COURSE OUTCOME 4

PROGRAM NO-1

AIM:

Create a Graphics package that has classes and interfaces for figures Rectangle, Triangle, Square and Circle. Test the package by finding the area of these figures.

ALGORITHM:

Step 1: Start.

Step 2: Define a class having name Product and members as pcode,pname and price.

Step 3: Declare three objects in the class and add the values of each data members into objects.

Step 4: Using if condition check which object has the lowest price and print it.

Step 5: Stop.

CODE:

Graphics.Shapes.java	package Graphics;
	interface figures{
	void Rectangle(double a,double b);
	void Triangle(double a,double b);
	void Square(double a);
	void Circle(double a);
	}
	public class Shapes implements figures{
	@Override
	public void Rectangle(double a,double b){
	System.out.println("Area:"+(a*b));
	}
	@Override
	public void Triangle(double a,double b){
	System.out.println("Area:"+((a*b)/2));
	}
	@Override
	public void Square(double a) {
	System.out.println("Area:"+(a*a));
	}
	@Override
	public void Circle(double a) {
	double pi = 3.14;

	System.out.println("Area:"+(pi*a*a));
	}
	public static void main(String[] args) {
	}
	}

CO4Q1.java	import Graphics.Shapes;
	import java.util.Scanner;
	
	public class CO4Q1 {
	public static void main(String[] args) {
	int ch;
	double l,b;
	Shapes ob = new Shapes();
	Scanner sc = new Scanner(System.in);
	do
	{
	System.out.println("\n1.Rectangle\n2.Triangle\n3.Square\n4.Circle\n5.Exit\nEnter your choice:");
	ch = sc.nextInt();
	switch(ch){
	case 1:System.out.println("Enter the length and breadth:");
	l = sc.nextDouble();
	b = sc.nextDouble();
	ob.Rectangle(l, b);
	break;
	case 2:System.out.println("Enter the breadth and height:");
	l = sc.nextDouble();
	b = sc.nextDouble();
	ob.Triangle(l, b);
	break;
	case 3:System.out.println("Enter the side:");
	l = sc.nextDouble();
	ob.Square(l);
	break;
	case 4:System.out.println("Enter the radius:");
	l = sc.nextDouble();
	ob.Circle(l);
	break;
	case 5:System.exit(0);
	break;
	default:System.out.println("Invalid choice");
	}
	}while(true);
	}
	}

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
1.Rectangle
2.Triangle
3.Square
4.Circle
5.Exit
Enter your choice:
3
Enter the side:
5
Area:25.0

1.Rectangle
2.Triangle
3.Square
4.Circle
5.Exit
Enter your choice:
5
BUILD SUCCESSFUL (total time: 14 seconds)
```

PROGRAM NO-2

AIM:

Create an Arithmetic package that has classes and interfaces for the 4 basic arithmetic operations. Test the package by implementing all operations on two given numbers

ALGORITHM:

Step 1: Start
Step 2: To create a package named arithmetic, create a folder of the same name in the directory. Here inside that we have another module named operation
Step 3: Inside arithmetic package, create modules to perform addition, subtraction, multiplication and division of 2 numbers.
Step 4: Outside the folder, write another program that access the above module and print the output.
Step 5:Stop

CODE:

Arithmetic.operations.java

```
package arithmetic;

interface basic
{
   void addition(double a,double b);
   void subtraction(double a,double b);
   void multiplication(double a,double b);
   void division(double a,double b);
}


public class operations implements basic
{
   @Override
   public void addition(double a, double b) {
      System.out.println(a+" + "+b+" = "+(a+b));
   }

