Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Fastest car is : BMW
```

Test Case - 2

User Output

```
Fastest car is : Maruthi
```

S.No: 17	Exp. Name: ***Write the code to create an exception***	**Date: 2023-11-06**

Aim:

Write a Java program to create an exception.

Source Code:

q221/Exception1.java

```
package q221;
public class Exception1
{
        public static void main(String args[])
        {
                int d = 0;
                try
                {
                        int a = 42/d;
                }
                catch(ArithmeticException e){
                        System.out.println("Exception caught : divide by zero occurred");
                    }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Exception caught : divide by zero occurred
```

S.No: 18	Exp. Name: ***Write the code for handling the exception***	**Date: 2023-11-06**

Aim:

Write a Java code for handling the exception.

Source Code:

q222/handleError.java

```
package q222;
import java.util.Random;
public class handleError {
	public static void main(String args[]) {
		int a = 0, b = 0, c = 0;
		Random r = new Random(100);
	for(int i=0;i<32;i++)
	{
		try
		{
			b=r.nextInt();
			c=r.nextInt();
			a=12345/(b/c);
		}
		catch(ArithmeticException e)
		{
			System.out.println("Division by zero.");
			a=0;
		}
		System.out.println("a: "+a);
	}
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
a: 12345
Division by zero.
a: 0
a: -1028
Division by zero.
a: 0
a: 12345
a: -12345
Division by zero.
a: 0
a: 3086
a: 12345
a: -12345
a: 12345
```

```
Division by zero.
a: 0
a: -12345
a: 12345
a: 342
a: 12345
a: -12345
a: 12345
a: -12345
Division by zero.
a: 0
a: -4115
Division by zero.
a: 0
a: -4115
a: 6172
a: 6172
Division by zero.
a: 0
Division by zero.
a: 0
Division by zero.
a: 0
a: 12345
a: -280
a: -12345
Division by zero.
a: 0
```

S.No: 19	Exp. Name: ***Write the code to create an exception using the predefined exception***	**Date: 2023-11-06**

Aim:

Write a Java code to create an exception using the predefined exception

Source Code:

```
q223/exception2.java
```

```
package q223;
public class exception2
{
        public static void main(String args[])
        {
                int d,a;
                try
                {
                        d=0;
                        a=42/d;
                }
                catch(ArithmeticException e)
                {
                        System.out.println("Exception raised -Division by zero.");
                        }
                        System.out.println("After catch statement.");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Exception raised -Division by zero.
After catch statement.
```

S.No: 20	Exp. Name: ***Write the code for creating your own exception***	**Date: 2023-11-06**

Aim:

Write a Java code for creating your own exception

Source Code:

q224/demo.java

```
package q224;
class MyException extends Exception
{
        private int ex;
        MyException(int a)
        {
                ex=a;
        }
        public String toString()
        {
                return "MyException["+ex+"] is less than zero";
        }
}
public class demo
{
        static void sum(int a,int b)throws MyException
        {
                if(a<0)
                throw new MyException(a);
                else
                System.out.println(a+b);
        }
        public static void main(String args[])
        {
                try{
                        sum(-10,10);
                }
                catch(MyException e)
                {
                        System.out.println(e);
                }
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
MyException[-10] is less than zero
```

S.No: 21	Exp. Name: ***program that takes inputs 5 numbers, each between 10 and 100***	**Date: 2023-12-14**

Aim:

Write java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it's not a duplicate of any number already read. Display the complete set of unique values input after the user enters new values

Source Code:

Duplicate.java

```
import java.util.Scanner;
class Duplicate{
        static boolean isDuplicate(int ele,int arr[]){
                for(int i=0;i<5;i++){
                        if(ele == arr[i]){
                                return true;
                        }
                }
                return false;
        }
        public static void main(String[] args){
                Scanner inp = new Scanner(System.in);
                int num[]=new int[5];
                System.out.println("Enter 5 unique values between 10 & 100 ");
                int c=0;
                while(c<5){
                        int element = inp.nextInt();
                        if(element>10 && element<100){
                                if(isDuplicate(element,num) == true){
                                        System.out.println("Duplicate value found, retry");
                                }else{
                                        num[c]=element;
                                        c++;
                                }
                                }else{
                                System.out.println("Entered value must be in between 10 &
100");
                                }
                        }
                System.out.print("The five unique values are :");
                for(int i=0;i<5;i++){
                        System.out.print(num[i]+" ");
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
Enter 5 unique values between 10 & 100
25
15
30
0
Entered value must be in between 10 & 100
34
89
The five unique values are :25 15 30 34 89
```

Test Case - 2

User Output

