**Program 1**

07/06/2021

**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 package contain functions to assign calculate and display.

Step 3: Read inputs from user and assign values to objects.

Step 4: Perform desired operations.

Step 5: Print the Outputs.

Step 6: Stop

**Source Code:**

***test.java***

package Main;

import java.util.Scanner;

import packoops.Dimension;

public class test {

public static void main(String[] args) {

Scanner obk=new Scanner(System.in);

Dimension obj=new Dimension();

int ch=0;

while(ch<5) {

System.out.println("Choose the Shape to find the area:"+"\n 1.Rectangle \t 2.Triangle \n 3.Circle \t 4.Sqaure \t 5.Exit");

ch=obk.nextInt();

switch(ch) {

case 1:

obj.rectangle();

break;

case 2:

obj.triangle();

break;

case 3:

obj.circle();

break;

case 4:

obj.square();

break;

case 5: break;

default:

System.out.println("invalid choice");

break;

}

}

}

}

***Dimension.java***

package packoops;

import java.util.Scanner;

interface calculation{

public void rectangle();

public void triangle();

public void circle();

public void square();

}

public class Dimension implements calculation {

double area;

Scanner obj1=new Scanner(System.*in*);

public void rectangle() {

int l,b;

System.*out*.println("Enter the length of the rectangle:");

l=obj1.nextInt();

System.*out*.println("Enter the Breath of the rectangle:");

b=obj1.nextInt();

area=l\*b;

System.*out*.println("Area:"+area+"\n");

}

public void triangle() {

int h,b;

System.*out*.println("Enter the base of the Triangle:");

b=obj1.nextInt();

System.*out*.println("Enter the height of the Triangle:");

h=obj1.nextInt();

area=(h\*b)/2;

System.*out*.println("Area:"+area+"\n");

}

public void circle() {

float r;

System.*out*.println("Enter the Radius of the Circle:");

r=obj1.nextInt();

area=3.14\*r\*r;

System.*out*.println("Area:"+area+"\n");

}

public void square() {

int s;

System.*out*.println("Enter the Side of the Square:");

s=obj1.nextInt();

area=s\*s;

System.*out*.println("Area:"+area+"\n");

}

}

# Output :



**Program 2**

10/06/2021

**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: Define a package contain functions to assign calculate and display.

Step 3: Read inputs from user and assign values to objects.

Step 4: Perform desired operations.

Step 5: Print the Outputs.

Step 6: Stop

**Source Code:**

***oper.java***

package Main;

import packoops.Operation;

import Graphics.Dimension;

import java.util.Scanner;

public class oper {

public static void main(String[] args) {

Operation obj=new Operation();

Scanner obk=new Scanner(System.in);

System.out.println("Enter the 1st number");

int a=obk.nextInt();

System.out.println("Enter the 2st number");

int b=obk.nextInt();

int ch=0;

obj.data(a, b);

while(ch<5) {

System.out.println("Choose the Operation :"+"\n 1.Addition \t 2.Subtration \n 3.Multiplication \t 4.Division \t 5.Exit");

ch=obk.nextInt();

switch(ch) {

case 1:

obj.add();

break;

case 2:

obj.sub();

break;

case 3:

obj.mult();

break;

case 4:

obj.div();

break;

case 5: break;

default:

System.out.println("invalid choice");

break;

}

}

}

}

***Operation.java***

package packoops;

interface arith {

public void add();

public void sub();

public void mult();

public void div();

}

public class Operation implements arith{

int x,y;

public void data(int a,int b) {

x=a;

y=b;

}

public void add() {

int s;

s=x+y;

System.*out*.println("Addition :"+s);

}

public void sub() {

int m;

m=x-y;

System.*out*.println("Subtration :"+m);

}

public void mult() {

int p;

p=x\*y;

System.*out*.println("Multiplication :"+p);

}

public void div() {

float d;

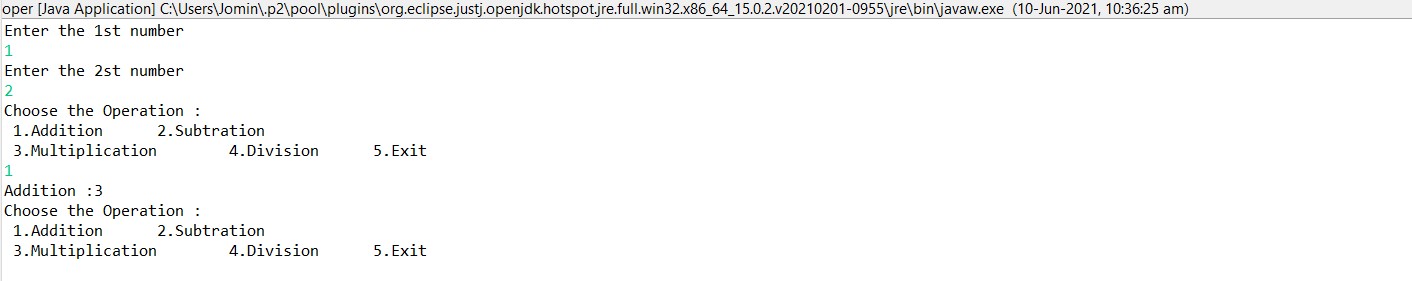
d=x/y;

