

<b>Ex. No.: 1.(A)</b>	<b>Implement The Program By Using Sequential Search Algorithm</b>
<b>Date:</b>	

### **AIM:**

To develop a Java program by using sequential search algorithm.

### **ALGORITHM:**

Step 1: Start the program.

Step 2: Start from the leftmost element of arr[] and one by one compare x with each element of arr[]

Step 3: Let the element to be search be x.

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: Stop the program.

### **PROGRAM:**

```

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

```

## OUTPUT:

```
G:\JS\OOPS\Sample>javac SequentialSearch.java
G:\JS\OOPS\Sample>java SequentialSearch
Your target integer 4 is in index 5 of the array
G:\JS\OOPS\Sample>
```

## RESULT:

Thus the java program has been executed successfully and the output is verified.

<b>Ex. No.: 1.(B)</b>	<b>Implement The Program By Using Binary Search Algorithm</b>
<b>Date:</b>	

**AIM:**

To develop a Java program by using Binary search algorithm.

**ALGORITHM:**

Step 1: Compare x with the middle element.

Step 2: If x matches with middle element, we return the mid index.

Step 3: Else if x is greater then the mid element, then x can only lie in the right half subarray after the mid element. So we recur for right half.

Step 4: Else(x is smaller) recur for the left half.

**PROGRAM:**

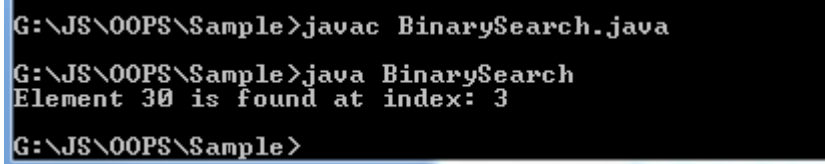
```

class BinarySearch
{
public static void binarySearch(int arr[], int first, int last, int key)
{
int mid = (first + last)/2;
while( first <= last )
{
if ( arr[mid] < key )
{
first = mid + 1;
}else if ( arr[mid] == key )
{
System.out.println("Element "+key+" is found at index: " + (mid+1));
break;
}else
{
last = mid - 1;
}
mid = (first + last)/2;
}
if ( first > last )
{
System.out.println("Element is not found!");
}
}
public static void main(String args[])
{

```

```
int arr[] = { 10,20,30,40,50};  
int key = 30;  
int last=arr.length-1;  
binarySearch(arr,0,last,key);  
}  
}
```

### OUTPUT:



```
G:\JS\OOPS\Sample>javac BinarySearch.java  
G:\JS\OOPS\Sample>java BinarySearch  
Element 30 is found at index: 3  
G:\JS\OOPS\Sample>
```

### RESULT:

Thus the java program has been executed successfully and the output is verified.

<b>Ex. No.: 1. (c)</b>	<b>Implement The Program by Using Selection Sort</b>
<b>Date:</b>	

**AIM:**

Solve problems by using Selection Sort

**ALGORITHM:**

Step 1: Set Min\_Index to 0

Step 2: Search for the smallest element in the array

Step 3: Swap with value with the element at the Min\_Index

Step 4: Increment Min\_Index to point to next element

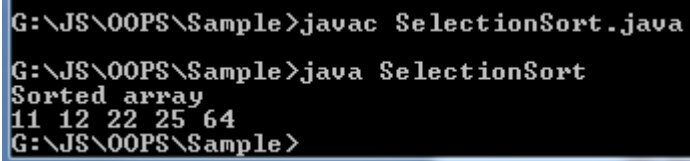
Step 5: Repeat until the complete array is sorted

**PROGRAM:**

```
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]+" ");
}
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 :**



```
G:\JS\OOPS\Sample>javac SelectionSort.java
G:\JS\OOPS\Sample>java SelectionSort
Sorted array
11 12 22 25 64
G:\JS\OOPS\Sample>
```

### **RESULT :**

The above program has been executed successfully and the required output is required

<b>Ex. No.: 1. (D)</b>	<b>Implement The Program by Using Insertion Sort</b>
<b>Date:</b>	

**AIM:**

Solve problems by using Insertion Sort

**ALGORITHM:**

Step 1: If the element is the first element, assume that it is already sorted, return 1

Step 2: Pick the next element, and store it separately in a key

Step 3: Now, compare the key with all elements in the sorted array

Step 4: If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right

Step 5: Insert the value

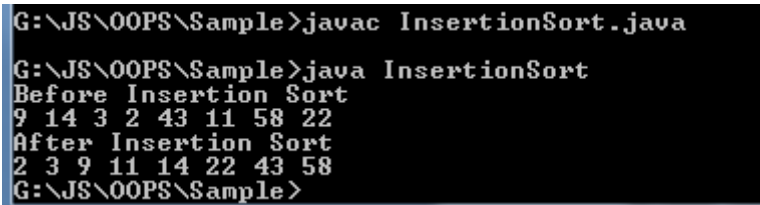
Step 6: Repeat until array is sorted

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

### OUTPUT:



```
G:\JS\OOPS\Sample>javac InsertionSort.java  
G:\JS\OOPS\Sample>java InsertionSort  
Before Insertion Sort  
9 14 3 2 43 11 58 22  
After Insertion Sort  
2 3 9 11 14 22 43 58  
G:\JS\OOPS\Sample>
```

### RESULT:

The above program has been executed successfully and the required output is required



<b>Ex. No.: 2. (A)</b>	<b>Implement The Program by Using Stack</b>
<b>Date:</b>	

**AIM:**

Develop stack class structures using classes and objects

**ALGORITHM:**

Step 1: Define a class stack and implement the following methods inside the class

Step 2: Define a method `_init_()` to initialize a list and the stack size

Step 3: Define a method `is empty()` to check if the stack is empty or not

Step 4: Define a method `is full()` to check if the stack is full

Step 5: Define a method `push()` to insert an element into the stack

Step 6: Define a method `pop()` to delete an element from the list

Step 7: Define a method `display()` to display the elements in the stack

Step 8: Create an object reference for the class stack to access the methods inside the class

Step 9: Repeat the user choices through while loop

**PROGRAM:**

```
import java.io.*;
class EXCE extends Exception
{
    String s;
    EXCE(String s, int i)
    {
        this.s=s;
    }
    public String toString()
    {
        return s;
    }
}
class Stack
{
    int top;
    int a[];
    final int MAX=6;
    Stack()
    {
        top=-1;
    }
}
```

```

a=new int[MAX];
}
void push() throws Exception
{
try
{
if(top==MAX-1)
throw new EXCE("STACK FULL",0);
else
{
DataInputStream d=new DataInputStream(System.in);
top++;
System.out.print("Enter the element to be insert in stack:");
a[top]=Integer.parseInt(d.readLine());
}
}
catch(EXCE e)
{
System.out.print(e);
}
}
void pop()
{
try
{
if(top== -1)
throw new EXCE("STACK EMPTY",1);
else
{
System.out.println("Removed Element From stack :"+a[top]);
top--;
}
}
catch(EXCE e)
{
System.out.println(e);
}
}
void display()
{
if(top== -1)
System.out.println("The stack is empty");
else
{
System.out.println("The satch elements are:");
for(int i=0; i<=top; i++)
System.out.print(a[i]+" ");
}
}
}
class StackEx

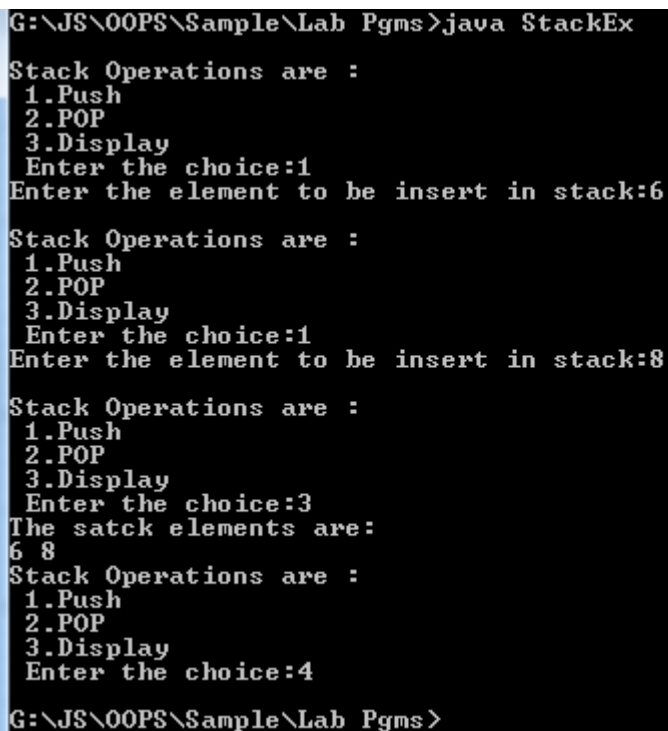
```

```

{
public static void main(String a[]) throws Exception
{
int ch;
Stack s =new Stack();
DataInputStream d=new DataInputStream(System.in);
do
{
System.out.print("\nStack Operations are : \n 1.Push \n 2.POP \n 3.Display\n Enter the choice:");
ch=Integer.parseInt(d.readLine());
switch(ch)
{
case 1: s.push();
break;
case 2: s.pop();
break;
case 3: s.display();
break;
}
}while(ch<=3);
}
}

```

### OUTPUT:



```

G:\JS\OOPS\Sample\Lab Pgms>java StackEx
Stack Operations are :
 1.Push
 2.POP
 3.Display
Enter the choice:1
Enter the element to be insert in stack:6
Stack Operations are :
 1.Push
 2.POP
 3.Display
Enter the choice:1
Enter the element to be insert in stack:8
Stack Operations are :
 1.Push
 2.POP
 3.Display
Enter the choice:3
The satchk elements are:
6 8
Stack Operations are :
 1.Push
 2.POP
 3.Display
Enter the choice:4
G:\JS\OOPS\Sample\Lab Pgms>

```

### RESULT:

The above program has been executed successfully and the required output is required

<b>Ex. No.: 2. (B)</b>	<b>Implement The Program by Using Queue</b>
<b>Date:</b>	

**AIM:**

Develop queue data structures using classes and objects

**ALGORITHM:**

Step 1: Define a class queue and implement the following methods inside the class

Step 2: Define a method `_init_()` to initialize a list and the queue size

Step 3: Define a method `is empty()` to check if the queue is empty or not

Step 4: Define a method `isfull()` to check if the queue is full

Step 5: Define a method `enqueue()` to insert an element into the queue

Step 6: Define a method `dequeue()` to delete an element from the queue

Step 7: Define a method `display()` to display the elements in the queue

Step 8: Create an object reference for the class queue to access the methods inside the class

Step 9: Repeat the user choices through while loop

**PROGRAM:**

```
class Queue
{
private static int front, rear, capacity;
private static int queue[];
Queue(int size)
{
front = rear = 0;
capacity = size;
queue = new int[capacity];
}
static void queueEnqueue(int item)
{
if (capacity == rear)
{
System.out.printf("\nQueue is full\n");
return;
}
else
{
queue[rear] = item;
```

```

    rear++;
    }
    return;
    }
    static void queueDequeue()
    {
    if (front == rear)
    {
    System.out.printf("\nQueue is empty\n");
    return;
    }
    else
    {
    for (int i = 0; i < rear - 1; i++)
    {
    queue[i] = queue[i + 1];
    }
    if (rear < capacity)
    queue[rear] = 0;

    rear--;
    }
    return;
    }
    static void queueDisplay()
    {
    int i;
    if (front == rear)
    {
    System.out.printf("Queue is Empty\n");
    return;
    }
    for (i = front; i < rear; i++)
    {
    System.out.printf(" %d , ", queue[i]);
    }
    return;
    }
    static void queueFront()
    {
    if (front == rear)
    {
    System.out.printf("Queue is Empty\n");
    return;
    }
    System.out.printf("\nFront Element of the queue: %d", queue[front]);
    return;
    }
    }
    public class QueueEx

```

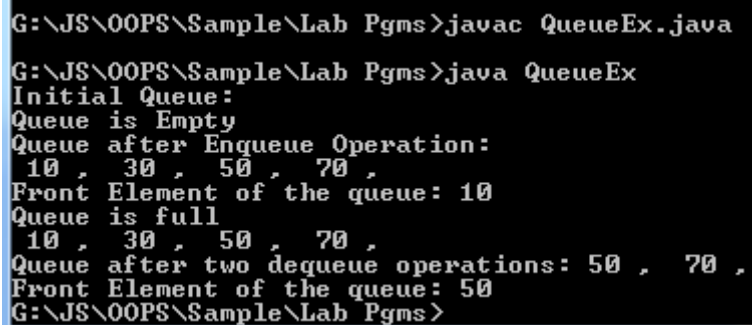
```

{
public static void main(String[] args)
{
Queue q = new Queue(4);
System.out.println("Initial Queue:");
q.queueDisplay();
q.queueEnqueue(10);
q.queueEnqueue(30);
q.queueEnqueue(50);
q.queueEnqueue(70);
System.out.println("Queue after Enqueue Operation:");
q.queueDisplay();
q.queueFront();
q.queueEnqueue(90);

q.queueDisplay();
q.queueDequeue();
q.queueDequeue();
System.out.printf("\nQueue after two dequeue operations:");
q.queueDisplay();
q.queueFront();
}
}

```

## OUTPUT:



```

G:\JS\OOPS\Sample\Lab Pgms>javac QueueEx.java
G:\JS\OOPS\Sample\Lab Pgms>java QueueEx
Initial Queue:
Queue is Empty
Queue after Enqueue Operation:
10 , 30 , 50 , 70 ,
Front Element of the queue: 10
Queue is full
10 , 30 , 50 , 70 ,
Queue after two dequeue operations: 50 , 70 ,
Front Element of the queue: 50
G:\JS\OOPS\Sample\Lab Pgms>

```

## RESULT:

The above program has been executed successfully and the required output is required

<b>Ex. No.: 3</b>	<b>Implement The Program by Using Java Application</b>
<b>Date:</b>	

### AIM:

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 salary.

### ALGORITHM:

Step 1: Create a class called Employee with emp\_name , emp\_id , address , mail\_id , mobile\_no as members and compute the payslip

Step 2: Extends the employee class for further classes called programmer , assistant professor , associate professor and professor

Step 3: Create class called pay slip and generate pay slip according to the position of the employee

### PROGRAM:

```
import java.io.*;
class Employee
{
String emp_name,mail_id, address, mobile,designation;
int id,basic_pay;
void input() throws Exception
{
DataInputStream d=new DataInputStream(System.in);
System.out.print("Enter the id :");
id=Integer.parseInt(d.readLine());
System.out.print("Enter the Name :");
emp_name=d.readLine();
System.out.print("Enter the Address :");
address=d.readLine();
System.out.print("Enter Mobile number :");
mobile=d.readLine();
System.out.print("Enter the Mail id :");
mail_id=d.readLine();
}
void display(String designation,int basic_pay)
{
float da=0.97f *basic_pay ;
float hra= 0.1f*basic_pay;
```

```

float pf= 0.12f*basic_pay;
float sfclub = 0.001f*basic_pay;
float gross_pay = basic_pay + da + hra ;
float net_pay = gross_pay - pf - sfclub;
System.out.println("-----");
System.out.println("| EMPLOYEE DETAILS MAINTENANCE |");
System.out.println("-----");
System.out.println("Employee Id :"+id);
System.out.println("Employee Name :"+emp_name);
System.out.println("Employee Address :"+address);
System.out.println("Employee Mobile Number :"+mobile);
System.out.println("Employee Mail ID :"+mail_id);
System.out.println("Designation of an employee :"+designation);
System.out.println("Basic pay of an Employee :"+basic_pay);
System.out.println("DA :"+da);
System.out.println("HRA :"+hra);
System.out.println("Gross Salary :"+gross_pay);
System.out.println("PF :"+pf);
System.out.println("Staff club amount :"+sfclub);
System.out.println("Net pay :"+net_pay);
System.out.println("-----");
}
}
class Lecturer extends Employee
{
void cal() throws Exception
{
DataInputStream d=new DataInputStream(System.in);
designation = "Lecturer";
System.out.print("Enter the basic pay of an employee :");
basic_pay = Integer.parseInt(d.readLine());
display(designation,basic_pay);
}
}
class AP extends Employee
{
void cal() throws Exception
{
DataInputStream d=new DataInputStream(System.in);
designation = "Assistant Professor";
System.out.print("Enter the basic pay of an employee :");
basic_pay = Integer.parseInt(d.readLine());
display(designation,basic_pay);
}
}
class Associate extends Employee
{
void cal() throws Exception

```



```

{
DataInputStream d=new DataInputStream(System.in);
designation = "Associate Professor";
System.out.print("Enter the basic pay of an employee :");
basic_pay = Integer.parseInt(d.readLine());
display(designation,basic_pay);
}
}
class Professor extends Employee
{
void cal() throws Exception
{
DataInputStream d=new DataInputStream(System.in);
designation = "Professor";
System.out.print("Enter the basic pay of an employee :");
basic_pay = Integer.parseInt(d.readLine());
display(designation,basic_pay);
}
}
class EmployeeSalary
{
public static void main(String a[]) throws Exception
{
DataInputStream d=new DataInputStream(System.in);
System.out.print("1.Lecturer\n2.Assistant Professor\n3.Associate Professor
\n4.Professor\nEnter the choice:");
int ch=Integer.parseInt(d.readLine());
switch(ch)
{
case 1: Lecturer e=new Lecturer();
e.input();
e.cal();
break;
case 2: AP e1=new AP();
e1.input();
e1.cal();
break;
case 3: Associate e2=new Associate();
e2.input();
e2.cal();
break;
case 4: Professor e3=new Professor();
e3.input();
e3.cal();
break;
}
}
}

```

## OUTPUT:

```
G:\JS\OOPS\Sample\Lab Pgms>javac EmployeeSalary.java
Note: EmployeeSalary.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

```
G:\JS\OOPS\Sample\Lab Pgms>java EmployeeSalary
```

```
1.Lecturer
2.Assistant Professor
3.Associate Professor
4.Professor
Enter the choice:1
Enter the id :454
Enter the Name :ghghg
Enter the Address :hghgh
Enter Mobile number :56565
Enter the Mail id :ggh
Enter the basic pay of an employee :4500
```

```
-----
! EMPLOYEE DETAILS MAINTENANCE !
-----
```

```
Employee Id :454
Employee Name :ghghg
Employee Address :hghgh
Employee Mobile Number :56565
Employee Mail ID :ggh
Designation of an employee :Lecturer
Basic pay of an Employee :4500
DA :4365.0
HRA :450.0
Gross Salary :9315.0
PF :540.0
Staff club amount :4.5
Net pay :8770.5
-----
```

## RESULT:

The above program has been executed successfully and the required output is required.

<b>Ex. No.: 4</b>	<b>Implement The Program by Using Abstract Class</b>
<b>Date:</b>	

### **AIM:**

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 , Traingle and Circle such that each one of the classes extends the class shape.

### **ALGORITHM:**

Step 1: Declare three different classes for rectangle , square and circle

Step 2: Declare two methods of the same name but with a different number of arguments or with different data types

Step 3: Call these methods using objects

Step 4: Call the corresponding methods as per the number of arguments or their datatypes

Step 5: Display the result

### **PROGRAM:**

```
import java.io.*;
import java.util.*;
abstract class shape
{
float a,b;
abstract void area() throws Exception;
}
class Rectangle extends shape
{
void area() throws Exception
{
DataInputStream d= new DataInputStream(System.in);
System.out.print("Enter the Length of Rectangle:");
a=Float.parseFloat(d.readLine());
System.out.print("Enter the Breadth of Rectangle:");
b=Float.parseFloat(d.readLine());
float ar = a*b;
System.out.println("Area of rectangle = "+ar);
}
}
class Triangle extends shape
{
void area() throws Exception
{
```

```

DataStream d= new DataInputStream(System.in);
System.out.print("Enter the Breadth of Triangle:");
a=Float.parseFloat(d.readLine());
System.out.print("Enter the Height of Triangle:");
b=Float.parseFloat(d.readLine());
float ar = 0.5f*a*b;
System.out.println("Area of Triangle = "+ar);
}
}
class Circle extends shape
{
void area() throws Exception
{
DataStream d= new DataInputStream(System.in);
System.out.print("Enter the radius of a Circle:");
a=Float.parseFloat(d.readLine());
float ar = 3.14f*a*a;
System.out.println("Area of Circle = "+ar);
}
}
class AbstractClass
{
public static void main(String a[]) throws Exception
{
DataStream d= new DataInputStream(System.in);
System.out.println("\nArea of Different shapes:\n 1. Rectangle \n 2. Triangle \n 3. Circle\n Enter
Your Choice :");
int ch=Integer.parseInt(d.readLine());
switch(ch)
{
case 1: Rectangle r=new Rectangle();
r.area();
break;
case 2: Triangle t=new Triangle();
t.area();
break;
case 3: Circle c = new Circle();
c.area();
break;
}
}
}

```

## OUTPUT:

```
G:\JS\OOPS\Sample\Lab Pgms>javac AbstractClass.java
Note: AbstractClass.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

G:\JS\OOPS\Sample\Lab Pgms>java AbstractClass

Area of Different shapes:
1. Rectangle
2. Triangle
3. Circle
Enter Your Choice :
1
Enter the Length of Rectangle:10
Enter the Breadth of Rectangle:25
Area of rectangle = 250.0

G:\JS\OOPS\Sample\Lab Pgms>
```

## RESULT :

The above program has been executed successfully and the required output is required.

<b>Ex. No.: 5</b>	<b>Implement The Program by Using Interface</b>
<b>Date:</b>	

**AIM:**

Solve the above problems using an interface

**ALGORITHM:**

Step 1: Declare three different classes for rectangle , triangle and circle

Step 2: Declare two methods of the same name but with a different number of arguments or with different data types

Step 3: Call these methods using objects

Step 4: Call the corresponding methods as per the number of arguments or their datatypes