```
Enter 5 unique values between 10 & 100
48
92
34
92
Duplicate value found, retry
39
23
The five unique values are :48 92 34 39 23
```

S.No: 22	Exp. Name: ***A program to illustrate threads***	**Date: 2023-12-14**

Aim:

Write Java program(s) on creating multiple threads, assigning priority to threads, synchronizing threads, suspend and resume threads

Source Code:

`TestThread.java`

```
class RunnableDemo implements Runnable {
        public Thread t;
        public String threadName;
        boolean suspended=false;
        RunnableDemo(String name) {
                threadName=name;
                        System.out.println("Creating "+threadName);
                        }
        public void run() {
                        System.out.println("Running "+threadName);
                try{
                                for(int i=10;i>0;i--) {
                                        System.out.println("Thread: "+threadName+", "+i);
                                        Thread.sleep(100);
                                synchronized(this) {
                                        while(suspended) {
                                                wait();
                                        }
                                }
                        }
                }
                catch(InterruptedException e) {
                        System.out.println("Thread "+threadName+" interrupted.");
                                }
                        System.out.println("Thread "+threadName+" exiting.");
                        }
                        public void start() {
                                System.out.println("Starting "+threadName);
                                if(t==null) {
                                        t=new Thread(this,threadName);
                                        t.start();
                                        }
                                }
                                        void suspend() {
                                                suspended=true;
                                                }
                                                        synchronized void resume() {
                                                                suspended=false;
                                                                notify();
                                                                }
                        }
        public class TestThread{
                public static void main(String args[]) {
                                RunnableDemo R1=new RunnableDemo("Thread-1");
                                R1.start();
                        RunnableDemo R2=new RunnableDemo("Thread-2");
R2.start();
                        try{
                                Thread.sleep(100);
                                R1.suspend();
                                System.out.println("Suspending First Thread");
                                Thread.sleep(100);
                                R1.resume();
                                System.out.println("Resuming First Thread");
                                System.out.println("Suspending thread Two");
                        }
                        catch(InterruptedException e) {
                                System.out.println("Caught: "+e);
                        }
                                try{
                                        System.out.println("Waiting for threads to 
finish.");
                                        R1.t.join();
                                        R2.t.join();
                        }
                                catch(InterruptedException e){
                                        System.out.println(e);
                                        }
                        System.out.println("Main thread exiting.");
                }
        }
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Running Thread-2
Thread: Thread-2, 10
Thread: Thread-1, 10
Suspending First Thread
Thread: Thread-2, 9
Thread: Thread-2, 8
Resuming First Thread
Suspending thread Two
Thread: Thread-1, 9
Thread: Thread-1, 8
Resuming thread Two
Waiting for threads to finish.
Thread: Thread-2, 7
Thread: Thread-1, 7
Thread: Thread-2, 6
Thread: Thread-1, 6
Thread: Thread-2, 5
Thread: Thread-1, 5
Thread: Thread-2, 4
Thread: Thread-1, 4
Thread: Thread-2, 3
Thread: Thread-1, 3
Thread: Thread-2, 2
Thread: Thread-1, 2
Thread: Thread-2, 1
```

```
Thread: Thread-1, 1
Thread Thread-2 exiting.
Thread Thread-1 exiting.
Main thread exiting.
```

S.No: 23	Exp. Name: ***Write the code to print a file into n parts***	**Date: 2023-12-14**

Aim:

Write a Java code to print a file into **n** parts

Source Code:

q226/split1.java

