetails()
{
double BasicPay;
getEmployeeedetails();
Scanner in = new Scanner(System.in);
System.out.println("Enter the Basic Pay of the AssociateProfessor:");
BasicPay=in.nextDouble();
display();
pay_calulation(BasicPay);
}
}

```

```

class Professor extends Emp
{
void profDetails()
{
double BasicPay;
getEmployeeedetails();
Scanner in = new Scanner(System.in);
System.out.println("Enter the Basic Pay of the Professor:");
BasicPay=in.nextDouble();
display();
pay_calulation(BasicPay);
}
}
class Employee
{
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.println("Choose the type Employee");
System.out.println(" 1.Programmer \n 2.Assistant Professor \n 3.Associate Professor \n 4.Professor");
int ch=in.nextInt();
switch(ch)
{
case 1:
System.out.println("PROGRAMMER DETAILS");
Programmer p=new Programmer();
p.Programmerdetails();
break;

```

```

case 2:
System.out.println("ASSISTANT PROFESSOR DETAILS");
AssistantProfessor ap=new AssistantProfessor();
ap.APDetails();
break;
case 3:
System.out.println("ASSOCIATE PROFESSOR DETAILS");
AssociateProfessor asp=new AssociateProfessor();
asp.ASPDetails();
break;
case 4:
System.out.println("PROFESSOR DETAILS");
Professor pf=new Professor();
pf.profDetails();
break;
}}}

```

Sample Output

```

Choose the type Employee
1.Programmer
2.Assistant Professor
3.Associate Professor
4.Professor
2
ASSISTANT PROFESSOR DETAILS
Enter the Emp_id. :
125
Enter the Employee Name:
Harini.S
Enter the Employee Address:
Trichy
Enter the Employee Email id :
harini_81@gmail.com
Enter the Mobile No:
9768124561
Enter the Basic Pay of the AssistantProfessor:
20000
***** Employee Salary Bill Details *****
Emp_ID:125
Name:Harini.S
Address:Trichy
Email Id :harini_81@gmail.com
Mobile No:9768124561
Basic Pay:20000.0
Dearness Allowance:19400.0
House Rent Allowance:2000.0
Provident Fund:2400.0
Gross Pay:41400.0
Net Pay:37000.0

```

Result:

Thus the Java application has been created with with employee class and pay slips are generated for the employees with their gross and net salary.

Ex. No.:4

Date:

ABSTRACT CLASS

Aim

To write a Java Program to create an abstract class named Shape and provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape.

Procedure:

1. Start the program
2. Define the abstract class **shape**.
3. Define the class Rectangle with PrintArea() method that extends(makes use of) Shape.
4. Define the class Triangle with PrintArea() method that extends(makes use of) Shape.
5. Define the class Circle with PrintArea() method that extends(makes use of) Shape.
6. Print the area of the Rectangle, Triangle and Circle .
7. Stop the Program.

Program

```
import java.util.Scanner;
abstract class Shape
{
    double a,b, area;
    abstract void printArea(); // Cannot be Defined
}
class Rectangle extends Shape
{
    void printArea()
    {
        Scanner ip = new Scanner(System.in);
        System.out.println("enter the length");
        a=ip.nextInt();
        System.out.println("enter the bredth");
        b=ip.nextInt();
        area=a*b;
        System.out.println("The area is"+area);
    }
}
```

```

class Triangle extends Shape
{
    void printArea()
    {
        Scanner ip = new Scanner(System.in);
        System.out.println("enter the Height");
        a=ip.nextInt();
        System.out.println("enter the Base");
        b=ip.nextInt();
        area=0.5*a*b;
        System.out.println("The area is"+area);
    }
}

class Circle extends Shape
{
    void printArea(){
        Scanner ip = new Scanner(System.in);
        System.out.println("enter the radius");
        a=ip.nextInt();
        area=3.14*a*a;
        System.out.println("The area is"+area);
    }
}

class ShapeArea
{
    public static void main(String args[])
    {
        Scanner ip = new Scanner(System.in);
    }
}

```

```

        System.out.println("Enter your choice");
        System.out.println("1.Rectangle\n 2.Circle \n 3.Triangle");
        int ch=ip.nextInt();
        switch(ch){
            case 1:Rectangle r= new Rectangle(); r.printArea(); break;
            case 2:Circle c= new Circle();    c.printArea(); break;
            case 3:Triangle t = new Triangle();    t.printArea(); break;
            default: System.out.println("Invalid input");
        }
    }
}

```

Sample Output