Step 5: Declare an interface by using interface keyword

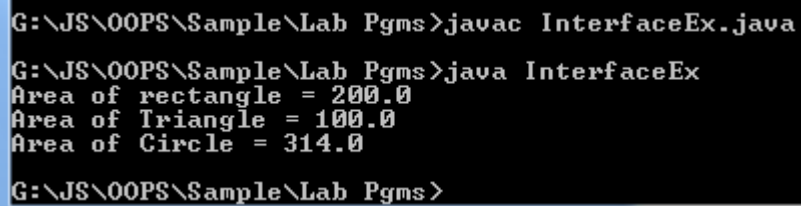
Step 6: Display the result

**PROGRAM:**

```
import java.io.*;
import java.util.*;
interface shape
{
float a=10,b=20;
void area();
}
class Rectangle implements shape
{
public void area()
{
float ar = a*b;
System.out.println("Area of rectangle = "+ar);
}
}
class Triangle implements shape
{
public void area()
{
float ar = 0.5f*a*b;
System.out.println("Area of Triangle = "+ar);
}
}
```

```
class Circle implements shape
{
public void area()
{
float ar = 3.14f*a*a;
System.out.println("Area of Circle = "+ar);
}
}
class InterfaceEx
{
public static void main(String a[])
{
Rectangle r=new Rectangle();
r.area();
Triangle t=new Triangle();
t.area();
Circle c = new Circle();
c.area();
}
}
```

#### **OUTPUT:**



```
G:\JS\OOPS\Sample\Lab Pgms>javac InterfaceEx.java
G:\JS\OOPS\Sample\Lab Pgms>java InterfaceEx
Area of rectangle = 200.0
Area of Triangle = 100.0
Area of Circle = 314.0
G:\JS\OOPS\Sample\Lab Pgms>
```

#### **RESULT:**

The above program has been executed successfully and the required output is required.

<b>Ex. No.: 6</b>	<b>Implement The Program by Using Exception Handling</b>
<b>Date:</b>	

**AIM:**

Implement exception handling and creation of user defined exceptions

**ALGORITHM:**

Step 1: Create user-defined exception by extending exception class

Step 2: Define a default constructor in your own exception class

Step 3: Define a parameterized constructor with string as a parameter call superclass constructor

Step 4: Create an object of user-defined exception class and throw it using throw clause

**PROGRAM:**

```
import java.io.*;
import java.util.*;
class blooderr extends Exception
{
String m;
blooderr(String h)
{
m=h;
}
public String toString()
{
return m;
}
}
class blood
{
float weight;
int age;
void check() throws Exception
{
try
{
DataInputStream d= new DataInputStream(System.in);
System.out.print("Enter the weight of the person :");
weight=Float.parseFloat(d.readLine());
System.out.print("Enter the age of the person :");
```



```

age=Integer.parseInt(d.readLine());
if(weight>=50 && age>=18)
System.out.println("You are elligible");
else
throw new blooderr("You are not elligible for blood donation");
}
catch(blooderr e)
{
System.out.println(e);
}
}
}
class UserException
{
public static void main(String a[]) throws Exception
{
blood b=new blood();
b.check();
}
}

```

### OUTPUT:

```

G:\JS\OOPS\Sample\Lab Pgms>javac UserException.java
Note: UserException.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

G:\JS\OOPS\Sample\Lab Pgms>java UserException
Enter the weight of the person :76
Enter the age of the person :17
You are not elligible for blood donation

G:\JS\OOPS\Sample\Lab Pgms>java UserException
Enter the weight of the person :67
Enter the age of the person :20
You are elligible

G:\JS\OOPS\Sample\Lab Pgms>

```

### RESULT:

The above program has been executed successfully and the required output is required.

<b>Ex. No.: 7</b>	<b>Implement The Program by Using Multithread</b>
<b>Date:</b>	

**AIM:**

Write a java program that implements a multithreaded 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

**ALGORITHM:**

Step 1: Create a class with the name , “even implements runnable and odd implements runnable”

Step 2: Create thread objects and random class object

Step 3: Pass the objects of our class to thread class

Step 4: Call the start method

**PROGRAM:**

```
import java.util.*;
class SquareThread extends Thread
{
    public int n;
    public SquareThread(int no)
    {
        this.n = no;
        System.out.println("Square Thread Started");
        start();
    }
    public void run()
    {
        System.out.println("The Square of the Random Number " + n + " is " +(n*n));
    }
}
class CubeThread extends Thread
{
    public int n;
    public CubeThread(int no)
    {
        this.n = no;
        System.out.println("Cube Thread Started");
        start();
    }
    public void run()
```

```

{
System.out.println("The Cube of the Random Number " + n + " is " + (n*n*n));
}
}
class RandThread extends Thread
{
public RandThread()
{
start();
}
public void run()
{
int v = 0;
Random r = new Random();
try
{
for (int i = 1; i<=10; i++)
{
v = r.nextInt(100);
System.out.println("Generated Random Number is " + v);
if (v % 2 == 0)
new SquareThread(v);
else
new CubeThread(v);
Thread.sleep(1000);
}
}
catch (InterruptedException e)
{
System.out.println("Randon Number Thread Interrupted");
}
}
}
public class MultiThreadDemo
{
public static void main(String[] args)
{
RandThread r = new RandThread();
}
}