System.*out*.println("Division :"+d);

}

}

**Output:**

****

**Program 3**

10/06/2021

**Aim:** Write a user defined exception class to authenticate the

user name and password.

**Algorithm:**

Step 1: Start

Step 2: Read inputs as username and password.

Step 3: Verify the username and password.

Step 4: If its true; Print Authentication successful.

Step 5: Else print invalid.

Step 6: Stop

**Source Code:**

package packoops;

import java.util.Scanner;

class authException extends Exception

{

public authException(String s) {

super(s);

}

}

public class login

{

public static void main(String[] args) {

String username = "jomin";

String passcode = "123";

String user\_name,password;

Scanner sc = new Scanner(System.*in*);

try

{

System.*out*.println("Username:");

user\_name = sc.nextLine();

System.*out*.println("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)

{

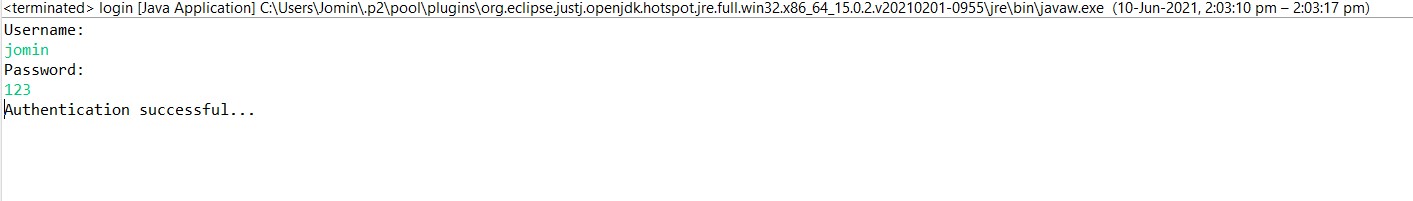
System.*out*.println("Exception caught "+e);

}

}

}

**Output:**

****

**Program 4**

10/06/2021

**Aim:** Find the average of N positive integers, raising a user

defined exception for each negative input.

**Algorithm:**

Step 1: Start

Step 2: Enter a limit n.

Step 3: Read n elements.

Step 4: Iterate the loop .

Step 5: if(num[i]>0) then

Step 6: total += num[i] ,count++;

Step 7: Else Print error message

Step 8: Average=total/count

Step 9: Print Average

Step 10: Stop

**Source Code:**

package packoops;

import java.util.Scanner;

class NegativeIntegerException extends Exception

{

public NegativeIntegerException(String s)

{

super(s);

}

}

public class average {

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) {

}

}

}

**Output:**

****

**Program 5**

01/09/2021

**Aim:** Define 2 classes; one for generating multiplication table

of 5 and other for displaying first N prime numbers.

Implement using threads. (Thread class).

**Average:**

Step 1: Start

Step 2: Define a class for multiplication and another for prime numbers.

Step 3: Print the multiplication table of 5.

Step 4: Input the limit for prime numbers.

Step 5: Print prime numbers.

Step 6: Stop

**Source Code:**

package packoops;

import java.util.Scanner;

public class mult {

public static void main(String[] args) throws InterruptedException {

multiplication\_tbl a = new multiplication\_tbl();

prime\_num b = new prime\_num();

a.start();

a.*sleep*(200);

b.start();

b.*sleep*(200);

}

}

class multiplication\_tbl extends Thread{

public void run(){

int n=10;

Scanner sc= new Scanner(System.*in*);

System.*out*.println("Multiplication of Five");

for(int i=1;i<=10;i++)

{

System.*out*.println(+i+"x 5 =>"+(i\*5));

}

}

}

class prime\_num extends Thread{

public void run() {

int n,i=3,count,m;

Scanner sc= new Scanner(System.*in*);

System.*out*.println("Enter limit for prime numbers:");

n=sc.nextInt();

if(n >=1)