```

D:\JavaPrograms>java ShapeArea
Enter your choice
1.Rectangle
 2.Circle
 3.Triangle
1
enter the length
12
enter the bredth
4
The area is48.0

```

```

D:\JavaPrograms>java ShapeArea
Enter your choice
1.Rectangle
 2.Circle
 3.Triangle
2
enter the radius
2
The area is12.56

```

Result:

Thus the design and implementation of Abstract class has been successfully executed.

Ex 5- Interfaces

Aim:

To write a java program to implement the interface shape to find the area of various shapes.

Procedure:

1. Start the program
2. Create an interface shape with a public method printarea()
3. Create a class circle that implements shape interface to find and print the area of a circle
4. Create a class rectangle that implements shape interface to find and print the area of a rectangle
5. Create a class triangle that implements shape interface to find and print the area of a triangle
6. Stop the program

Program

```
import java.util.Scanner;
interface ishape
{
    public void printArea(); // Cannot be Defined
}
class Rectangle implements ishape
{
    double a,b, area;
    public void printArea()
    {
        Scanner ip = new Scanner(System.in); System.out.println("enter the length"); a=ip.nextInt();
        System.out.println("enter the bredth"); b=ip.nextInt();
        area=a*b;
        System.out.println("The area is"+area);
    }
}
class Triangle implements ishape
{
    double a,b, area;
    public void printArea()
    {
        Scanner ip = new Scanner(System.in); System.out.println("enter the Height"); a=ip.nextInt();
        System.out.println("enter the Base"); b=ip.nextInt();
        area=0.5*a*b;
    }
}
```

```

System.out.println("The area is"+area);
}
}
class Circle implements ishape
{
double a,b, area;
public void printArea(){
Scanner ip = new Scanner(System.in); System.out.println("enter the radius"); a=ip.nextInt();
area=3.14*a*a; System.out.println("The area is"+area);
}}
class InterShapeArea
{
public static void main(String args[])
{
Scanner ip = new Scanner(System.in);
System.out.println("Enter your choice"); System.out.println("1.Rectangle\n 2.Circle \n 3.Triangle"); int
ch=ip.nextInt();
switch(ch){
case 1:Rectangle r= new Rectangle(); r.printArea(); break; case 2:Circle c= new Circle();
c.printArea(); break;
case 3:Triangle t = new Triangle(); t.printArea(); break; default: System.out.println("Invalid
input");
}
}
}
}

```

Sample output

```

D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java InterShapeArea
Enter your choice
1.Rectangle
2.Circle
3.Triangle
3
enter the Height
12
enter the Base
3
The area is18.0

```

Result

Thus the java program to find the area of various shapes using an interface was successfully executed.

**Ex.
No.:6**

Date:

EXCEPTION HANDLING

Aim

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

Procedure:

1. Start the program
2. Define the exception for getting age from the user.
3. If the age is greater than or equal to 18, print.
4. If the age is less than 18 throw the exception to the user as
 'InvalidAgeException'
5. Stop the Program.

Program

```
class InvalidAgeException extends Exception
{
    InvalidAgeException(String s)
    {
        super(s);
    }
}
class TestCustomException
{

    void validate(int age)throws InvalidAgeException
    {
        if(age<18)
            throw new InvalidAgeException("not valid");
        else
            System.out.println("welcome to vote");
    }

    public static void main(String args[])
    {
        TestCustomException ob=new TestCustomException();
        try{
            ob.validate(11);
        }catch(Exception m)
        {
            System.out.println("Exception occurred: "+m);
        }

        System.out.println("rest of the code...");
    }
}
```

Sample Output

```
D:\JavaPrograms>java TestCustomException
Exception occurred: InvalidAgeException: not valid
rest of the code...
```

Result:

Thus the user defined exception has been successfully implemented.

Ex. No.: 7

Date:

MULTITHREADING

Aim

To write a java program that implements a multi-threaded application.

Procedure:

1. Start the program
2. Design the first thread that generates a random integer for every 1 second .
3. If the first thread value is even, design the second thread as the square of the number and then print it.
4. If the first thread value is odd, then third thread will print the value of cube of the number.
5. Stop the program.

Program

```
import java.util.Random;
class Even implements Runnable
{
    public int x;
    public Even(int x)
    {
        this.x=x;
    }
    public void run()
    {
        System.out.println("Thread Name:Even Thread and square is: " + x * x);
    }
}

class Odd implements Runnable
{
    public int x;
    public Odd(int x)
    {
```

