

MAHA BARATHI ENGINEERING COLLEGE  
CHINNASALEM – 606 201

**Department of Computer Science and Engineering**

**LAB MANUAL**



SUBJECT CODE : CS3381  
SUBJECT NAME : OBJECT ORIENTED PROGRAMMING LABORATORY  
YEAR/ SEMESTER : II / III  
REGULATION : 2021

**PREPARED BY,**  
A.ANISHA DARATHY(AP/CSE)

**APPROVED BY,**  
Mr. N. KATHIRKUMAR (HOD / CSE)

## CONTENTS

**Subject Code : CS3381**

**Subject Name : OBJECT ORIENTED PROGRAMMING LABORATORY**

**Year & Semester : II / III**

<b>S. NO.</b>	<b>LIST OF EXPERIMENTS / PROGRAMS</b>	<b>PAGE NO.</b>
1A	SEQUENTIAL SEARCH	1
1B	BINARY SEARCH	3
1C	SELECTION SORT	5
1D	INSERTION SORT	7
2A	STACK USING CLASSES AND OBJECTS.	9
2B	QUEUE USING CLASSES AND OBJECTS.	12
3	GENERATING PAY SLIP FOR EMPLOYEES	15
4	FINDING AREA USING ABSTRACT CLASS	19
5	FINDING AREA USING INTERFACE	22
6A	JAVA EXCEPTION HANDLING	25
6B	USER DEFINED EXCEPTION HANDLING	26
7	MULTITHREADING	28
8	INPUT/OUTPUT FILES	32
9	GENERIC CLASSES	35
10A	JAVAFX CONTROLS AND LAYOUTS	37
10B	JAVAFX MENUS	39

# CS3381 / OBJECT ORIENTED PROGRAMMING LABORATORY

## LIST OF EXPERIMENTS

### COURSE OBJECTIVES:

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

### LIST OF EXPERIMENTS

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

**TOTAL: 45 PERIODS**

### COURSE OUTCOMES:

On completion of this course, the students will be able to:

- CO1:** Design and develop java programs using object oriented programming concepts
- CO2:** Develop simple applications using object oriented concepts such as package, exceptions
- CO3:** Implement multithreading, and generics concepts
- CO4:** Create GUIs and event driven programming applications for real world problems
- CO5:** Implement and deploy web applications using Java.

### LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:

#### SOFTWARE:

Operating Systems: Linux / Windows  
Front End Tools: Eclipse IDE / Netbeans IDE

#### HARDWARE:

Standalone desktops - 30 Nos. (or) Server supporting 30 terminals or more.

**EX.NO:1.A**

**DATE :**

## **SEQUENTIAL SEARCH**

### **Aim**

To write a Java program to solve problems in array by using sequential search algorithm.

### **Algorithm:**

1. Start the program
2. Create SequentialSearch class with one dimensional array.
3. Read the search element from the user.
4. Compare the search element with the first element in the list
5. If both are matched, then display "Given element is found!!!" and terminate the function
6. If both are not matched, then compare search element with the next element in the list.
7. Repeat steps 3 and 4 until search element is compared with last element in the list.
8. If last element in the list also doesn't match, then display "Element is not found!!!" and terminate the function.
9. Stop the program.

### **Program:**

```
public class SequentialSearch{
    public static void main(String[] args){
        int[] One={2,9,6,7,4,5,3,0,1};
        int target=4;
        sequentialSearch(One,target);
    }
    public static void sequentialSearch(int[] a,int b){
        int index=-1;
        for(int i=0;i<a.length;i++){
            if(a[i]==b){
                index=i;
                break;
            }
        }
        if(index== -1){
            System.out.println("Your target integer does not exist in the array");
        }
    }
}
```

```
else{  
System.out.println("Your target integer is in index"+index+"of the array");  
}  
}  
}
```

**Output:**

Your target integer is in index 4 of the array

**Result:**

Thus the program for solving problems in array using sequential search algorithm has been executed and the output was verified.

**EX.NO:1.B**

**DATE:**

## **BINARY SEARCH**

### **Aim**

To write a Java program to solve problems in array by using binary search algorithm.

### **Algorithm**

1. Start the program
2. Create BinarySearch class with one dimensional array.
3. Read the search element from the user.
4. Find the middle element in the sorted list.
5. Compare the search element with the middle element in the sorted list.
6. If both are matched, then display "Given element is found!!!" and terminate the function.
7. If both are not matched, then check whether the search element is smaller or larger than the middle element.
8. If the search element is smaller than middle element, repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.
9. If the search element is larger than middle element, repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.
10. Repeat the same process until we find the search element in the list or until sublist contains only one element.
11. If that element also doesn't match with the search element, then display "Element is not found in the list!!!" and terminate the function.
12. Stop the program.

### **Program:**

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

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

**Output:**

Element is found at index : 2

**Result:**

Thus the program for solving problems in array using binary search algorithm has been executed and the output was verified.

**EX.NO:1.C.**

## **SELECTION SORT**

**DATE:**

### **Aim:**

To write a Java program to solve problems in array by using selection sort algorithm.

### **Algorithm:**

1. Start the program.
2. Create a SelectionSort class with one dimensional array.
3. Select the first element of the list (i.e., Element at first position in the list).
4. Compare the selected element with all the other elements in the list.
5. In every comparison, if any element is found smaller than the selected element (for Ascending order), then both are swapped.
6. Repeat the same procedure with element in the next position in the list till the entire list is sorted.
7. Stop the program

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



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

**Output:**

Before Selection Sort

9 14 3 2 43 11 58 22

After Selection Sort

2 3 11 14 22 43 58

**Result:**

Thus the program for solving problems in array using selection sort algorithm has been executed and the output was verified.

**EX.NO:1.D**

**DATE:**

## **INSERTION SORT**

### **Aim**

To write a Java program to solve problems in array by using insertion sort algorithm.

### **Algorithm**

1. Start the program.
2. Create a InsertionSort class with one dimensional array.
3. Assume that first element in the list is in sorted portion and all the remaining elements are in unsorted portion.
4. Take first element from the unsorted portion and insert that element into the sorted portion in the order specified.
5. Repeat the above process until all the elements from the unsorted portion are moved into the sorted portion.
6. Stop the program.