{

System.*out*.println("First prime" +n+ "numbers are:");

System.*out*.println("2");

for(count = 2; count <= n; i++)

{

for(m= 2; m < i;m++)

{

if(i%m == 0)

break;

}

if(m== i)

{

System.*out*.println(i);

count++;

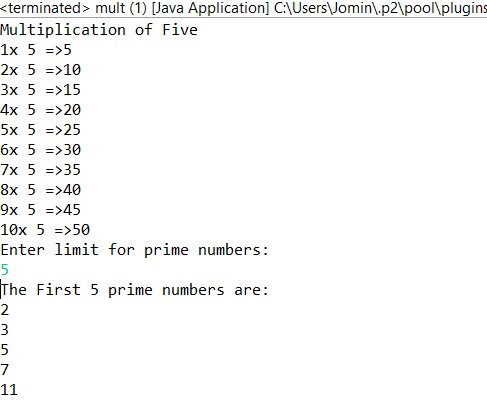
}

}

}

}}

**Output:**

****

**Program 6**

17/06/2021

**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

Step 2: Define a class for Fibonacci and another for even numbers.

Step 3: Print the Fibonnaci series of 20 number.

Step 4: Input the limit for even numbers.

Step 5: Print n even numbers.

Step 6: Stop

**Source Code:**

package packoops;

import java.util.Scanner;

class Fibonacci implements Runnable{

public void run(){

int a=0,b=1,c=0,l=20;

System.*out*.println("FIBONACCI SERIES UPTO "+l+": \n");

while (l>0)

{

System.*out*.print(c+" ");

a=b;

b=c;

c=a+b;

l=l-1;

if(l%10==0)

{

System.*out*.println("\n");

}

}

}

}

class EvenNumber implements Runnable{

public void run(){

int n;

Scanner sc=new Scanner(System.*in*);

System.*out*.println("Enter the limit : ");

n=sc.nextInt();

System.*out*.println("Even Numbers from 1 to "+n+"\n");

for(int i=1;i<=n;i++) {

if(i%2==0) {

System.*out*.println(i);

}

}

}

}

public class fib {

public static void main(String[] args) {

Fibonacci obj1=new Fibonacci();

Thread t1=new Thread(obj1);

t1.start();

EvenNumber obj2=new EvenNumber();

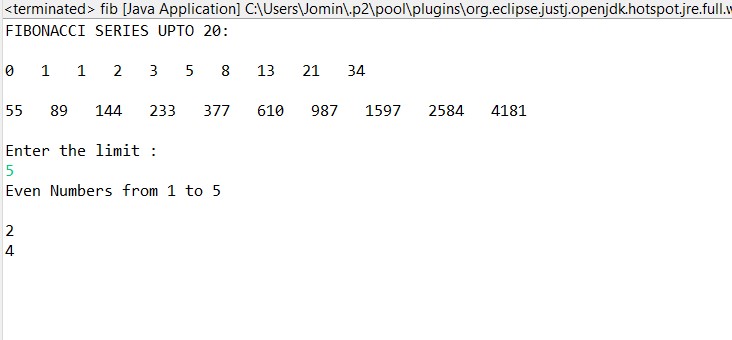
Thread t2=new Thread(obj2);

t2.start();

}

}

**Output:**

****

**Program 7**

21/06/2021

**Aim:** Producer/Consumer using ITC.

**Algorithm:**

**Source Code:**

package packoops;

import java.util.ArrayList;

import java.util.List;

public class sales {

public static void main(String[] args) {

List<Integer> sharedList = new ArrayList<Integer>();

Thread t1 = new Thread(new Thread( new Producer(sharedList)));

Thread t2 = new Thread(new Thread( new Consumer(sharedList)));

t1.start();

t2.start();

}

}

class Producer implements Runnable

{

List<Integer> sharedList = null;

final int MAX\_SIZE=5;

private int i = 0;

public Producer(List<Integer> sharedList) {

super();

this.sharedList = sharedList;

}

@Override

public void run() {

while(true) {

try

{

produce(i++);

}catch(InterruptedException exception) {

}

}

}

public void produce(int i) throws InterruptedException {

synchronized (sharedList) {

while(sharedList.size() == MAX\_SIZE) {

System.out.println("SharedList is full !! Waiting for consumer to consume....");

sharedList.wait();

}

}

synchronized (sharedList) {

System.out.println("producer produced the element"+i);

sharedList.add(i);

Thread.sleep(100);

sharedList.notify();

}

}

}

class Consumer implements Runnable

{

List<Integer> sharedList = null;

public Consumer(List<Integer> sharedList) {

super();

this.sharedList = sharedList;