   @Override
   public void subtraction(double a, double b) {
      System.out.println(a+" - "+b+" = "+(a-b));
   }
```

	@Override
	public void multiplication(double a, double b) {
	System.out.println(a+" x "+b+" = "+(a*b));
	}
	
	@Override
	public void division(double a, double b) {
	System.out.println(a+" / "+b+" = "+(a/b));
	}
	
	}

CO4Q2.java	package c04q2;
	//Create an Arithmetic package that has classes and interfaces for the 4 basic arithmetic
	//operations. Test the package by implementing all operations on two given numbers
	import arithmetic.operations;
	import java.util.Scanner;
	public class C04Q2
	{
	public static void main(String[] args)
	{
	double n1,n2;
	int ch;
	operations ob = new operations();
	Scanner sc = new Scanner(System.in);
	System.out.println("Enter the 2 numbers:");
	n1 = sc.nextDouble();
	n2 = sc.nextDouble();
	
	System.out.println("\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\nEnter the choice:");
	ch = sc.nextInt();
	switch(ch)
	{
	case 1:ob.addition(n1,n2);
	break;
	case 2:ob.subtraction(n1,n2);
	break;
	case 3:ob.multiplication(n1,n2);
	break;
	case 4:ob.division(n1,n2);
	break;
	default:System.out.println("Invalid choice");

	} } }

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the 2 numbers:
5
5

1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter the choice:
3
5.0 x 5.0 = 25.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

PROGRAM NO-3

AIM:

Write a user defined exception class to authenticate the user name and password.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a User-defined exception named authException.

Step 3: Check for the validity of the username and the password against the required parameters. If any parameter is not met, then throw the authentication failed exception.

Step 6: To authenticate the credentials provided, check if the passwords and usernames given and entered are matching. If they match, display login successful, otherwise throw an exception.

Step 7: Stop the program.

CODE:

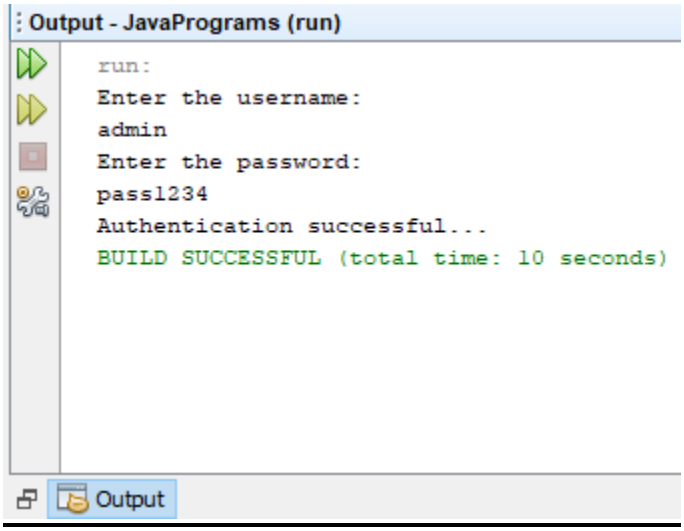
CO4Q3.java	package javaprograms;
	//Write a user defined exception class to authenticate the user name and password
	
	import java.util.Scanner;
	
	class authException extends Exception
	{
	
	public authException(String s) {
	super(s);
	}
	
	}
	public class CO4Q3
	{
	public static void main(String[] args) {

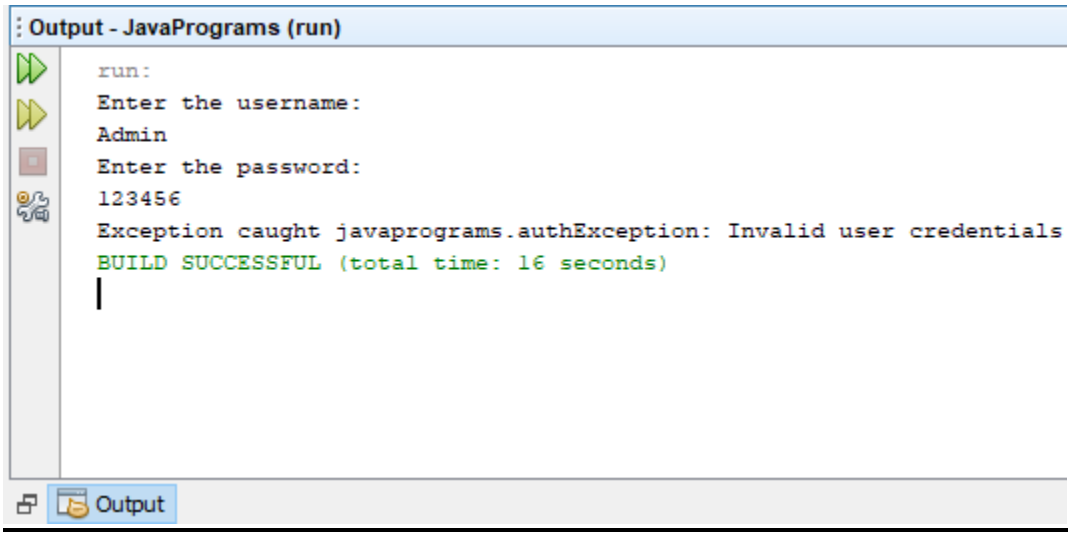
```
        String username = "admin";
        String passcode = "pass1234";
        String user_name,password;
        Scanner sc = new Scanner(System.in);
        try
        {
           System.out.println("Enter the username:");
           user_name = sc.nextLine();
//           sc.nextLine();
           System.out.println("Enter the password:");
           password = sc.nextLine();
           if(username.equals(user_name) && passcode.equals(password))
           {
              System.out.println("Authentication successful...");
           }
           else
              throw new authException("Invalid user credentials");

        }
        catch(authException e)
        {
           System.out.println("Exception caught "+e);
        }
     }

}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:


Output - JavaPrograms (run)
run:
Enter the username:
admin
Enter the password:
pass1234
Authentication successful...
BUILD SUCCESSFUL (total time: 10 seconds)
Output


Output - JavaPrograms (run)
run:
Enter the username:
Admin
Enter the password:
123456
Exception caught javaprograms.authException: Invalid user credentials
BUILD SUCCESSFUL (total time: 16 seconds)
Output

PROGRAM NO-4

AIM:

Find the average of N positive integers, raising a user defined exception for each negative input.

ALGORITHM:

Step 1: Start the program.
Step 2: Define a class named '*NegativeIntegerException*' which throws a negative number exception.
Step 3: Get user inputs for integers at run time and throw an exception if the entered number is negative.
Step 4: Else, proceed to calculate the average and display the result.
Step 5: Stop the program.

CODE:

CO4Q4.java

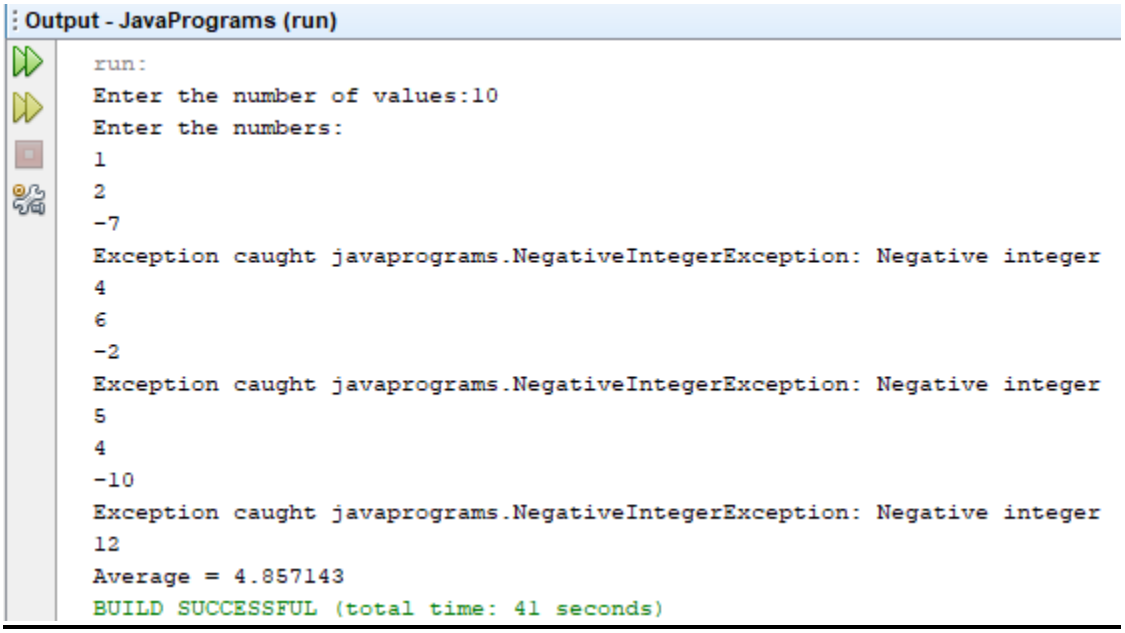
```
package javaprograms;
//Find the average of N positive integers, raising a user defined exception for each negative
//input

import java.util.Scanner;

class NegativeIntegerException extends Exception
{
   public NegativeIntegerException(String s)
   {
      super(s);
   }
}
public class CO4Q4 {
   public static void sample()
   {
      try {
         int n,count=0;
         float num[];
         float total=0;
         Scanner sc = new Scanner(System.in);
         System.out.print("Enter the number of values:");
```

```
            n = sc.nextInt();
            num = new float[n];
            System.out.println("Enter the numbers:");
            for(int i=0;i<n;i++)
            {
               num[i] = sc.nextInt();
               try{
               if(num[i]<0)
               {
                  throw new NegativeIntegerException("Negative integer");
               }
               else
               {
                  total += num[i];
                  count++;
               }
               }catch(NegativeIntegerException e)
               {
                  System.out.println("Exception caught "+e);
               }
            }
            System.out.println("Average = "+(total/count));
         } catch (Exception e) {
            System.out.println("Exception caught "+e);
         }
      }
      public static void main(String[] args) {
         try {
            sample();
         } catch (Exception e) {
         }
      }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:


Output - JavaPrograms (run)
run:
Enter the number of values:10
Enter the numbers:
1
2
-7
Exception caught javaprograms.NegativeIntegerException: Negative integer
4
6
-2
Exception caught javaprograms.NegativeIntegerException: Negative integer
5
4
-10
Exception caught javaprograms.NegativeIntegerException: Negative integer
12
Average = 4.857143
BUILD SUCCESSFUL (total time: 41 seconds)

PROGRAM NO-5

AIM:

Define 2 classes; one for generating multiplication table of 5 and other for displaying first N prime numbers. Implement using threads. (Thread class)

ALGORITHM:

Step 1: Start the program.

Step 2: Define classes named '*MultiplicationTable*' and '*PrimeNumbers*' that extends *Thread class* and contain methods to compute the multiplication table of 5 and to generate first N prime prime numbers respectively.

Step 3: Define a main method to create objects for the classes and invoke the associated methods.

Step 4: Stop the program.

CODE:

CO4Q5.java	//Define 2 classes; one for generating multiplication table of 5 and other for displaying first
	//N prime numbers. Implement using threads. (Thread class)
	package javaprograms;
	
	import java.util.Scanner;
	import java.util.logging.Level;
	import java.util.logging.Logger;
	
	class MultiplicationTable extends Thread
	{
	public void run()
	{
	System.out.println("Thread is running");
	for(int i=1;i<=10;i++)
	{

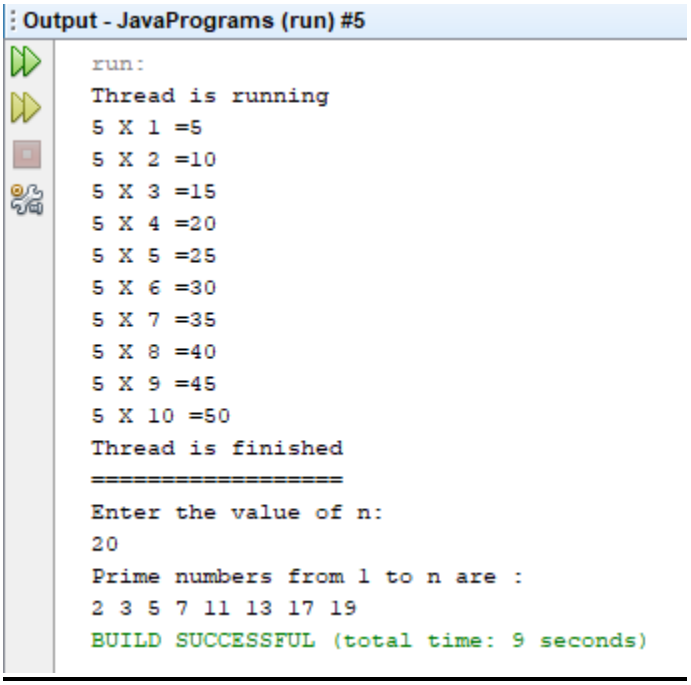
```
            try
            {
               System.out.println("5 X "+i+" ="+(5*i));
               Thread.sleep(200);
            } catch (InterruptedException ex) {