### **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={9,14,3,2,43,11,58,22};
    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+"");  
    }  
    }  
}
```

**Output:**

Before Insertion Sort

9 14 3 2 43 11 58 22

After Insertion Sort

2 3 9 11 14 22 43 58

**Result:**

Thus the program for solving problems in array using insertion sort algorithm has been executed and the output was verified.

**EX.NO:2.A**

## **STACK**

**DATE:**

### **Aim**

To write a Java program to develop a stack data structures using classes and objects.

### **Algorithm:**

1. Start the program.
2. Create a stack class and define stack size.
3. Declare all the **functions** used in stack implementation.
4. Create a one dimensional array with fixed size
5. Define a integer variable '**top**' and initialize with '**-1**'.
6. In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.
7. Check whether **stack** is **FULL**. If it is **FULL**, then display "**Stack is FULL!**" and terminate the function. If it is not full, then increment **top** value by one and set stack[top] to value.
8. Check whether **stack** is **EMPTY**. If it is **EMPTY**, then display "**Stack is EMPTY!!!**" and terminate the function. If it is not empty, then define a variable '**i**' and initialize with top. Display **stack[i]** value and decrement **i** value by one (**i--**).
9. Stop the program

### **Program:**

```
class Stack{
    private int arr[];
    private int top;
    private int capacity;
    Stack(int size){
        arr=new int[size];
        capacity=size;
        top=-1;
    }
    public void push(int x){
        if(isFull()){
            System.out.println("Overflow\nProgram Terminated\n");
            System.exit(-1);
        }
        System.out.println("Inserting"+x);
        arr[++top]=x;
    }
}
```

```

public int pop(){
    if(isEmpty()){
        System.out.println("Underflow\nProgram Terminated");
        System.exit(-1);
    }

    System.out.println("Removing"+peek());
    return arr[top--];
}

public int peek(){
    if(!isEmpty()){
        return arr[top];
    }
    else{
        System.exit(-1);
    }
    return -1;
}

public int size(){
    return top+1;
}

public boolean isFull(){
    return top==capacity-1;
}
}

class Main{
    public static void main(String[] args){
        Stack stack=new Stack(3);
        stack.push(1);
        stack.push(2);
        stack.pop();
        stack.pop();
        stack.push(3);
        System.out.println("The top element is"+stack.peek());
        System.out.println("The stack size is"+stack.size());
        stack.pop();
        if(stack.isEmpty()){
            System.out.println("The stack is empty");
        }
        else{
            System.out.println("The stack is not empty");
        }
    }
}

```

**Output:**

Inserting 1

Inserting 2

Removing 2

Removing 1

Inserting 3

Thr Top element is 3

The Stack size is 1

Removing 3

The Stack is empty

**Result:**

Thus the Java program to develop a stack data structures using classes and objects has been executed and the output was verified.

**EX.NO:2.B**

**QUEUE**

**DATE:**

### **Aim**

To write a Java program to develop a queue data structures using classes and objects.

### **Algorithm:**

1. Create a class DemoQueue.
2. Declare all the **user defined functions** which are used in queue implementation.
3. Create a one dimensional array with above defined SIZE
4. Define two integer variables '**front**' and '**rear**' and initialize rear with '**-1**'.
5. Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.
6. Check whether queue is **FULL**. If it is **FULL**, then display "**Queue is FULL!!!** . If it is **NOT FULL**, then increment **rear** value by one (**rear++**) and set queue[rear] = item.
7. Check whether queue is **EMPTY**. If it is **EMPTY**, then display "**Queue is EMPTY!!!**" and terminate the function. If it is not empty, then define an integer variable '**i**' and set '**i = front+1**'.
8. Display 'queue[i]' value and increment '**i**' value by one (**i++**). Repeat the same until '**i**' value reaches to rear .
9. Stop the program.

### **Program:**

```
public class DemoQueue{
    private int maxSize;
    private int[]queueArray;
    private int front;
    private int rear;
    private int currentSize;
    public DemoQueue(int size){
        this.maxSize=size;
        this.queueArray=new int[size];
        front=0;
        rear=-1;
        currentSize=0;
    }
    public void insert(int item){
        if(isQueueFull()){
            System.out.println("Queue is full!");
            return;
        }
    }
```

```

    if(rear==maxSize-1){
        rear=-1;
    }

    queueArray[++rear]=item;
    currentSize++;
    System.out.println("Item added to queue:"+item);
}
public int delete(){
    if(isQueueEmpty()){
        throw new RuntimeException("Queue is empty");
    }

    int temp=queueArray[front++];
    if(front==maxSize){
        front=0;
    }
    currentSize--;
    return temp;
}
public int peek(){
    return queueArray[front];
}
public boolean isQueueFull(){
    return(maxSize==currentSize);
}
public boolean isQueueEmpty(){
    return(currentSize==0);
}
public static void main(String[]args){
    DemoQueue queue=new DemoQueue(10);
    queue.insert(2);
    queue.insert(3);
    System.out.println("Item deleted from queue:"+queue.delete());
    System.out.println("Item deleted from queue:"+queue.delete());
    queue.insert(5);
    System.out.println("Item deleted from queue:"+queue.delete());
}
}

```



**Output:**

Item added to queue : 2

Item added to queue : 3

Item deleted from queue : 2

Item deleted from queue : 3

Item added to queue : 5

Item deleted from queue : 5

**Result:**

Thus the Java program to develop a queue data structures using classes and objects has been executed and the output was verified.

**EX.NO:3**  
**DATE:**

## **GENERATING PAY SLIP FOR EMPLOYEES**

### **Aim**

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

### **Algorithm**

13. Start the program
14. Create Employee class with Emp\_name, Emp\_id, Address, Mail\_id, Mobile\_no as members.
15. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class.
16. 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.
17. Generate pay slips for the employees with their gross and net salary.
18. Stop the program

### **PROGRAM:**