}

@Override

public void run() {

while(true) {

try

{

consume();

}catch(InterruptedException exception) {

}

}

}

public void consume() throws InterruptedException {

synchronized (sharedList) {

while(sharedList.isEmpty()) {

System.out.println("SharedList is Empty !! Waiting for producer to produce the object....");

sharedList.wait();

}

}

synchronized (sharedList) {

Thread.sleep(1000);

System.out.println("consumed the Element"+sharedList.remove(0));

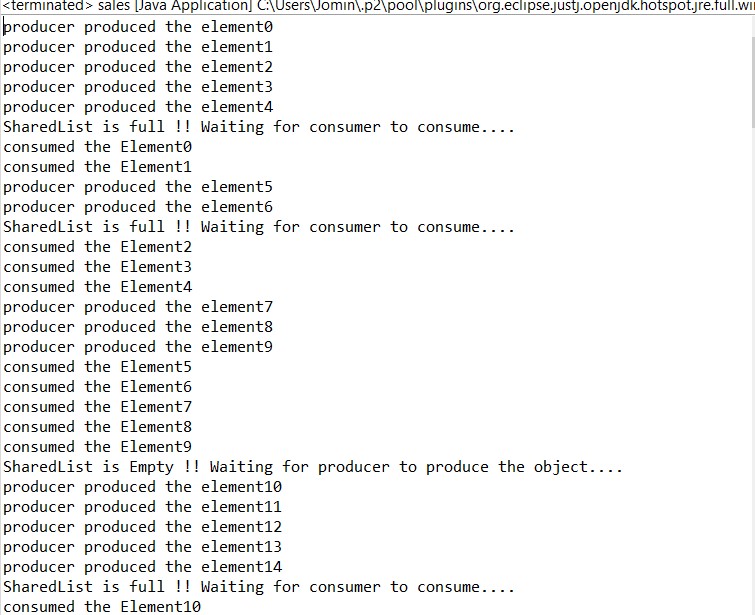
sharedList.notify();

}

}

}

**Output:**

****

**Program 8**

24/06/2021

**Aim:** Program to create a generic stack and do the Push and

Pop operations.

**Algorithm:**

Step 1: Start

Step 2: Define a stack .

Step 3: Push();

If top<n

Top++

Stack[top]=n.

Step 4: Pop()

If top!=1

.top--

Step 5: Print the stack.

Step 6: Stop

**Source Code:**

package Graphics;

import java.util.\*;

public class Stack {

int top=-1,ch,item,i;

int a[] = new int[10];

Scanner sc = new Scanner(System.*in*);

public static void main(String[] args) {

Stack st = new Stack ();

st.stack();

}

public void stack(){

System.*out*.println("Enter the size of the stack:");

int N=sc.nextInt();

do

{

System.*out*.println("\nEnter your choice ");

System.*out*.println("\n 1.Push \n 2.Pop \n 3.Display \n");

ch=sc.nextInt();

switch(ch){

case 1:

System.*out*.println("Enter the element to be inserted:");

item=sc.nextInt();

if(top==N-1) {

System.*out*.println("Stack overflow!");

}

else {

top++;

a[top]=item;

}

break;

case 2:

if(top==-1) {

System.*out*.println("Stack is empty!!");

}

else {

item=a[top];

top--;

System.*out*.println("Deleted element is:" +item);

}

break;

case 3:

if(top==-1)

{

System.*out*.println("Stack is empty!!");

}

else

{

System.*out*.println("Displaying elements in the Stack:\n");

for(int i=top;i>=0;i--)

{

System.*out*.println(a[i]);

}

}

break;

case 4: System.*out*.println("Enter a valid choice");

}

}

while(ch!=5);

}

}

**Output:**

****

**Program 9**

24/06/2021

**Aim:** Using generic method perform Bubble sort.

**Algorithm:**

Step 1: Start

Step 2: Look at the first number in the list.

Step 3: Compare the current number with the next number.

Step 4: Is the next number smaller than the current number? If so, swap the two numbers around. If not, do not swap.

Step 5: Move to the next number along in the list and make this the current number.

Step 6: Repeat from step 2 until the last number in the list has been reached.

Step 7: If any numbers were swapped, repeat again from step 1.

Step 8: If the end of the list is reached without any swaps being made, then the list is ordered and the algorithm can stop.