            }
        }
        System.out.println("Thread is finished");
        System.out.println("==================");
    }
}
class PrimeNumbers extends Thread
{
    public void run()
    {
      Scanner sc = new Scanner(System.in);
      int i =0;
      int num =0;
      String  primeNumbers = "";
      System.out.println("Enter the value of n:");
      int n = sc.nextInt();
      for (i = 1; i <= n; i++)
      {
        int counter=0;
        for(num =i; num>=1; num--)
        {
             if(i%num==0)
             {
                  counter = counter + 1;
             }
           }
           if (counter ==2)
           {
             primeNumbers = primeNumbers + i + " ";
           }
      }
      System.out.println("Prime numbers from 1 to n are :");
      System.out.println(primeNumbers);
    }
}
public class CO4Q5 {
    public static void main(String[] args) throws InterruptedException {
        MultiplicationTable MTOb = new MultiplicationTable();
        MTOb.start();
        Thread.sleep(5000);
```

```
        PrimeNumbers PNob = new PrimeNumbers();
        PNob.start();
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:



PROGRAM NO-6

AIM:

Define 2 classes; one for generating Fibonacci numbers and other for displaying even numbers in a given range. Implement using threads. (Runnable Interface)

ALGORITHM:

Step.1: Start the program.

Step.2: Define classes named '*Fibonacci*' and '*Even*' that implements *Runnable interface* and contain methods to generate Fibonacci series and to display even numbers in a given range respectively.

Step.3: Define a main method to create objects for the classes and invoke the associated methods.

Step.4: Stop the program.

CODE:

CO4Q6.java

```
//Define 2 classes; one for generating Fibonacci numbers and other for displaying even
//numbers in a given range. Implement using threads. (Runnable Interface)
package javaprograms;

import java.util.Scanner;

class Fibonacci implements Runnable
{
   int n,first,second,t;
   String str;

   public Fibonacci(int num)
   {
      n = num;
      first = 0;
      second = 1;
   }

   @Override
   public void run()
```

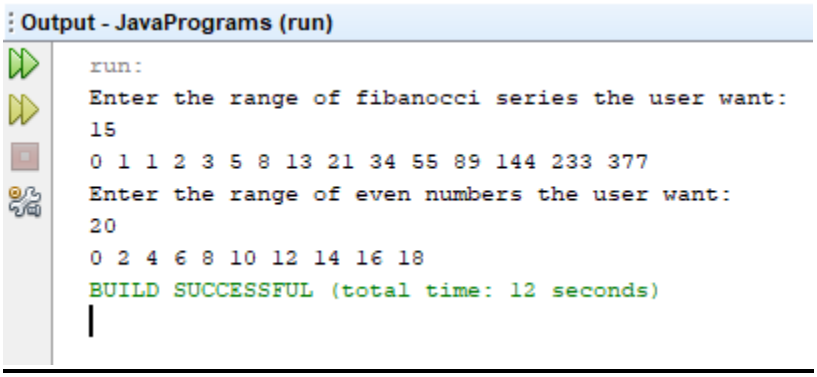
```
    {
        str = first+" "+second;
        for(int i=0;i<=n-3;i++)
        {
            t = first + second;
            first = second;
            second = t;
            str += " "+t;
        }
        System.out.println(str);
    }

}
class Even implements Runnable
{
    int n;
    String str;
    public Even(int n)
    {
        this.n = n;
        str = "";
    }
    @Override
    public void run()
    {
        for(int i=0;i<n;i=i+2)
            if(i%2==0)
            {
                str+=i+" ";
            }
        System.out.println(str);
    }

}
public class CO4Q6 {
    public static void main(String[] args) throws InterruptedException {
        int n1,n2;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the range of fibanocci series the user
want:");
        n1 = sc.nextInt();
        Fibonacci fibob = new Fibonacci(n1);
        Thread th = new Thread(fibob);
        th.start();
        Thread.sleep(400);
        System.out.println("Enter the range of even numbers the user
```

```
want:");
        n2 = sc.nextInt();
        Even evob = new Even(n2);
        Thread th2 = new Thread(evob);
        th2.start();
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:



PROGRAM NO-7

AIM:

Producer/Consumer using ITC

ALGORITHM:

Step 1: Start the program.
Step 2: Define a class that contains two threads that will simulate producer and consumer.
Step 3: Define a class ‘Producer’ and ‘Consumer’ which will contain a LinkedList of integers.
Step 4: Define a method produce() that produces items till its capacity is reached and notify the consumer thread to start consuming.
Step 5: Define a method consume() that consumes items till the list is empty and notify the producer thread to start producing.
Step 6: Stop the program.

CODE:

CO4Q7.java	package javaprograms;
	//Producer/Consumer using ITC
	
	import java.util.ArrayList;
	import java.util.List;
	
	class Producer implements Runnable
	{
	List<Integer> flist;
	int max_size = 5;
	int i=0;
	Producer(List<Integer> flist)
	{
	this.flist = flist;
	}
	@Override

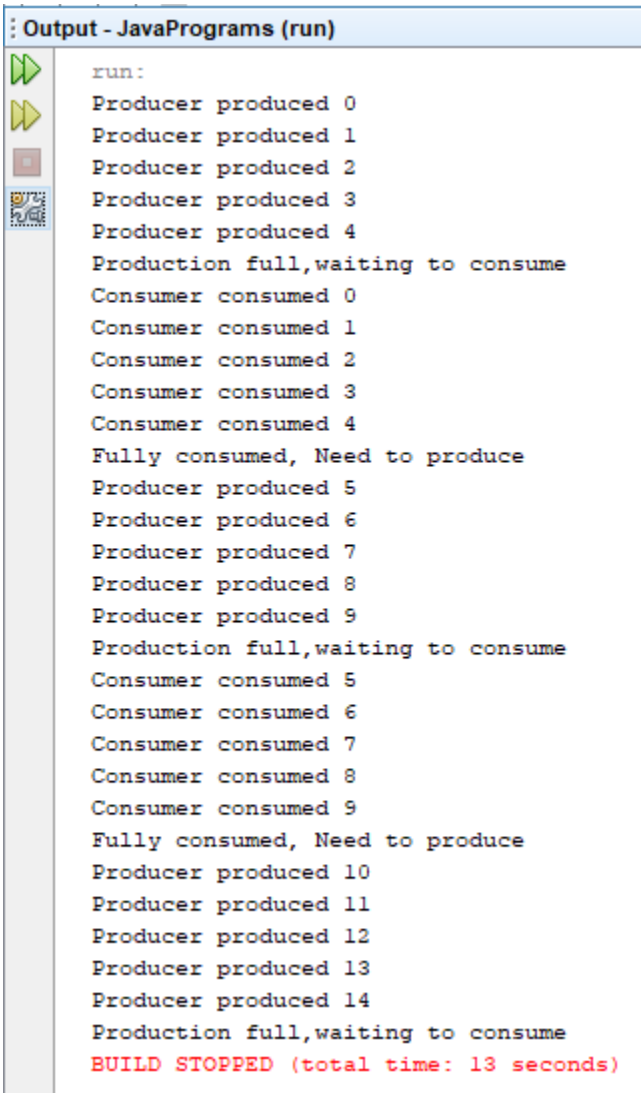
```
    public void run()
    {
        while(true)
        {
            try
            {
                produce(i++);
            } catch (Exception e)
            {
                System.out.println("Intteruption "+e);
            }
        }
    }
    public void produce(int i) throws InterruptedException
    {
        synchronized (flist)
        {
            while(flist.size()==max_size)
            {
                System.out.println("Production full,waiting to consume");
                flist.wait();
            }
        }
        synchronized(flist)
        {
            System.out.println("Producer produced "+i);
            flist.add(i);
            flist.notify();
        }
    }