```
import java.util.Scanner;
class Employee
{
int Emp_id;
String Emp_name;
String address;
String mail_id;
int mobile_no;
double bp,gross_salary,net_salary;
Employee()
{}
Employee(int id,String name,String addr,String mail, int mob)
{
this.Emp_id=id;
this.Emp_name=name;
this.address=addr;
this.mail_id=mail;
this.mobile_no=mob;
}
void computePay()
{
```

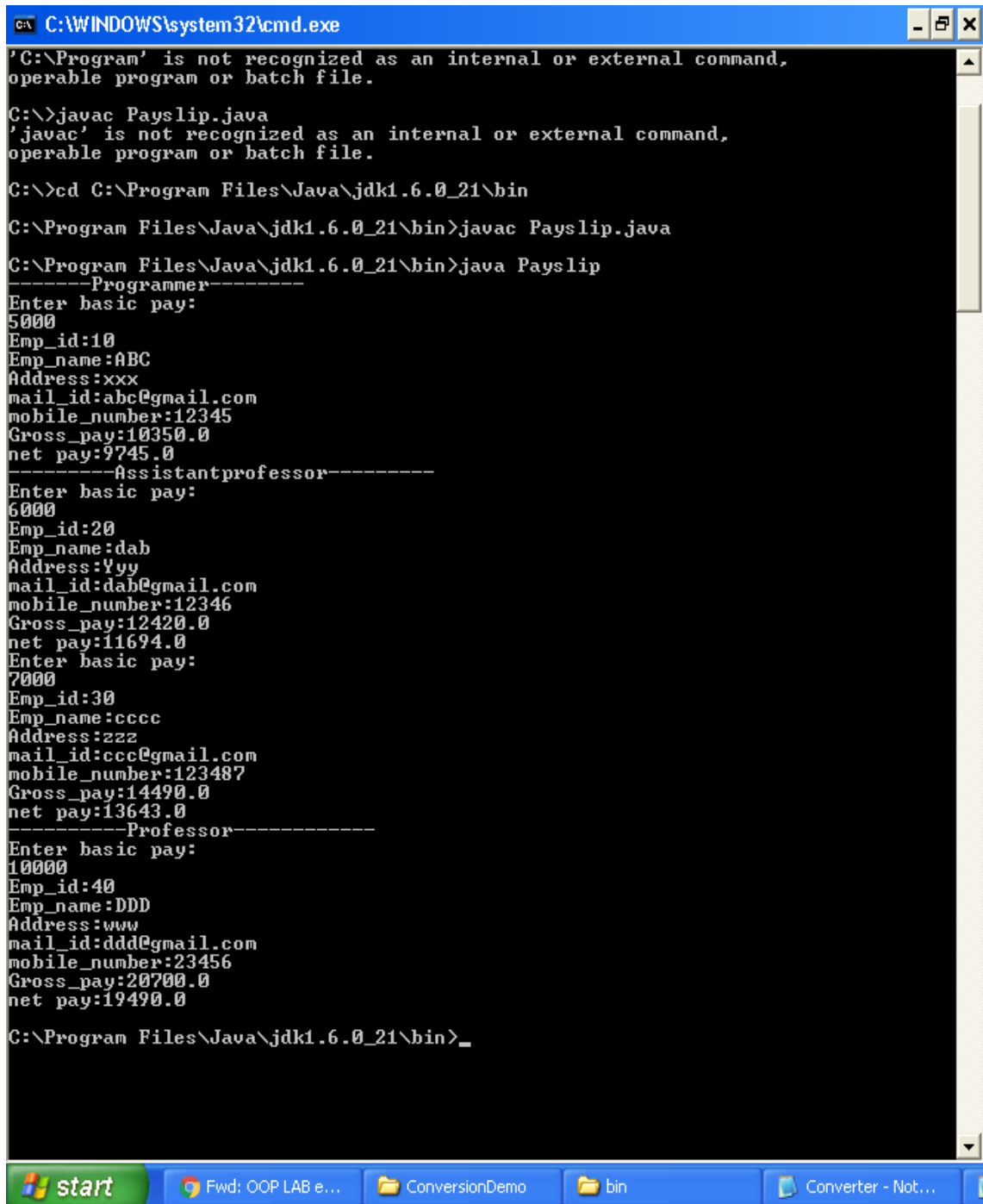
```

System.out.println("Enter basic pay:");
Scanner input=new Scanner(System.in);
bp=input.nextDouble();
double da,hra,pf,fund;
da=(bp*97/100);
hra=(bp*10/100);
pf=(bp*12/100);
fund=(bp*0.1/100);
gross_salary=bp+da+hra;
net_salary=bp+da+hra-(pf+fund);
System.out.println("Emp_id:"+Emp_id);
System.out.println("Emp_name:"+Emp_name);
System.out.println("Address:"+address);
System.out.println("mail_id:"+mail_id);
System.out.println("mobile_number:"+mobile_no);
System.out.println("Gross_pay:"+gross_salary);
System.out.println("net pay:"+net_salary);
}
}
class Programmer extends Employee
{
public Programmer(int id,String name,String addr,String mail, int mob)
{
super(id,name,addr,mail,mob);
}
}
class Asst_prof extends Employee
{
public Asst_prof(int id,String name,String addr,String mail, int mob)
{
super(id,name,addr,mail,mob);
}
}
class Assoc_prof extends Employee
{
public Assoc_prof(int id,String name,String addr,String mail, int mob)
{
super(id,name,addr,mail,mob);
}
}
class Prof extends Employee
{
public Prof(int id,String name,String addr,String mail, int mob)
{
super(id,name,addr,mail,mob);
}
}

```

```
}  
public class Payslip  
{  
public static void main(String[] args)  
{  
Programmer p = newProgrammer(10,"ABC","xxx","abc@gmail.com",12345);  
System.out.println("-----Programmer -----");  
    omputepay();  
Asst_prof ap =newAsst_prof(20,"dab","Yyy","dab@gmail.com",12346);  
System.out.println("-----Assistantprofessor -----");  
ap.computePay();  
Assoc_prof ac=newAssoc_prof(30,"cccc","zzz","ccc@gmail.com",123487);  
ac.computePay();  
Prof pro=newProf(40,"DDD","www","ddd@gmail.com",23456);  
System.out.println("-----Professor -----");  
pro.computePay();  
}  
}
```

## OUTPUT:



```
C:\WINDOWS\system32\cmd.exe
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.