Step 9: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class bubblesort {

public static void main(String[] args) {

int num, i, j, temp;

Scanner x = new Scanner(System.*in*);

System.*out*.println("Enter the limit to be sort:");

num = x.nextInt();

int array[] = new int[num];

System.*out*.println("Enter " + num + " integers: ");

for (i = 0; i < num; i++)

array[i] = x.nextInt();

for (i = 0; i < ( num - 1 ); i++) {

for (j = 0; j < num - i - 1; j++) {

if (array[j] > array[j+1])

{

temp = array[j];

array[j] = array[j+1];

array[j+1] = temp;

}

}

}

System.*out*.println("Sorted list of integers:");

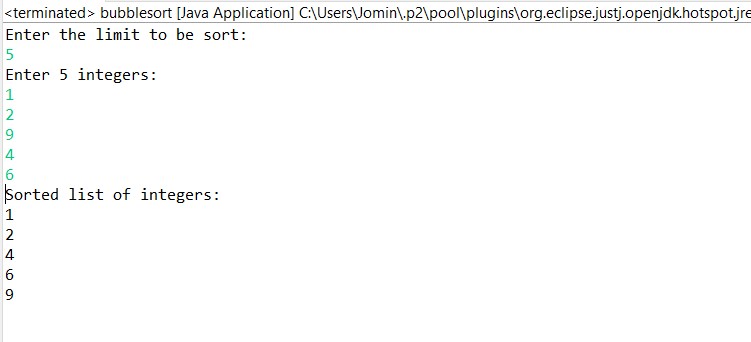
for (i = 0; i < num; i++)

System.*out*.println(array[i]);

}

}

**Output:**

****

**Program 10**

24/06/2021

**Aim:** Maintain a list of Strings using Array List from

collection framework, perform built-in operations.

**Algorithm:**

Step 1: Start

Step 2: Define an array.

Step 3: Define objects to array.

Step 4: Perform operation on array.

Step 5: Print the array after operations.

Step 6: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class Arraylist

{

public static void main(String[] args)

{

ArrayList<String> obj=new ArrayList<String>();

obj.add("one");

obj.add("Three");

obj.add("four");

obj.add("five");

obj.add(1,"two");

System.*out*.println("\n Array list after operation:");

for(String str:obj)

System.*out*.println(str);

obj.remove("five");

obj.remove(3);

System.*out*.println("\n Array list after remove operation");

for(String str:obj)

System.*out*.println(str);

Collections.*sort*(obj);

System.*out*.println(" \n Array list after sorting");

for(String str : obj)

System.*out*.println(str);

System.*out*.println("\n Object at index 2:"+obj.get(2));

System.*out*.println("\n Six is in the arraylist:"+obj.contains("six"));

System.*out*.println("\n Two is in the arraylist:"+obj.contains("one"));

System.*out*.println("\n Size of the arraylist :"+obj.size());

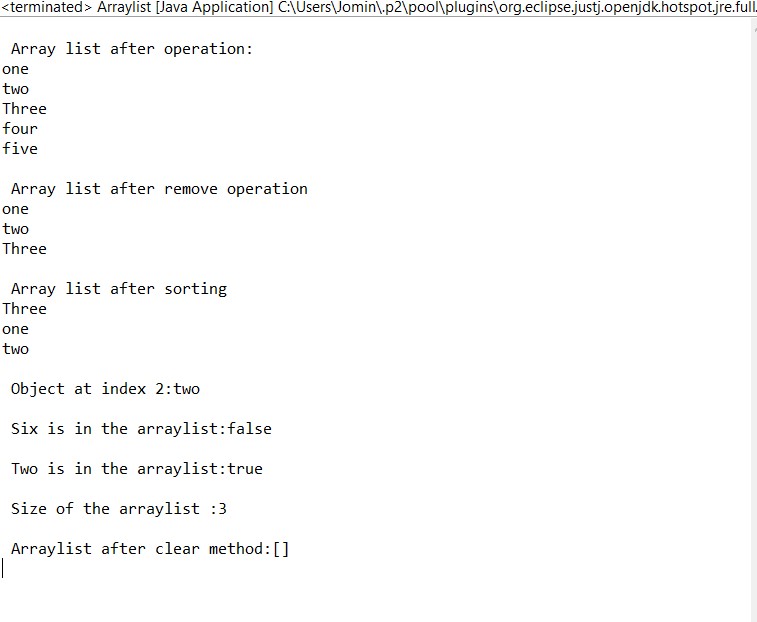
obj.clear();

System.*out*.println("\n Arraylist after clear method:"+obj);

}

}

**Output:**

****

**Program 11**

24/06/2021

**Aim:** Program to remove all the elements from a linked list.

**Algorithm:**

Step 1: Start

Step 2: Define a Linked list.

Step 3: Define objects to Linked list using add().