}
class Consumer implements Runnable
{
    List<Integer> flist;
    Consumer(List<Integer> flist)
    {
        this.flist = flist;
    }

    @Override
    public void run()
    {
        while(true)
        {
            try
```

```
            {
                consume();
            } catch (Exception e)
            {
                System.out.println("Exception "+e);
            }
        }
    }
    public void consume() throws InterruptedException
    {
        synchronized (flist)
        {
            while(flist.isEmpty())
            {
                System.out.println("Fully consumed, Need to produce");
                flist.notify();
                Thread.sleep(500);
                flist.wait();
            }
        }
        synchronized(flist)
        {
            Thread.sleep(1000);
            System.out.println("Consumer consumed "+flist.remove(0));
        }
    }

}
public class CO4Q7 {
    public static void main(String[] args)
    {
        List<Integer> flist = new ArrayList<Integer>();
        Thread th1 = new Thread(new Producer(flist));
        Thread th2 = new Thread(new Consumer(flist));
        th1.start();
        th2.start();
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:


Output - JavaPrograms (run)
run:
Producer produced 0
Producer produced 1
Producer produced 2
Producer produced 3
Producer produced 4
Production full,waiting to consume
Consumer consumed 0
Consumer consumed 1
Consumer consumed 2
Consumer consumed 3
Consumer consumed 4
Fully consumed, Need to produce
Producer produced 5
Producer produced 6
Producer produced 7
Producer produced 8
Producer produced 9
Production full,waiting to consume
Consumer consumed 5
Consumer consumed 6
Consumer consumed 7
Consumer consumed 8
Consumer consumed 9
Fully consumed, Need to produce
Producer produced 10
Producer produced 11
Producer produced 12
Producer produced 13
Producer produced 14
Production full,waiting to consume
BUILD STOPPED (total time: 13 seconds)

PROGRAM NO-8

AIM:

Program to create a generic stack and do the Push and Pop operations.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class '*StackElement*' that contains methods to push, pop and display the stack elements.

Step 4: Display the results.

Step 5: Stop the program.

CODE:

CO4Q8.java

```
class StackElement<T>
{
   T value;
   StackElement<T> next;
   public StackElement(T value,StackElement<T> next)
   {
      this.value = value;
      this.next = next;
   }
   public StackElement<T> getNext()
   {
      return next;
   }
   public T getValue()
   {
      return value;
   }

}
public class CO4Q8 <T>
{
   int size;
```

```
    StackElement<T> top;
    public CO4Q8()
    {
        size = 0;
        top = null;
    }
    public void push(T newValue)
    {
        StackElement<T> newElement = new
StackElement<T>(newValue,top);
        top = newElement;
        size++;
    }
    public T pop()
    {
        StackElement<T> oldTop = top;
        if(size==0)
        {
            return null;
        }
        top = top.getNext();
        size--;
        return oldTop.getValue();
    }

    public static void main(String[] args)
    {
        CO4Q8 <String> strStack = new CO4Q8<String>();
        strStack.push("Apple");
        strStack.push("Mango");
        strStack.push("Watermelon");
        strStack.push("Cherry");
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
    }
}
```

PROGRAM NO-9

AIM:

Using generic method, perform Bubble sort.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class using generics

Step 3: Define a function '*bubblesort*'.