```

this.x=x;
}
public void run()
{
System.out.println("Thread Name:Odd Thread and cube is :"+ x * x * x);
}
}

class GenRandom extends Thread
{
String tname;
Random r;
Thread t1,t2;
public GenRandom(String s)
{
tname=s;
}

public void run()
{
int num=0;
r=new Random();
try {
for(int i=0;i<5;i++)
{
num=r.nextInt(10);
System.out.println("main thread and generated number is"+num);
if(num%2==0)
{
t1=new Thread(new Even(num));
t1.start();
}
else
{
t2=new Thread(new Odd(num));
t2.start();
}
Thread.sleep(2000);
System.out.println(" -----");
}
}
catch(InterruptedException ex)
{
System.out.println(ex.getMessage());
}
}
}
class OddEven

```

```

{
public static void main(String[] args)
{
GenRandom gr=new GenRandom("one");
gr.start();
}
}

```

Sample Output

```

D:\JavaPrograms>javac ex9.java

D:\JavaPrograms>java OddEven
main thread and generated number is9
Thread Name:Odd Thread and cube is :729
-----
main thread and generated number is5
Thread Name:Odd Thread and cube is :125
-----
main thread and generated number is0
Thread Name:Even Thread and square is: 0
-----
main thread and generated number is3
Thread Name:Odd Thread and cube is :27
-----
main thread and generated number is2
Thread Name:Even Thread and square is: 4
-----

```

Result:

Thus the implementation of multithreading has been done using three threads.

Ex. No.: 8

Date:

FILE OPERATIONS

Aim

To write a Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file in bytes.

Procedure:

1. Start the program
2. Read the filename from the user.
3. Use getName() Method to display the filename.
4. Use getPath() Method to display the path of the file.
5. Use getParent() Method to display its parent's information.
6. Use exists() Method to display whether the file exist or not
7. Use isFile() and isDirectory() Methods to display whether the file is file or directory.
8. Use canRead() and canWrite() methods to display whether the file is readable or writable.
9. Use lastModified() Method to display the modified information.
10. Use length() method to display the size of the file.
11. Use isHidden() Method to display whether the file is hidden or not.

Program

// UserFileHandler.java

```
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 ");
    }
}
```

```

// TestFile.java
import java.io.File;
import java.util.Scanner;
public class TestFile
{
    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();
    }
}

```

Sample Output

```

D:\JavaPrograms>java TestFile
File Handler
*****
Enter the file path
d:\javaprograms\ex1.java
The File ex1.java is Available at:d:\javaprograms
File is Only Readable
Total length of the file is :1803 bytes

D:\JavaPrograms>java TestFile
File Handler
*****
Enter the file path
d:\javaprograms\shape.java
File does not exists

```

Result:

Thus the information of the file has been displayed successfully using various file methods.

Ex 9- Generic Classes

Aim:

To develop a java program using generic classes.

Procedure:

1. Start the program
2. Create a generic class with a private data
3. Create a constructor to initialize the data
4. Define a method getData() that returns T type variable
5. Create an instance for the generic class of type integer and print the integer value.
6. Create an instance for the generic class of type string and print the string text.
7. Stop the program

Program

```
class Generic_Demo {
    public static void main(String[] args) {

        // initialize generic class
        // with Integer data
        GenericsClass<Integer> intObj = new GenericsClass<Integer>(5);
        System.out.println("Generic Class returns: " + intObj.getData());

        // initialize generic class
        // with String data
        GenericsClass<String> stringObj = new GenericsClass<String>("Java Programming");
        System.out.println("Generic Class returns: " + stringObj.getData());
    }
}

// create a generics class
class GenericsClass<T> {

    // variable of T type
    private T data;

    public GenericsClass(T data) {
        this.data = data;
    }

    // method that return T type variable
    public T getData() {
        return this.data;
    }
}
```

```
}
```

Sample output

```
D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>java Generic_Demo
Generic Class returns: 5
Generic Class returns: Java Programming