Step 4: Delete the elements in list using clear()

Step 5: Print the Outputs.

Step 6: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class linklstremove {

public static void main(String[] args) {

LinkedList<String> list = new LinkedList<String>();

list.add("A");

list.add("B");

list.add("C");

list.add("E");

list.add("F");

list.add("G");

System.*out*.println("The Original linked list: " + list);

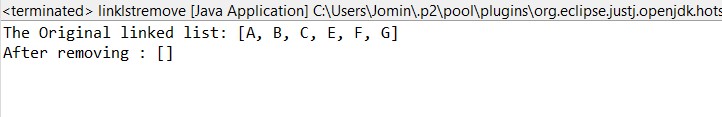
list.clear();

System.*out*.println("After removing : " + list);

}

}

**Output:**

****

**Program 12**

24/06/2021

**Aim:** Program to remove an object from the Stack when the

position is passed as parameter.

**Algorithm:**

Step 1: Start

Step 2: Define a Stack.

Step 3: Define objects to Stack using add().

Step 4: Enter the element to remove the item.

Step 5: Remove the item using remove(key).

Step 6: Print outputs

Step 7: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class stack {

public static void main(String[] args) {

Scanner s = new Scanner(System.*in*);

int i,n;

System.*out*.println("Enter the list of numbers to be added:");

int a=s.nextInt();

Stack <Integer> st = new Stack <Integer>();

for(i=1;i<=a;i++) {

System.*out*.println("Enter the "+(i)+"st number:");

n=s.nextInt();

st.add(n);

}

System.*out*.println("Original Stack:"+st);

System.*out*.println("\nEnter the index to be removed:");

int index = s.nextInt();

int rm\_ele = st.remove(index);

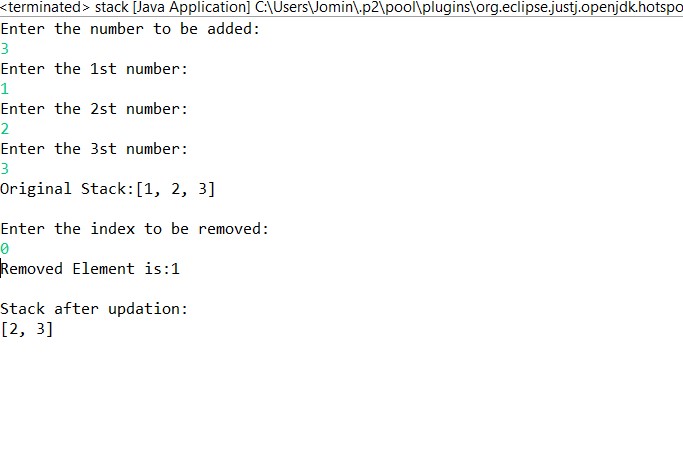
System.*out*.println("Removed Element is:"+rm\_ele);

System.*out*.println("\nStack after updation:\n"+st);

}

}

**Output:**

****

**Program 13**

24/06/2021

**Aim:** Program to demonstrate the creation of queue object

using the PriorityQueue class.

**Algorithm:**

Step 1: Start

Step 2: Define a Priority Queue.

Step 3: Enter the limit.

Step 4: Add n elements to Queue.

Step 5: Perform operations like remove(), add() etc.

Step 6: Print output.

Step 7: Stop

**Source Code:**

package packoops;

import java.util.Iterator;

import java.util.PriorityQueue;

import java.util.Scanner;

public class queuepri

{

public static void main(String args[])

{

PriorityQueue<String> queue=new PriorityQueue<String>();

Scanner sc=new Scanner(System.in);

System.out.println("Enter Number Of elements ");

int n=sc.nextInt();

System.out.println("Enter the elements ");

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

{

String st=sc.next();

queue.add(st);

}

System.out.println("head:"+queue.element());

System.out.println("head:"+queue.peek());

System.out.println("Iterating the queue elements\n ");

Iterator itr=queue.iterator();

while(itr.hasNext()){

System.out.println(itr.next());

}

queue.remove();

queue.poll();

System.out.println("After removing two elements \n");