C:\>javac Payslip.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\>cd C:\Program Files\Java\jdk1.6.0_21\bin
C:\Program Files\Java\jdk1.6.0_21\bin>javac Payslip.java
C:\Program Files\Java\jdk1.6.0_21\bin>java Payslip
-----Programmer-----
Enter basic pay:
5000
Emp_id:10
Emp_name:ABC
Address:xxx
mail_id:abc@gmail.com
mobile_number:12345
Gross_pay:10350.0
net_pay:9745.0
-----Assistantprofessor-----
Enter basic pay:
6000
Emp_id:20
Emp_name:dab
Address:Yyy
mail_id:dab@gmail.com
mobile_number:12346
Gross_pay:12420.0
net_pay:11694.0
Enter basic pay:
7000
Emp_id:30
Emp_name:cccc
Address:zzz
mail_id:ccc@gmail.com
mobile_number:123487
Gross_pay:14490.0
net_pay:13643.0
-----Professor-----
Enter basic pay:
10000
Emp_id:40
Emp_name:DDD
Address:www
mail_id:ddd@gmail.com
mobile_number:23456
Gross_pay:20700.0
net_pay:19490.0

C:\Program Files\Java\jdk1.6.0_21\bin>_
```

## RESULT:

The program for developing a Java application with employee class and generate pay slips for the employees with their gross and net salary has been executed successfully and output was verified.

**EX.NO:4**  
**DATE:**

## FINDING AREA USING ABSTRACT CLASS

### **Aim**

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

### **Algorithm**

1. Start theprogram
2. Define the abstract class**Shapes**.
3. Define the class Rectangle with PrintArea() method that extends(makes useof) 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 theProgram.

### **PROGRAM:**

```
import java.util.*;
abstract class Shapes
{
double a,b;
abstract void printArea();
}
class Rectangle extends Shapes
{
void printArea()
{
System.out.println("\t\t Calculating Area ofRectangle");
Scanner in=new Scanner(System.in);
System.out.println("\nEnter length");
a=in.nextDouble();
System.out.println("\n Enter Breadth:");
b=in.nextDouble();
double area=a*b;
System.out.println("Area of Rectangle:"+area);
}
}
```

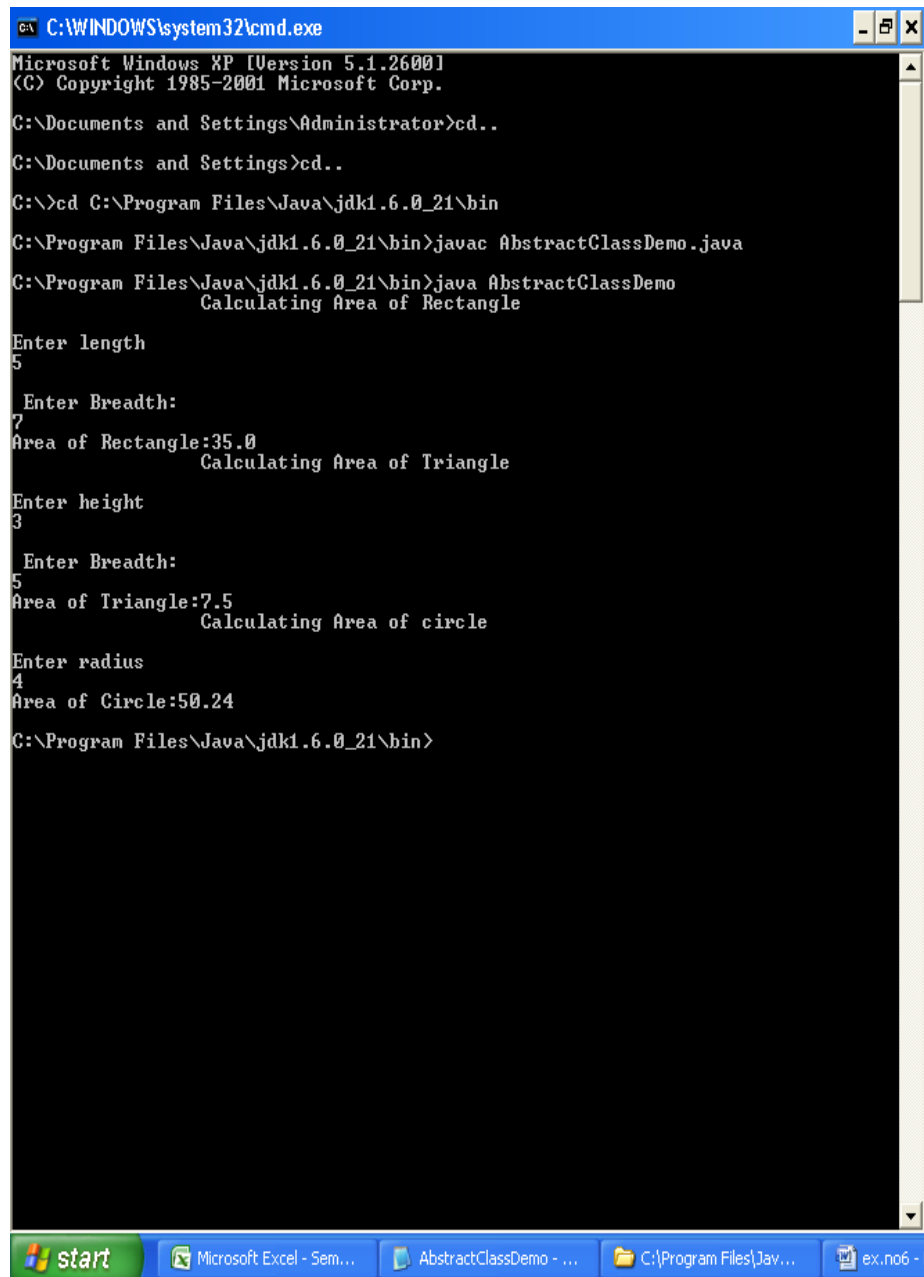
```