```

## OUTPUT:

```
G:\JS\OOPS\Sample\Lab Pgms>javac MultiThreadDemo.java
G:\JS\OOPS\Sample\Lab Pgms>java MultiThreadDemo
Generated Random Number is 73
Cube Thread Started
The Cube of the Random Number 73 is 389017
Generated Random Number is 53
Cube Thread Started
The Cube of the Random Number 53 is 148877
Generated Random Number is 53
Cube Thread Started
The Cube of the Random Number 53 is 148877
Generated Random Number is 59
Cube Thread Started
The Cube of the Random Number 59 is 205379
Generated Random Number is 39
Cube Thread Started
The Cube of the Random Number 39 is 59319
Generated Random Number is 65
Cube Thread Started
The Cube of the Random Number 65 is 274625
Generated Random Number is 89
Cube Thread Started
The Cube of the Random Number 89 is 704969
Generated Random Number is 91
Cube Thread Started
The Cube of the Random Number 91 is 753571
Generated Random Number is 49
Cube Thread Started
The Cube of the Random Number 49 is 117649
Generated Random Number is 63
Cube Thread Started
The Cube of the Random Number 63 is 250047
G:\JS\OOPS\Sample\Lab Pgms>
```

## RESULT:

The above program has been executed successfully and the required output is required.

<b>Ex. No.: 8</b>	<b>Implement The Program by Using File Operations</b>
<b>Date:</b>	

### **AIM:**

Write a program to perform file operations

### **ALGORITHM:**

Step 1: Create a new file

Step 2: Write and read an existing file

Step 3: Create an object of the file class

Step 4: Perform the operations and display the result

### **PROGRAM:**

```
import java.io.*;
import java.util.*;
class WriteToFile
{
public static void main(String[] args)
{
try
{
FileWriter fwrite = new FileWriter("G:/JS/OOPS/Sample/Lab Pgms/FileExample.txt");
fwrite.write("This is a Sample File.");
fwrite.close();
System.out.println("Content is successfully wrote to the file.");
File f1 = new File("G:/JS/OOPS/Sample/Lab Pgms/FileExample.txt");
Scanner dataReader = new Scanner(f1);
System.out.println("Read the File contents are:");
while (dataReader.hasNextLine())
{
String fileData = dataReader.nextLine();
System.out.println(fileData);
}
System.out.println("The absolute path of a file:"+f1.getAbsolutePath());
System.out.println("The File Length :"+f1.length());
System.out.println("The File Access Read Mode :"+f1.canRead());
System.out.println("The File Access Write Mode :"+f1.canWrite());
dataReader.close();
}
catch (IOException e)
```

```
{  
System.out.println("Unexpected error occurred");  
e.printStackTrace();  
}  
}  
}
```

### OUTPUT:

```
G:\JS\OOPS\Sample\Lab Pgms>javac WriteToFile.java  
G:\JS\OOPS\Sample\Lab Pgms>java WriteToFile  
Content is successfully wrote to the file.  
Read the File contents are:  
This is a Sample File.  
The absolute path of a file:G:\JS\OOPS\Sample\Lab Pgms\FileExample.txt  
The File Length :22  
The File Access Read Mode :true  
The File Access Write Mode :true  
G:\JS\OOPS\Sample\Lab Pgms>
```

### RESULT:

The above program has been executed successfully and the required output is required.

<b>Ex. No.: 9</b>	<b>Implement The Program by Using Generic Classes</b>
<b>Date:</b>	

**AIM:**

Develop applications to demonstrate the features of generic classes

**ALGORITHM:**

Step 1: Create an array

Step 2: Define a generic class

Step 3: Pass multiple type parameters in generic classes

Step 4: Display the result

**PROGRAM:**

```
import java.util.*;
class Generic1
{
    public static < E > void findMax(E[] elements)
    {
        Arrays.sort(elements);
        System.out.println("The Maximum Element is " + elements[elements.length-1]);
    }
    public static void main( String args[] ) {
        Integer[] intArray = { 10, 20, 30, 40, 50 };
        Character[] charArray = { 'J', 'A', 'V', 'A', 'T','P','O','I','N','T' };
        Double[] doubleArray={ 1.5,2.5,3.9,22.5,9.0};
        findMax( intArray );
        findMax( charArray );
        findMax(doubleArray);
    }
}
```

## OUTPUT:

```
G:\JS\OOPS\Sample\Lab Pgms>javac Generic1.java
G:\JS\OOPS\Sample\Lab Pgms>java Generic1
The Maximum Element is 50
The Maximum Element is 0
The Maximum Element is 22.5
G:\JS\OOPS\Sample\Lab Pgms>
```

## RESULT:

The above program has been executed successfully and the required output is required.