Iterator<String> itr2=queue.iterator();

while(itr2.hasNext()){

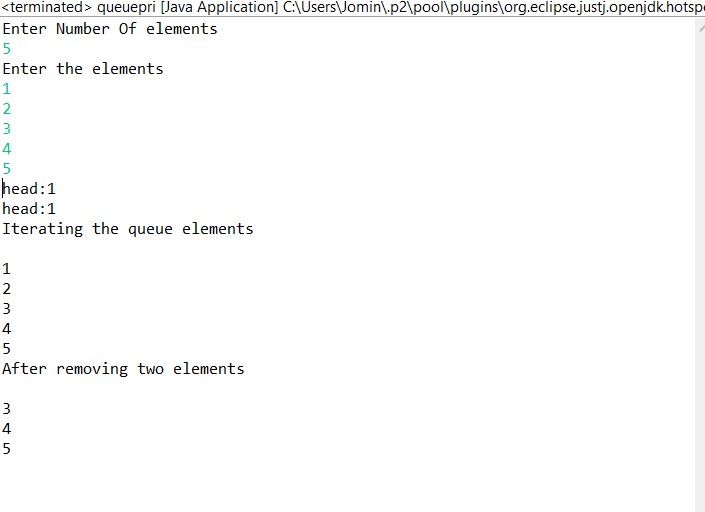
System.out.println(itr2.next());

}

}

}

**Output:**

****

**Program 14**

24/06/2021

**Aim:** Program to demonstrate the addition and deletion of

elements in deque.

**Algorithm:**

Step 1: Start

Step 2: Define an Array.

Step 3: Define objects to Array using add(),

Step 4: Perform desired operations.

Step 5: Remove the item using pop().

Step 6: Print outputs

Step 7: Stop

**Source Code:**

package packoops;

import java.util.ArrayDeque;

import java.util.Deque;

public class deque {

public static void main(String[] args) {

Deque<Integer> deque = new ArrayDeque<Integer>();

deque.add(1);

deque.add(2);

deque.add(3);

System.out.println("Inserting three elements : ");

for (Integer integer : deque) {

System.out.println(integer);

}

deque.pop();

System.out.println("After popping : ");

for (Integer integer : deque) {

System.out.println(integer);

}

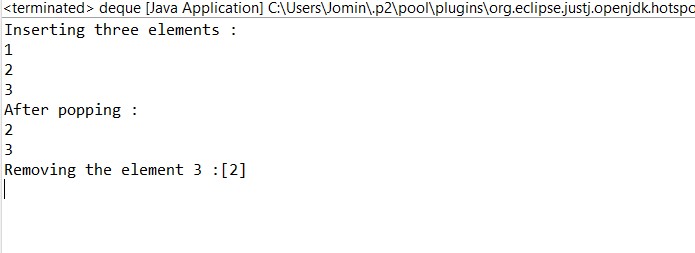
deque.remove(3);

System.out.println("Removing the element 3 :"+deque);

}

}

**Output:**

****

**Program 15**

24/06/2021

**Aim:** Program to demonstrate the creation of Set object using

the Linked Hashset class.

**Algorithm:**

Step 1: Start

Step 2: Define a Linkedhash set.

Step 3: Define objects to Linked hash set using add().

Step 4: Perform desired operations like printing the elements and size of the hash set.

Step 5: Remove the item using remove(key).

Step 6: Print outputs

Step 7: Stop

**Source Code:**

package packoops;

import java.util.LinkedHashSet;

import java.util.Scanner;

import java.util.Set;

import java.util.Iterator;

public class linkedhastset {

public static void main(String[] args) {

Set<Integer> s = new LinkedHashSet<Integer>();

System.out.println("Enter the limit to be added");

Scanner inp=new Scanner(System.in);

int n=inp.nextInt();

System.out.println("Enter the Number:");

while(n!=0) {

int e = inp.nextInt();

s.add(e);

n--;

}

System.out.println("Set is "+s);

System.out.println("Enter the number to be deleted:");

int d= inp.nextInt();

if(s.remove(d)) {

System.out.println("Removed!!"+s);

}

else {

System.out.println("Element is not in the set");

}

Iterator<Integer> l = s.iterator();

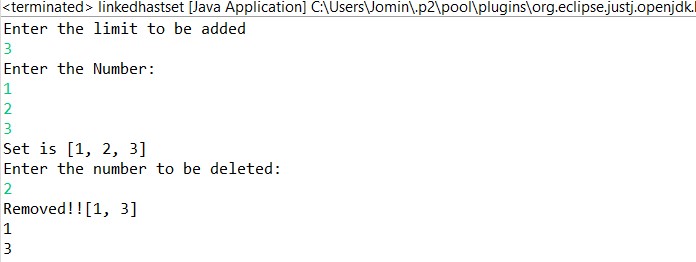
while(l.hasNext())

System.out.println(l.next());

}

}

**Output:**

****

**Program 16**

24/06/2021

**Aim:** Write a Java program to compare two hash set.

**Algorithm:**

Step 1: Start

Step 2: Define 2 hash sets h\_set and h\_set2.