D:\Subjects_Handled\CS8392-00P-2020-24\JavaPrograms>
```

Result

Thus the java program using generic class was successfully implemented.

Ex 10- Application using JavaFX

Aim:

To develop a simple JavaFX application to display a text message

Procedure:

1. Start the program
2. Create a new text shape with a text message to be displayed
3. Create a stack pane
4. Add the text shape to the stackpane
5. Create a scene container for all the objects
6. Host the scene by using the top level container, stage.
7. Stop the program

Program

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.text.Text;

public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            // create a new Text shape
            Text messageText = new Text("Hello World! Lets learn JavaFX.");

            // stack page
            StackPane root = new StackPane();

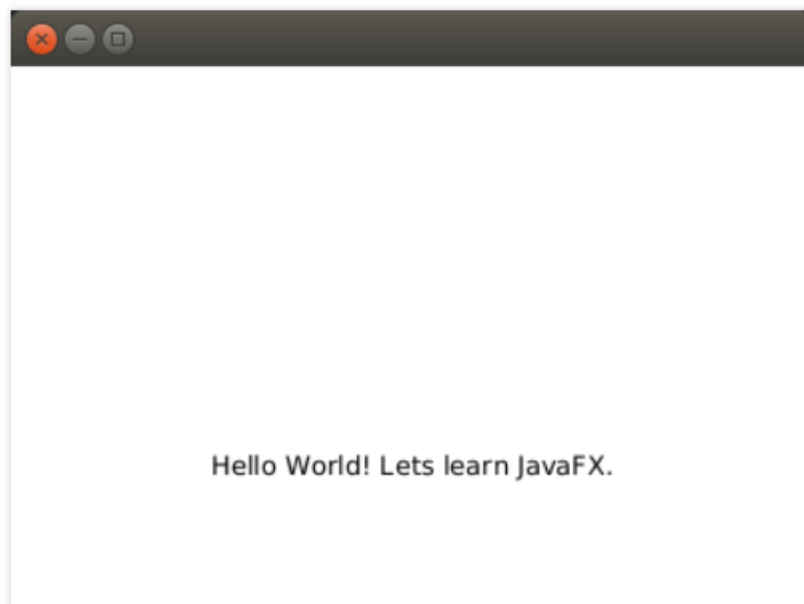
            // add Text shape to Stack Pane
            root.getChildren().add(messageText);

            Scene scene = new Scene(root,400,400);
```

```
scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
    primaryStage.setScene(scene);
    primaryStage.show();
} catch (Exception e) {
    e.printStackTrace();
}
}

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

Sample output



Result

Thus the java fx application to display a text message has been successfully executed.

Ex. No.: 11

Date:

MINI PROJECT(STUDENT MARKSHEET USING AWT EVENT HANDLING)

Aim

To generate student marksheet using AWT event handling.

Procedure:

1. Start the program
2. Using the AWT components design the necessary layout.
3. Get the details of student such as student name, register number, gender and marks.
4. Handle the action events and display the total and average.
5. Stop the program

Program

```
import java.awt.*;
import java.awt.event.*;

class Stupanel extends Panel implements ActionListener,ItemListener
{
    String s1,s2,s3;
    TextField t3,t4,t5,t6,t7;
    Button tot,avg;
    Checkbox c1,c2,c3,c4,m,f;
    CheckboxGroup cbg;
    Panel p1,p2,p3,p4;
    public Stupanel()
    {
        s3=" ";
```

```

tot=new Button("Total");
avg=new Button("Average");
c1=new Checkbox("CSE",true);
c2=new Checkbox("ECE");
c3=new Checkbox("IT");
c4=new Checkbox("EEE");
cbg=new CheckboxGroup();
m=new Checkbox("Male",cbg,false);
f=new Checkbox("Female",cbg,true);

p1=new Panel();
p1.setLayout(new GridLayout(2,2));

p1.add(new Label("Student Number "));
p1.add(new TextField(5));

p1.add(new Label("Student Name "));
p1.add(new TextField(15));
add(p1);

p2=new Panel();
p2.setLayout(new GridLayout(1,3));
p2.add(new Label("Gender"));
p2.add(m);
p2.add(f);
add(p2);

```

```
p3=new Panel();  
p3.setLayout(new GridLayout(1,5));  
p3.add(new Label("Degree"));  
p3.add(c1); p3.add(c2); p3.add(c3); p3.add(c4);  
add(p3);
```

```
p4=new Panel();  
p4.setLayout(new GridLayout(6,2));  
p4.add(new Label("Marks in OOPS"));  
t3=new TextField(3); p4.add(t3);
```

```
p4.add(new Label("Marks in DS"));  
t4=new TextField(3); p4.add(t4);
```

```
p4.add(new Label("Marks In DM"));  
t5=new TextField(3); p4.add(t5);
```

```
p4.add(new Label("Total"));  
t6=new TextField(3); p4.add(t6);
```

```
p4.add(new Label(" Average"));  
t7=new TextField(3); p4.add(t7);  
p4.add(tot); p4.add(avg);  
tot.addActionListener(this);  
avg.addActionListener(this);  
c1.addItemListener(this);  
c2.addItemListener(this);
```