class Triangle extends Shapes
{

void printArea()
{
System.out.println("\t\t Calculating Area ofTriangle");
Scanner in=new Scanner(System.in);
System.out.println("\nEnter height");
a=in.nextDouble();
System.out.println("\n Enter Breadth:");
b=in.nextDouble();
double area=0.5*a*b;
System.out.println("Area of Triangle:"+area);
}
}
class Circle extends Shapes
{
void printArea()
{
System.out.println("\t\t Calculating Area of circle");
Scanner in=new Scanner(System.in);
System.out.println("\nEnter radius");
a=in.nextDouble();
double area=3.14*a*a;
System.out.println("Area of Circle:"+area);
}
}
public class AbstractClassDemo
{
public static void main(String[] args)
{
Shapes obj;
obj=new Rectangle();
obj.printArea();
obj=new Triangle();
obj.printArea();
obj=new Circle();
obj.printArea();
}
}

```

## OUTPUT:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd..
C:\Documents and Settings>cd..
C:\>cd C:\Program Files\Java\jdk1.6.0_21\bin
C:\Program Files\Java\jdk1.6.0_21\bin>javac AbstractClassDemo.java
C:\Program Files\Java\jdk1.6.0_21\bin>java AbstractClassDemo
    Calculating Area of Rectangle

Enter length
5

    Enter Breadth:
7
Area of Rectangle:35.0
    Calculating Area of Triangle

Enter height
3

    Enter Breadth:
5
Area of Triangle:7.5
    Calculating Area of circle

Enter radius
4
Area of Circle:50.24

C:\Program Files\Java\jdk1.6.0_21\bin>
```

## RESULT:

The program for a 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 has been executed successfully and output was verified.



**EX.NO:5**

**FINDING AREA USING AN INTERFACE**

**DATE:**

**Aim**

To write a Java Program to find area of shapes using an interface named Shapes and provide three classes named Rectangle, Triangle and Circle such that each one of the classes implements the interface Shapes.

**Algorithm**

1. Start the program
2. Define the interface **Area**.
3. Define the class Rectangle with PrintArea() method that implements the interface Shapes.
4. Define the class Triangle with PrintArea() method that implements the interface Shapes.
5. Define the class Circle with PrintArea() method that implements the interface Shapes.
6. Interface Shapes.
7. Print the area of the Rectangle, Triangle and Circle.
8. Stop the Program.

**PROGRAM:**

```
import java.util.*;
interface Area
{
void printArea();
}
class Rectangle implements Area
{
private double a,b;
public void printArea()
{
System.out.println("\t\t Calculating Area ofRectangle");
Scanner in=new Scanner(System.in);
System.out.println("\nEnter length");
a=in.nextDouble();
System.out.println("\n Enter Breadth:");
b=in.nextDouble();
double area=a*b;
System.out.println("Area of Rectangle:"+area);
}
}
```

```

class Triangle implements Area
{

private double a,b;
public void printArea()
{
System.out.println("\t\t Calculating Area ofTriangle");
Scanner in=new Scanner(System.in);
System.out.println("\nEnter height");
a=in.nextDouble();
System.out.println("\n Enter Breadth:");
b=in.nextDouble();
double area=0.5*a*b;
System.out.println("Area of Triangle:"+area);
}
}
class Circle implements Area
{
private double a;
public void printArea()
{
System.out.println("\t\t Calculating Area of circle");
Scanner in=new Scanner(System.in);
System.out.println("\nEnter radius");
a=in.nextDouble();
double area=3.14*a*a;
System.out.println("Area of Circle:"+area);

}
}

public class InterfaceDemo
{
public static void main(String[] args)
{
Area obj;
obj=new Rectangle();
obj.printArea();
obj=new Triangle();
obj.printArea();
obj=new Circle();
obj.printArea();
}
}

```

## **OUTPUT:**

Calculating Area of Rectangle

Enter Length

4

Enter Breadth

5

Area of Rectangle:20.0

Calculating Area of Triangle

Enter Height

5

Enter Breadth

3

Area of Triangle:7.5

Calculating Area of Circle

Enter Radius

4

Area of Circle:50.24

BUILD SUCCESSFUL (total time: 18 seconds)

## **RESULT:**

The program for a to find area of shapes using an interface Shapes and provide three classes named Rectangle, Triangle and Circle such that each one of the classes implements the interface Shapes has been executed successfully and output was verified.

**Ex No:6A**

**JAVA EXCEPTION HANDLING**

**DATE:**

**Aim:**

To write for a java program creating simple java exception handling.

**Algorithm:**

1. Start
2. Create an exception under the class Exceptions
3. Initialize an string array and it can be iterated using for loop
4. In main method class, try and catch blocks are used to display the exception in the program.
5. Stop

**PROGRAM:**

```
class Exceptions {
public static void main(String[] args) {
String languages[] = { "C", "C++", "Java", "Perl", "Python" };
try {
for (int c = 1; c <= 5; c++) {
System.out.println(languages[c]);

}
}
catch (Exception e) {
System.out.println(e);
}
}
}
```

**OUTPUT :**

```
C++
Java
Perl
Python
java.lang.ArrayIndexOutOfBoundsException: 5
```

**RESULT:**

Thus the java program for exception handling was executed and output was verified.