Step 3: Define objects to hash h\_set and h\_set2 using add().

Step 4: Compare the hash sets; if its same.

Step 5: Print Yes

Step 6: Else Print No

Step 7: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class hashcomp {

public static void main(String[] args) {

HashSet<String> h\_set = new HashSet<String>();

h\_set.add("Red");

h\_set.add("Green");

h\_set.add("Black");

h\_set.add("White");

HashSet<String>h\_set2 = new HashSet<String>();

h\_set2.add("Red");

h\_set2.add("Pink");

h\_set2.add("Blue");

h\_set2.add("Orange");

HashSet<String>result\_set = new HashSet<String>();

for (String element : h\_set){

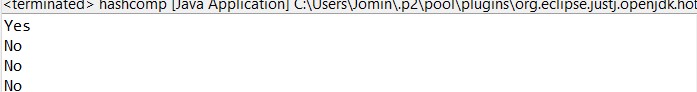
System.*out*.println(h\_set2.contains(element) ? "Yes" : "No");

}

}

}

**Output:**

****

**Program 17**

24/06/2021

**Aim:** Program to demonstrate the working of Map interface

by adding, changing and removing elements.

**Algorithm:**

Step 1: Start

Step 2: Define a class mapcoll

Step 2: Define a Map set mp.

Step 3: Define objects to Map set mp using put().

Step 4: Remove the elements using remove().

Step 5: Print the Outputs.

Step 6: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class mapcoll {

public static void main(String args[])

{

Map<Integer, String> mp = new HashMap<>();

//Inserting elements..

System.*out*.println("Enter the limit:");

Scanner inp = new Scanner(System.*in*);

int n= inp.nextInt();

System.*out*.println("Enter the Roll number and Name");

while(n!=0) {

int e= inp.nextInt();

String s= inp.next();

mp.put(e, s);

n--;

}

System.*out*.println("Initial Map:"+mp);

System.*out*.println("enter the num and name to update:");

Scanner in = new Scanner(System.*in*);

int e= in.nextInt();

String s= in.next();

mp.put(e, s);

System.*out*.println("Updated Map:"+mp);

//Removing..

System.*out*.println("Enter the Roll number to be removed:");

int r=inp.nextInt();

mp.remove(r);

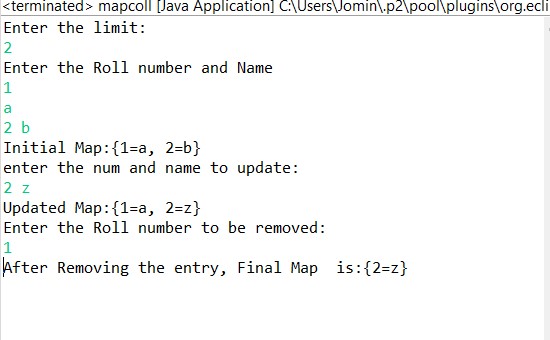
// Final Map..

System.*out*.println("After Removing the entry, Final Map is:"+mp);

}

}

**Output:**

****

**Program 18**

24/06/2021

**Aim:** Program to Convert Hash Map to Tree Map.

**Algorithm:**

Step 1: Start

Step 2: Create a class hashtotree

Step 3: Define a map.

Step 4: Define objects for map using put().

Step 5: Define a tree set treeMap

Step 6: Convert map set into tree using treeMap.putAll().

Step 7: Print the Outputs.

Step 8: Stop

**Source Code:**

package packoops;

import java.util.\*;

public class hashtotree {

public static void main(String args[]) {

Map<String, String> map = new HashMap<>();

System.*out*.println("Enter the limit:");

Scanner inp = new Scanner(System.*in*);

int n= inp.nextInt();

System.*out*.println("Enter the Roll number and Name");

while(n!=0) {

String e= inp.next();

String s= inp.next();

map.put(e, s);

n--;

}

System.*out*.println("HashMap:"+map);

Map<String, String> treeMap = new TreeMap<>();

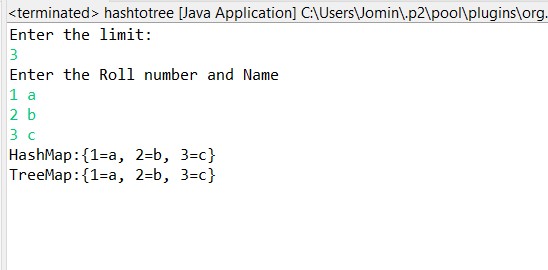
treeMap.putAll(map);

System.*out*.println("TreeMap:"+treeMap);

}

}

**Output:**

****