<b>Ex. No.: 10</b>	<b>Implement The Program by Using Application</b>
<b>Date:</b>	

**AIM:**

Develop applications using JavaFx Controls, layouts and menus

**ALGORITHM:**

Step 1: Create the menu class

Step 2: Create required number of menu items by menuItem class

Step 3: Add the menu items

Step 4: Display the result

**PROGRAM:**

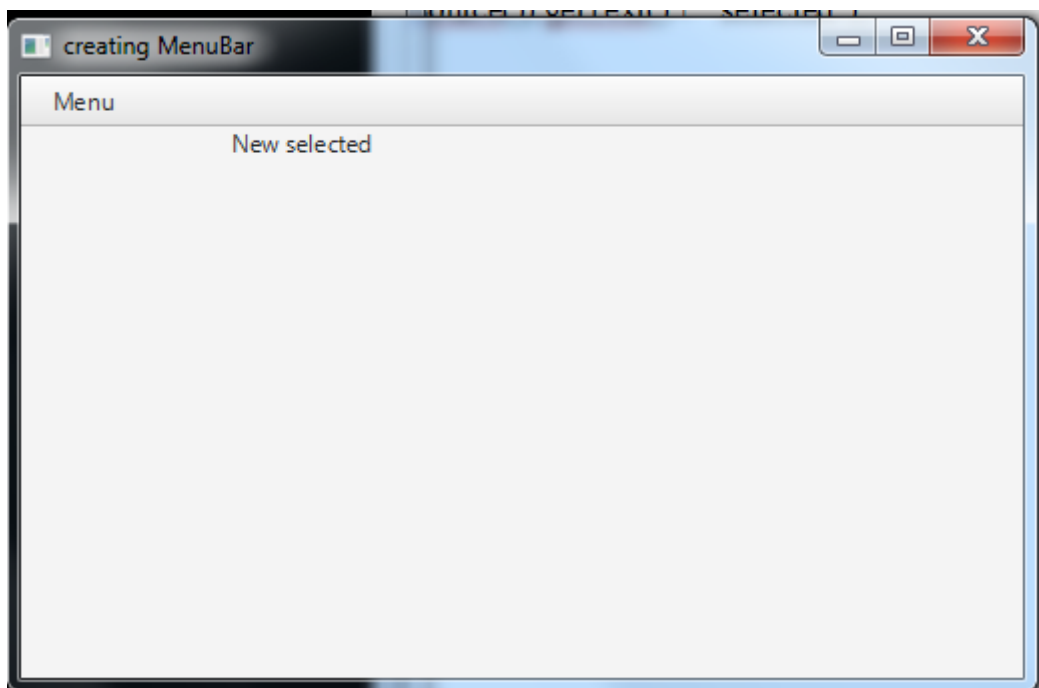
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.*;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.*;
import javafx.stage.Stage;
import javafx.*;
import java.time.LocalDate;
public class MenuBar_2 extends Application
{
    public void start(Stage s)
    {
        s.setTitle("creating MenuBar");
        Menu m = new Menu("Menu");
        MenuItem m1 = new MenuItem("New");
        MenuItem m2 = new MenuItem("Open");
        MenuItem m3 = new MenuItem("Save");
        m.getItems().add(m1);
        m.getItems().add(m2);
        m.getItems().add(m3);
        Label l = new Label("\t\t\t" + "no menu item selected");
        EventHandler<ActionEvent> event = new EventHandler<ActionEvent>()
```

```

{
public void handle(ActionEvent e)
{
l.setText("\t\t\t" + ((MenuItem)e.getSource()).getText() + " selected");
}
};
m1.setOnAction(event);
m2.setOnAction(event);
m3.setOnAction(event);
MenuBar mb = new MenuBar();
mb.getMenus().add(m);
VBox vb = new VBox(mb, l);
Scene sc = new Scene(vb, 500, 300);
s.setScene(sc);
s.show();
}
public static void main(String args[])
{
launch(args);
}
}

```

## OUTPUT:



## RESULT:

The above program has been executed successfully and the required output is required