```

c3.addItemListener(this);
c4.addItemListener(this);
m.addItemListener(this);
f.addItemListener(this);
add(p4);
}
public void actionPerformed(ActionEvent e)
{
    int no,m1,m2,m3,total;
    float average=0.0f;
    no=m1=m2=m3=total=0;
    m1=Integer.parseInt(t3.getText());
    m2=Integer.parseInt(t4.getText());
    m3=Integer.parseInt(t5.getText());
    total=m1+m2+m3;
    average= total/3;
    s1=String.valueOf(total);
    s2=String.valueOf(average);
    s3=e.getActionCommand();
    if(s3.equals("Total"))
    {
        t6.setText(s1);
    }
    if(s3.equals("Average"))
    {
        t7.setText(s2);
    }
}

```



```

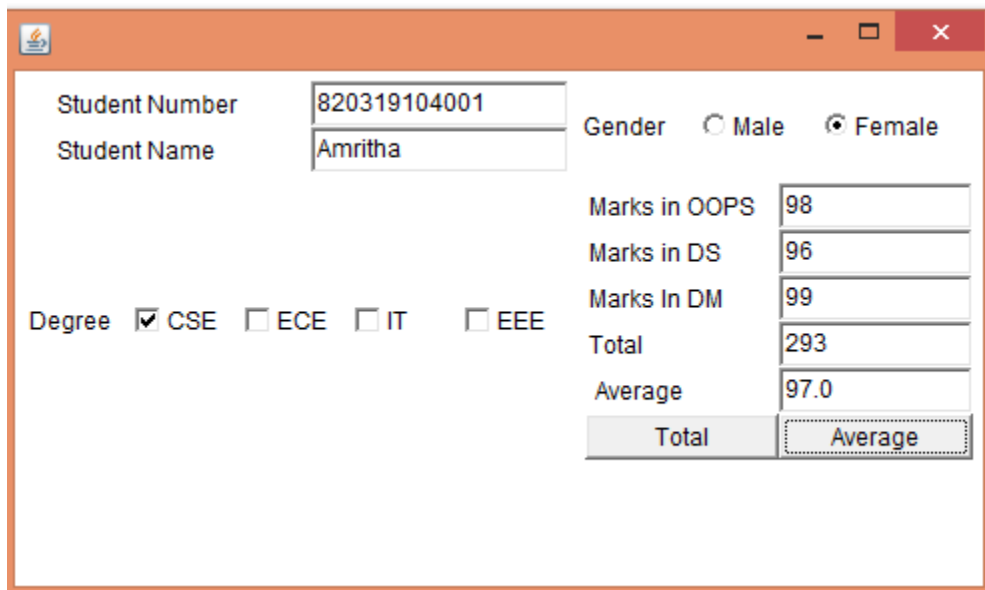
    }
    public void itemStateChanged(ItemEvent e)
    {
    }
}

class Student extends Frame {
public Student() {
    setLayout(new BorderLayout());
    add(new Stupanel(),BorderLayout.CENTER);
    setVisible(true);
    setSize(500,500);
    addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        dispose();
    }
    });
}
}

public class MarkSheet {
    public static void main(String[] args) {
        new Student();
    }
}

```

Sample Output



A screenshot of a Java Swing window titled "Sample Output" with an orange border. The window contains a form for generating a student marksheet. The form is organized into several sections:

- Student Information:** Two text input fields. The first is labeled "Student Number" and contains the value "820319104001". The second is labeled "Student Name" and contains the value "Amritha".
- Gender:** A label "Gender" followed by two radio buttons. The "Male" radio button is unselected, and the "Female" radio button is selected.
- Degree:** A label "Degree" followed by four checkboxes. The "CSE" checkbox is checked, while "ECE", "IT", and "EEE" are unchecked.
- Marks Table:** A table with two columns. The first column lists the subjects: "Marks in OOPS", "Marks in DS", "Marks In DM", "Total", and "Average". The second column contains the corresponding marks: "98", "96", "99", "293", and "97.0".
- Summary:** A table with two columns. The first column is labeled "Total" and the second column is labeled "Average".

Student Information	
Student Number	820319104001
Student Name	Amritha

Gender: ☐ Male ☒ Female

Degree: ☒ CSE ☐ ECE ☐ IT ☐ EEE

Subject	Marks
Marks in OOPS	98
Marks in DS	96
Marks In DM	99
Total	293
Average	97.0

Total	Average

Result:

Thus student marksheet was generated successfully.