**EX.NO:6B**

**DATE:**

**USER DEFINED EXCEPTION HANDLING**

**Aim**

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

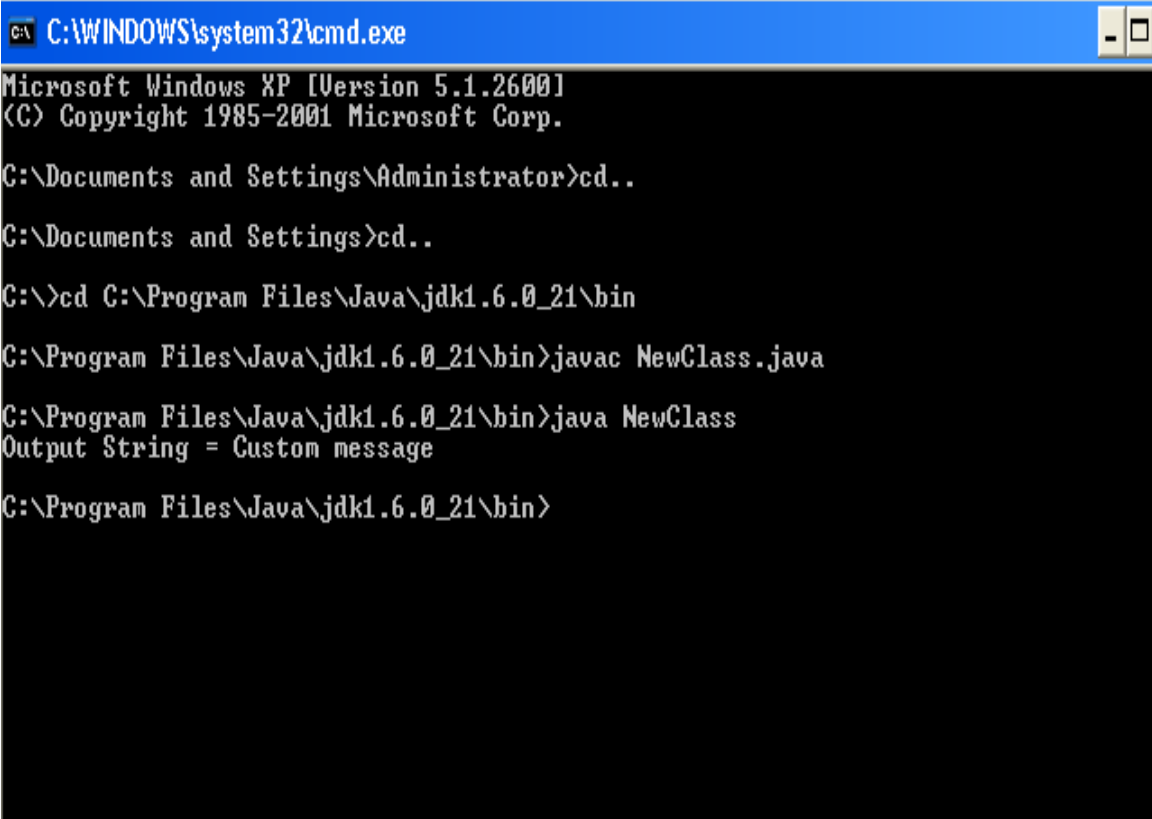
**Algorithm**

6. Start the program.
7. Create an exception under the class MyException
8. In exception block, assigns s1=s2
9. Define a function toString() to display the output message
10. In main method class, try and catch blocks are used to display the exception in the program.
11. Stop the program

**PROGRAM:**

```
import java.lang.*;
class MyException extends Exception {
String s1;
MyException(String s2) {
s1 = s2;
}
@Override
public String toString() {
return ("Output String = "+s1);
}
}
public class NewClass {
public static void main(String args[]) {
try {
throw new MyException("Custom message");
} catch(MyException exp) {
System.out.println(exp);
}}}
```

## OUTPUT:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd..

C:\Documents and Settings>cd..

C:\>cd C:\Program Files\Java\jdk1.6.0_21\bin

C:\Program Files\Java\jdk1.6.0_21\bin>javac NewClass.java

C:\Program Files\Java\jdk1.6.0_21\bin>java NewClass
Output String = Custom message

C:\Program Files\Java\jdk1.6.0_21\bin>
```

## RESULT:

Thus the Java program to implement user defined exception handling has been executed successfully and output was verified.

**EX.NO:7**

## MULTITHREADING

**DATE:**

### **Aim**

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

### **Algorithm**

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.*;
class NumberGenerate
{
private int value;
private boolean flag;
public synchronized void put()
{
while(flag)
{
try
{
wait();
}
catch(InterruptedException e)
{}
}
flag=true;
Random random=new Random();
this.value=random.nextInt(10);
System.out.println("The Generated number is"+value);
notifyAll();
}
//consumer
public synchronized void get1()
{
```

```

{

try
{
wait();
}
catch(InterruptedException e)
{}
}
if(value%2==0)
{
System.out.println("Second is executing now...");
int ans=value*value;
System.out.println(value+"is Even Number Square is:"+ans);
}
flag=false;
notifyAll();
}
public synchronized void get2()
{
while(flag)
{
try
{
wait();
}
catch(InterruptedException e)
{
}
}
if(value%2!=0)
{
System.out.println("Third is executing now...");
int ans=value*value*value;
System.out.println(value+"is odd number and its cube is:"+ans);
}
flag=false;
notifyAll();
}
}
public class TestNumber
{
public static void main(String[] args)
{
final NumberGenerate obj=new NumberGenerate();
Thread producerThread=new Thread()
{

```



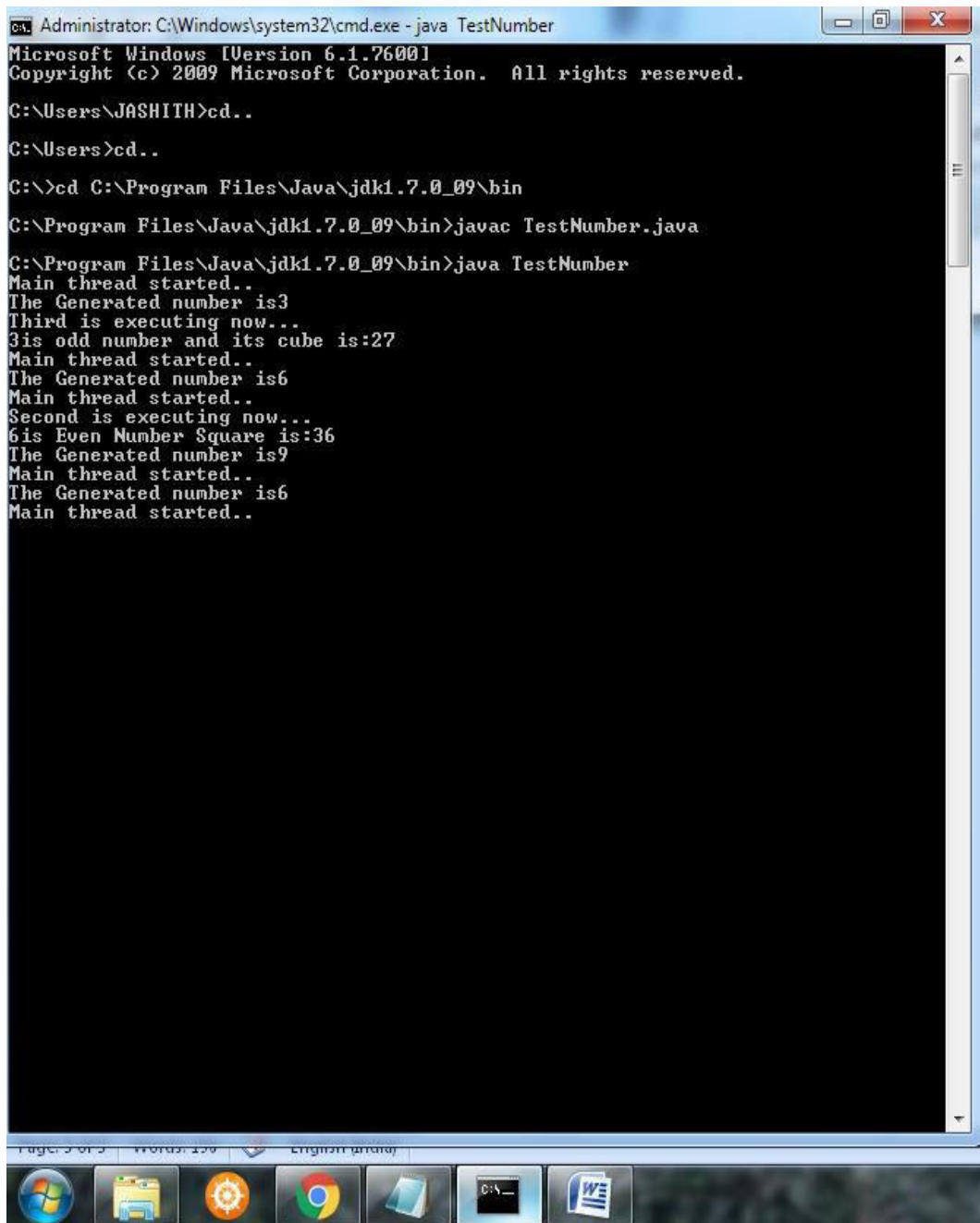
```

{
for(int i=1;i<=6;i++)
{
System.out.println("Main thread started..");

obj.put();
try
{
Thread.sleep(1000);
}
catch(InterruptedException e)
{}
}
};
Thread consumerThread1=new Thread()
{
public void run()
{
for(int i=1;i<=3;i++)
{
obj.get1();
try
{
Thread.sleep(2000);
}
catch(InterruptedException e)
{}}
};
Thread consumerThread2=new Thread()
{
public void run()
{
for(int i=1;i<=3;i++)
{
obj.get2();
try
{
Thread.sleep(3000);
}
catch(InterruptedException e)
{}}
};
producerThread.start();
consumerThread1.start();
consumerThread2.start();
}}