Step 6: Read the numbers and perform bubblesort.

Step 7: Stop the program.

CODE:

CO4Q9.java

```
import java.util.*;
public class CO4Q9<T extends Comparable<? super T>>
{
   T[] array;
   CO4Q9(T[] array)
   {
      this.array = array;
   }
   public T[] bubbleSort()
   {
      for(int i = array.length; i>1; i--)
      {
         for(int j=0; j<i-1; j++)
         {
            if(array[j].compareTo(array[j+1])>0)
            {
               swap(j,array);
            }
         }
      }
      return array;
```

```
    }
    public void swap(int index, T[] arr)
    {
        T temp = arr[index];
        arr[index] = arr[index+1];
        arr[index] = temp;
    }
    public static void main(String[] args)
    {
        Integer[] intArr = {31,2,53,4,25};
        CO4Q9<Integer> BubbleSort = new CO4Q9<Integer>(intArr);
        Integer[] SortedArray = BubbleSort.bubbleSort();
        System.out.println("Sorted Array:- "+Arrays.toString(SortedArray));
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Sorted Array:- [31, 2, 53, 4, 25]
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-10

AIM:

To maintain a list of Strings using ArrayList from collection framework, perform built-in operations.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main() to implement the concept.

Step 3: Declare an ArrayList of strings named '*fruits*' and start adding elements into it.

Step 4: Execute different ArrayList methods and display the results.

Step 5: Stop the program.

CODE:

CO4Q10.java

```
import java.util.*;

public class CO4Q10 {
   public static void main(String[] args) {
      ArrayList<String> fruits = new ArrayList<String>();
      fruits.add("Apple");
      fruits.add("Strawberry");
         fruits.add("Mango");
         fruits.add("Pineapple");
         fruits.add("Banana");
         fruits.add(4, "Grapes");
      System.out.println("Fruits-");
      for(String str: fruits)
         System.out.println(str+" ");
      fruits.remove("Banana");
      fruits.remove(1);
      System.out.println("\t**********");
      System.out.println("Fruits after items removed");
      for(String str: fruits)
         System.out.println(str+" ");
```

```
        Collections.sort(fruits);
        System.out.println("\t**********");
        System.out.println("Sorted-");
        for(String str: fruits)
           System.out.println(str+" ");
        System.out.println("\t**********");
        System.out.println("Total fruits-"+fruits.size());
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Fruits-
Apple
Strawberry
Mango
Pineapple
Grapes
Banana
        **********
Fruits after items removed
Apple
Mango
Pineapple
Grapes
        **********
Sorted-
Apple
Grapes
Mango
Pineapple
        **********
Total fruits-4
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-11

AIM:

To write a program to remove all the elements from a linked list.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main() to implement the concept.

Step 3: Create a LinkedList.

Step 4: Using add() to add to LinkedList

Step 5: Using clear() to remove all from the LinkedList

Step 6: Stop the program.

CODE:

CO4Q11.java

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


public class CO4Q11
{

    public static void main(String[] args) {
        int n;
        String data;
        LinkedList<String> ll = new LinkedList<String>();
        System.out.println("Enter the number of data");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        System.out.println("Enter the data");
        sc.nextLine();
        for(int i=0;i<n;i++)
        {
            data = sc.nextLine();
            ll.add(data);
        }
```

```
        System.out.println("LinkedList: "+ll);
        System.out.println("Removing all the elements....");
        ll.clear();
        System.out.println(ll);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
Enter the number of data
5
Enter the data
Apple
Mango
Orange
Grape
Kiwi
LinkedList: [Apple, Mango, Orange, Grape, Kiwi]
Removing all the elements....
[]
BUILD SUCCESSFUL (total time: 45 seconds)
```

PROGRAM NO-12

AIM:

To write a program to remove an object from the Stack when the position is passed as parameter.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a Stack of Strings and start adding elements into it using add() method.

Step 5: Using remove(index) method, remove the element and display the updated stack.

Step 6: Stop the program.

CODE:

CO4Q12.java

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

public class CO4Q12 {
   public static void main(String[] args) {
      int n;
      String str;
      Stack<String> s = new Stack<String>();
      System.out.println("Enter the number of elements:");
      Scanner sc = new Scanner(System.in);
      n = sc.nextInt();
      sc.nextLine();
      System.out.println("Enter the elements:");
      for(int i=0;i<n;i++)
      {
         str = sc.nextLine();
         s.add(str);
      }
      System.out.println("\nStack elements:"+s);
      System.out.println("\nTop element:"+s.peek());
      System.out.println("Popped element:"+s.pop());
```

```
        System.out.println("Stack elements after popped:"+s);
        System.out.println("\nRemove Element at position 1:"+s.remove(0));
        System.out.println("Stack elements after removed:"+s);
        System.out.println("\nRemove Luke Skywalker:");
        s.remove("Luke Skywalker");
        System.out.println("Stack elements after removing Luke
Skywalker:"+s);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the number of elements:
7
Enter the elements:
Han Solo
Leia
Luke Skywalker
Chewbacca
Yoda
R2-D2
C-3PO

Stack elements:[Han Solo, Leia, Luke Skywalker, Chewbacca, Yoda, R2-D2, C-3PO]

Top element:C-3PO
Popped element:C-3PO
Stack elements after popped:[Han Solo, Leia, Luke Skywalker, Chewbacca, Yoda, R2-D2]

Remove Element at position 1:Han Solo
Stack elements after removed:[Leia, Luke Skywalker, Chewbacca, Yoda, R2-D2]

Remove Luke Skywalker:
Stack elements after removing Luke Skywalker:[Leia, Chewbacca, Yoda, R2-D2]
BUILD SUCCESSFUL (total time: 43 seconds)
```

PROGRAM NO-13

AIM:

To write a program to demonstrate the creation of queue object using the PriorityQueue class.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a *PriorityQueue* of Strings and start adding elements into it using *add()* method.