```

## OUTPUT:



```
Administrator: C:\Windows\system32\cmd.exe - java TestNumber
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\JASHITH>cd..
C:\Users>cd..
C:\>cd C:\Program Files\Java\jdk1.7.0_09\bin
C:\Program Files\Java\jdk1.7.0_09\bin>javac TestNumber.java
C:\Program Files\Java\jdk1.7.0_09\bin>java TestNumber
Main thread started..
The Generated number is3
Third is executing now...
3is odd number and its cube is:27
Main thread started..
The Generated number is6
Main thread started..
Second is executing now...
6is Even Number Square is:36
The Generated number is9
Main thread started..
The Generated number is6
Main thread started..
```

## RESULT:

Thus the program that implements a multi-threaded application has been executed successfully and output was verified.

<b>EX.NO:8</b>	<b>INPUT/OUTPUT FILES</b>
<b>DATE:</b>	

### **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.

### **Algorithm**

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.
12. Stop the Program.

**PROGRAM:**

```
import java.util.*;
import java.io.*;
class FileDemo
{
public static void main(String[]args)
{
System.out.println("Enter the name of the file");
Scanner in=newScanner(System.in);
String s=in.nextLine();
File f1=new File(s);
System.out.println(".....");
System.out.println("FileName:"+f1.getName());
System.out.println("Path:"+f1.getPath());
System.out.println("Absolutepath:"+f1.getAbsolutePath());
System.out.println("This file is :"+(f1.exists()?"Exists":"Does not Exists"));
System.out.println("Is File:"+f1.isFile());
System.out.println("Is Directory:"+f1.isDirectory());
System.out.println("Is Readable:"+f1.canRead());
System.out.println("Is Writable:"+f1.canWrite());
System.out.println("Is Absolute:"+f1.isAbsolute());
System.out.println("FileSize:"+f1.length()+"bytes");
System.out.println("is Hidden:"+f1.isHidden());
}
}
```

## OUTPUT:



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrator>cd..
C:\Documents and Settings>cd..
C:\>cd C:\Program Files\Java\jdk1.6.0_21\bin
C:\Program Files\Java\jdk1.6.0_21\bin>javac FileDemo.java
C:\Program Files\Java\jdk1.6.0_21\bin>java FileDemo
Enter the name of the file
FileDemo.java
-----
FileName:FileDemo.java
Path:FileDemo.java
Absolute path:C:\Program Files\Java\jdk1.6.0_21\bin\FileDemo.java
This file is :Exists
Is File:true
Is Directory:false
Is Readable:true
Is Writable:true
Is Absolute:false
FileSize:894bytes
Is Hidden:false
C:\Program Files\Java\jdk1.6.0_21\bin>
```

## RESULT:

Thus the 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 has been executed successfully and output was verified.

**EX.NO:9**

## **GENERIC CLASSES**

**DATE:**

### **Aim**

To write a Java program to develop applications to demonstrate the features of generic classes.

### **Algorithm**

1. Start the program
2. Create a generic class to find area with two types.
3. Create an object for generic class.
4. Create an object for integer type for rectangle.
5. Create an object for double type for circle.
6. Use get() Method to display values.
7. Stop the Program.

### **PROGRAM:**

```
public class Area<T>{
//T is the Datatype like String,
private T t;
private void add(T t){
this.t=t;
}
public T get()
{
return t;
}
public void getArea()
{}

public static void main(String[]args){
//object of generic class Area with parameter Type as Integer
Area<Integer>rectangle=new Area<Integer>();
//object of generic class Area with parameter Type as Double
Area<Double>circle=new Area<Double>();
rectangle.add(10);
circle.add(2.5);
System.out.println(rectangle.get());
System.out.println(circle.get());
}
}
```

**Output:**

10  
2.5

**Result:**

To write a Java program to develop applications to demonstrate the features of generic classes has been executed and the output was verified.

**EX.NO:10A**

## **JAVAFX CONTROLS AND LAYOUTS**

**DATE:**

### **Aim**

To write a program to develop applications using JavaFX controls and layouts.

### **Algorithm:**

1. Start the program
2. Override start method and give title for stage.
3. Create a Flow pane layout by setting title, width and height .
4. Create a scene for Flow pane.
5. Add buttons to flow pane.
6. Handle the action events for buttons.
7. Show the stage and its scene.
8. Use launch(args) to start JavaFX application.
9. Stop the Program.

### **PROGRAM :**

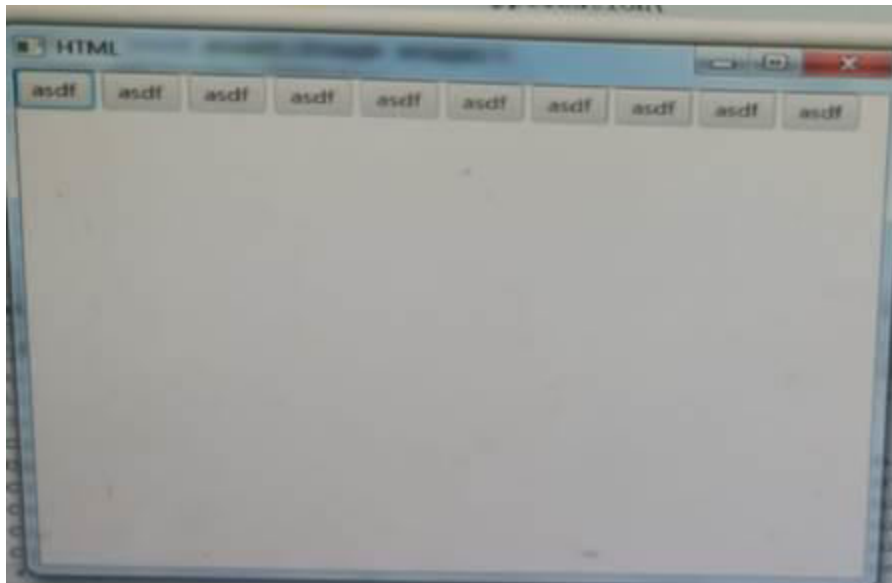
```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;
/*from w w w .java2 s.com*/
public class Main2 extends Application{

    public void start(Stage stage){
        stage.setTitle("HTML");
        stage.setWidth(500);
        stage.setHeight(500);
        Scene scene=new Scene(new Group());
        FlowPane flow=new FlowPane();
        flow.setVgap(8);
        flow.setHgap(4);
        flow.setPrefWrapLength(300);//preferred width=300
        for(int i=0;i<10;i++){
```



```
    flow.getChildren().add(new Button("asdf"));
}
scene.setRoot(flow);
stage.setScene(scene);
stage.show();
}
public static void main(String[]args){
    launch(args);
}
}
```

### **OUTPUT:**



### **RESULT:**

Thus the program to develop applications using JavaFX controls and layouts has been executed and the output was verified.

**EX.NO:10B**

## **JAVAFX MENUS**

**DATE:**

### **Aim**

To write a program to develop applications using JavaFX menus.

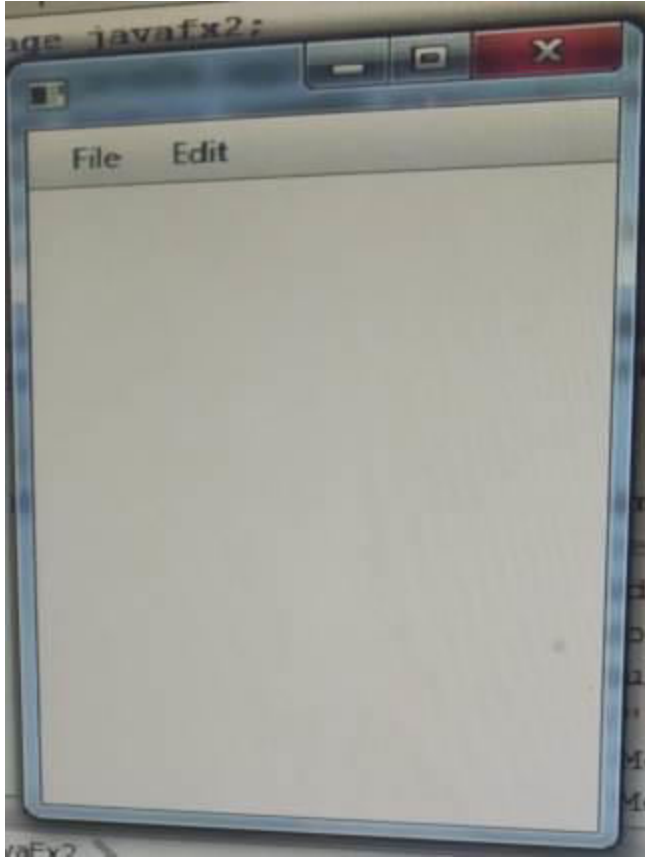
### **Algorithm:**

1. Start the program
2. Use launch(args) to start JavaFX application.
3. Create a Border pane layout .
4. Create a scene for Border pane.
5. Add menu bar to border pane.
6. Add menu items for each menu.
7. Show the stage and its scene.
8. Stop the Program.

### **PROGRAM :**

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
public class Application1 extends Application{
    public static void main(String[]args){
        launch(args);
    }
    public void start(Stage primaryStage)throws Exception{
        //TODO Auto-generated method stub
        BorderPane root=new BorderPane();
        Scene scene=new Scene(root,200,300);
        MenuBar menubar=new MenuBar();
        Menu FileMenu=new Menu("File");
        MenuItem filemenu1=new MenuItem("new");
        MenuItem filemenu2=new MenuItem("Save");
        MenuItem filemenu3=new MenuItem("Exit");
        Menu EditMenu=new Menu("Edit");
        MenuItem EditMenu1=new MenuItem("Cut");
        MenuItem EditMenu2=new MenuItem("Copy");
        MenuItem EditMenu3=new MenuItem("Paste");
        EditMenu.getItems().addAll(EditMenu1,EditMenu2,EditMenu3);
        root.setTop(menubar);
        FileMenu.getItems().addAll(filemenu1,filemenu2,filemenu3);
        menubar.getMenus().addAll(FileMenu,EditMenu);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

## OUTPUT:



## RESULT:

Thus the program to develop applications using JavaFX menus has been executed and the output was verified.