Step 4: Iterate and display the queue elements.

Step 5: Stop the program.

CODE:

CO4Q13.java

```
import java.util.Iterator;
import java.util.PriorityQueue;
import java.util.Scanner;


public class CO4Q13 {
    public static void main(String[] args) {
        int n;
        String str;
        PriorityQueue<String> pq = new PriorityQueue<String>();
        System.out.println("Enter the no. of data:");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        sc.nextLine();
        System.out.println("Enter the data:");
        for(int i=0;i<n;i++)
        {
            str = sc.nextLine();
            pq.add(str);
        }
        Iterator itr = pq.iterator();
```

```
        System.out.println("\nPriority Queue\n");
        while(itr.hasNext())
            System.out.println(itr.next()+" ");
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the no. of data:
10
Enter the data:
Inception
Interstellar
Finding Neverland
Forrest Gump
The Professor
Edge of Tomorrow
Sweeney Todd
The Lone Ranger
Edward Scissorhands
Ready Player One

Priority Queue

Edge of Tomorrow
Edward Scissorhands
Finding Neverland
Forrest Gump
Ready Player One
Inception
Sweeney Todd
The Lone Ranger
Interstellar
The Professor
BUILD SUCCESSFUL (total time: 1 minute 9 seconds)
```

PROGRAM NO-14

AIM:

To write a program to demonstrate the addition and deletion of elements in deque.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a *Dequeue* of Strings and perform the following methods:

- addFirst() - inserts the element passed in the parameter to the front of the *Deque* if there is space.
- addLast() - inserts the element passed in the parameter to the end of the *Deque* if there is space.
- removeFirst() - removes the first element of Deque.
- removeLast()- removes the last element of Deque.

Step 4: Display the results.

Step 5: Stop the program.

CODE:

CO4Q14.java	import java.util.Deque;
	import java.util.LinkedList;
	import java.util.Scanner;
	
	public class CO4Q14 {
	public static void main(String[] args) {
	int ch;
	String data;
	Deque<String> dq = new LinkedList<String>();
	Scanner sc = new Scanner(System.in);
	do

```
        {
            System.out.println("\nChoose a number...");
            System.out.println("1.Insert the element at first");
            System.out.println("2.Insert the element at last");
            System.out.println("3.Delete the element at first");
            System.out.println("4.Delete the element at last");
            System.out.println("5.Display");
            System.out.println("6.Exit");
            System.out.println("\nEnter the choice:");
            ch = sc.nextInt();
            sc.nextLine();
            switch(ch)
            {
                case 1: System.out.println("Enter the element to be inserted at
first:");
                        data = sc.nextLine();
                        dq.addFirst(data);
                        break;
                case 2: System.out.println("Enter the element to be inserted at
last:");
                        data = sc.nextLine();
                        dq.addLast(data);
                        break;
                case 3: System.out.println("Element deleted from the first
position");
                        dq.removeFirst();
                        break;
                case 4: System.out.println("Element deleted from the last
position");
                        dq.removeLast();
                        break;
                case 5: System.out.println("Elements:");
                        System.out.println(dq);
                        break;
                case 6: System.exit(0);
                        break;
                default:System.out.println("Invalid choice...");
            }
        }while(true);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
1
Enter the element to be inserted at first:
Iron Man

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
2
Enter the element to be inserted at last:
Captain America

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
1
Enter the element to be inserted at first:
Black Widow

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
2
Enter the element to be inserted at last:
Thor
```

```
Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
1
Enter the element to be inserted at first:
Hawkeye

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
2
Enter the element to be inserted at last:
Hulk

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
5
Elements:
[Hawkeye, Black Widow, Iron Man, Captain America, Thor, Hulk]

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
3
Element deleted from the first position
```

```
Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
5
Elements:
[Black Widow, Iron Man, Captain America, Thor, Hulk]

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
4
Element deleted from the last position

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
5
Elements:
[Black Widow, Iron Man, Captain America, Thor]
```

PROGRAM NO-15

AIM:

To write a program to demonstrate the creation of Set object using the LinkedHashset class.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare an *LinkedHashSet* of strings.

Step 4: Start adding elements into it using add() method.

Step 5: Execute some LinkedHashSet methods and display the results.

Step 6: Stop the program.

CODE:

CO4Q15.java

```
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Scanner;

public class CO4Q15 {
   public static void main(String[] args) {
      int n;
      String str;
      LinkedHashSet<String> lsh = new LinkedHashSet<String>();
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the no. of values");
      n = sc.nextInt();
      sc.nextLine();
      System.out.println("Enter the values");
      for(int i=0;i<n;i++)
      {
         str = sc.nextLine();
         lsh.add(str);
      }
      System.out.println("\nOriginal LinkedHashSet:"+lsh);
      System.out.println("Removed 'Rachel' from
```

```
LinkedHashSet:"+lsh.remove("Rachel"));
        System.out.println("Size of LinkedHashSet:"+lsh.size());
        System.out.println("Checking if 'Joey' is
present:"+lsh.contains("Joey"));
        System.out.println("Final LinkedHashSet:"+lsh);
        System.out.println("\nIterating...");
        Iterator itr = lsh.iterator();
        while(itr.hasNext())
           System.out.println(itr.next());

    }

}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the no. of values
6
Enter the values
Chandler
Joey
Ross
Rachel
Phoebe
Monica

Original LinkedHashSet:[Chandler, Joey, Ross, Rachel, Phoebe, Monica]
Removed 'Rachel' from LinkedHashSet:true
Size of LinkedHashSet:5
Checking if 'Joey' is present:true
Final LinkedHashSet:[Chandler, Joey, Ross, Phoebe, Monica]

Iterating...
Chandler
Joey
Ross
Phoebe
Monica
BUILD SUCCESSFUL (total time: 23 seconds)
```

PROGRAM NO-16

AIM:

To write a Java program to compare two hash sets.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare two HashSets (s1, s2) of strings and start adding elements into them using add() method.

Step 4: Display the elements of both the hashsets.

Step 5: Perform set operations.

- addAll() – gets all the elements
- retainAll() – gets the elements which are common
- removeAll() – gets the difference

Step 6: Display the results.

Step 7: Stop the program.

CODE:

CO4Q16.java	import java.util.*;
	
	public class CO4Q16 {
	public static void union(Set<String> s1,Set<String> s2)
	{
	Set<String> su = new HashSet<>(s1);
	su.addAll(s2);
	System.out.println("Union:"+su);
	}
	public static void intersection(Set<String> s1,Set<String> s2)
	{
	Set<String> su = new HashSet<>(s1);

```
        su.retainAll(s2);
        System.out.println("Intersection:"+su);
    }
    public static void difference(Set<String> s1,Set<String> s2)
    {
        Set<String> su1 = new HashSet<>(s1);
        su1.removeAll(s2);
        Set<String> su2 = new HashSet<>(s2);
        su2.removeAll(s1);
        System.out.println("Difference:First HashSet- "+su1+" and Second
HashSet- "+su2);
    }
    public static void main(String[] args) {
        Set<String> s1 = new HashSet<>();
        s1.add("Tokyo");
        s1.add("Berlin");
        s1.add("Rio");
        Set<String> s2 = new HashSet<>();
        s2.add("Tokyo");
        s2.add("Berlin");
        s2.add("Denver");
        System.out.println("Elements in first HashSet:"+s1);
        System.out.println("Elements in second HashSet:"+s2);
        union(s1,s2);
        intersection(s1,s2);
        difference(s1,s2);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Elements in first HashSet:[Rio, Berlin, Tokyo]
Elements in second HashSet:[Berlin, Tokyo, Denver]
Union:[Rio, Tokyo, Berlin, Denver]
Intersection:[Tokyo, Berlin]
Difference:First HashSet- [Rio] and Second HashSet- [Denver]
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-17

AIM:

To write a program to demonstrate the working of Map interface by adding, changing and removing elements.

ALGORITHM:

Step.1: Start the program.

Step.2: Define a class with a main () to implement the concept.

Step.3: Declare a *HashMap* and insert elements into it using *put()* method.

Step.4: To update any value in hashmap using *hashmap.put()* or *hashmap.replace()*.

Step.5: Using *remove(key)* method, remove elements from hashmap.

Step.6: Display the results.

Step.7: Stop the program.

CODE:

CO4Q17.java	import java.util.HashMap;
	import java.util.Map;
	
	public class CO4Q17 {
	  public static void main(String[] args) {
	  Map<Integer, String> hm = new HashMap<>();
	  hm.put(1,"The Professor(2018)");
	  hm.put(2,"Jojo Rabbit(2019)");
	  hm.put(3,"Just Mercy(2019)");
	  hm.put(4,"Back to the Future(1985)");
	  System.out.println("Map:"+hm);
	  hm.put(4,"Back to the Future 2(1989)");
	  System.out.println("Updated Map:"+hm);
	  hm.remove(3);
	  System.out.println("Final Map:"+hm);
	  }
	}

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Map:{1=The Professor(2018), 2=Jojo Rabbit(2019), 3=Just Mercy(2019), 4=Back to the Future(1985)}
Updated Map:{1=The Professor(2018), 2=Jojo Rabbit(2019), 3=Just Mercy(2019), 4=Back to the Future 2(1989)}
Final Map:{1=The Professor(2018), 2=Jojo Rabbit(2019), 4=Back to the Future 2(1989)}
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-18

AIM:

To write a program to Convert HashMap to TreeMap.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a *HashMap* and insert elements into it using *put()* method.

Step 4: Convert the above HashMap to TreeMap:

Map<i,s> treeMap = new TreeMap<>();

treeMap.putAll(hashMap);

Step 5: Display the resultant treeMap.

Step 6: Stop the program.

CODE:

CO4Q18.java

```
import java.util.*;

public class CO4Q18 {
   public static <i,s> Map<i,s> convert(Map<i,s> hashmap)
   {
      Map<i,s> treemap = new TreeMap<>();
      treemap.putAll(hashmap);
      return treemap;
   }
   public static void main(String[] args) {
      Map<String,Integer> hashmap = new HashMap<>();
      hashmap.put("Johnny Depp",1);
      hashmap.put("Tom Cruise",1);
      hashmap.put("Jackie Chan",1);
      hashmap.put("Keanu Reeves",1);
```

```
        System.out.println("HashMap:"+hashmap);
        Map<String,Integer>treemap = convert(hashmap);
        System.out.println("TreeMap:"+treemap);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
HashMap:{Tom Cruise=1, Johnny Depp=1, Keanu Reeves=1, Jackie Chan=1}
TreeMap:{Jackie Chan=1, Johnny Depp=1, Keanu Reeves=1, Tom Cruise=1}
BUILD SUCCESSFUL (total time: 0 seconds)
```