```
package q226;
import java.io.*;
import java.util.*;
public class split1{
        public static void main(String args[]){
try {
                        String inputfile="test.txt";
                        double nol=10.0;
                        File file=new File(inputfile);
                        Scanner input=new Scanner(file);
                        int count=2;
                        while(input.hasNextLine()) {
                                input.nextLine();
                                count++;
                        }
                        System.out.println("Lines in the file: "+count);
                        double temp=(count/nol);
                        int temp1=(int)temp;
                        int nof=0;
                        if(temp1==temp)
                                nof=temp1;
                        else
                                nof=temp1+1;
                        System.out.println("No. of files to be generated :"+nof);
                        BufferedReader br=new BufferedReader(new FileReader(inputfile));
                        String strLine;
                        for(int j=1;j<-nof;j++) {
                                FileWriter fw=new FileWriter("File"+j+".txt");
                                for(int i=1;i<nol;i++) {
                                        strLine=br.readLine();
                                        if(strLine!=null) {
                                                strLine=strLine+"\r\n";
                                                fw.write(strLine);
                                        }
                                }
                                        fw.close();
                                        }
                        br.close();
                }
                catch(Exception e) {
                        System.out.println("Error: "+e.getMessage());
                }
        }
}
```

test.txt

```
Insert text here : 1614065200486
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Lines in the file: 3
No. of files to be generated :1
```

S.No: 24	Exp. Name: ***program to create a super class called Figure that it returns the area of a rectangle and triangle***	**Date: 2023-12-07**

Aim:

Write a java program to create a super class called Figure that receives the dimensions of two dimensional objects. It also defines a method called area that computes the area of an object. The program derives two sub-classes from Figure. The first is Rectangle and second is Triangle. Each of the sub classes override area() so that it returns the area of a rectangle and triangle respectively

Source Code:

`AbstractAreas.java`

```
import java.util.*;
abstract class Figure {
    double dim1,dim2,dim3,dim4;
        Figure(double a,double b) {
                dim1=a;
                dim2=b;
                dim3=a;
                dim4=b;
        }
        abstract void area();
}

class Rectangle extends Figure {
        Rectangle(double a,double b) {
                super(a,b);
        }
        void area() {
                double Area=dim1*dim2;
                System.out.println("Rectangle:");
                System.out.println("Area is "+Area);
        }
}

class Triangle extends Figure {
        Triangle(double a,double b) {
                super(a,b);
        }
        void area() {
                double Area=(dim3*dim4)/2;
                System.out.println("Triangle:");
                System.out.println("Area is "+Area);
        }
}

class AbstractAreas {
        public static void main(String args[]) {
                System.out.println("Enter lenght and breadth of Rectangle :");
                Scanner
                        input=new Scanner(System.in);
                double dim1=input.nextDouble();
                double dim2=input.nextDouble();
                System.out.println("Enter height and side of Triangle :");
                Scanner input1=new Scanner(System.in);
                double dim3=input1.nextDouble();
                double dim4=input1.nextDouble();
                Rectangle r=new Rectangle(dim1,dim2);
                Triangle t=new Triangle(dim3,dim4);
                Figure figuref;
                figuref=r;
                figuref.area();
                figuref=t;
                figuref.area();
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter lenght and breadth of Rectangle :
12
14
Enter height and side of Triangle :
7
5
Rectangle:
Area is 168.0
Triangle:
Area is 17.5
```

Test Case - 2

User Output

```
Enter lenght and breadth of Rectangle :
4
8
Enter height and side of Triangle :
5
3
Rectangle:
Area is 32.0
Triangle:
Area is 7.5
```

S.No: 25	Exp. Name: ***Write a Java program demonstrating the usage of Threads***	**Date: 2023-12-14**

Aim:

Write a Java program that uses three threads to perform the below actions:

1. First thread should print "Good morning" for every 1 second for 2 times
2. Second thread should print "Hello" for every 1 seconds for 2 times
3. Third thread should print "Welcome" for every 3 seconds for 1 times

Write appropriate **constructor** in the `Printer` class which implements `Runnable` interface to take three arguments : **message**, **delay** and `count` of types **String**, **int** and **int** respectively.

Write code in the `Printer.run()` method to print the **message** with appropriate **delay** and for number of times mentioned in **count**.

Write a class called `ThreadDemo` with the `main()` method which instantiates and executes three instances of the above mentioned `Printer` class as threads to produce the desired output.

[**Note:** If you want to sleep for **2** seconds you should call `Thread.sleep(2000);` as the `Thread.sleep(...)` method takes milliseconds as argument.]

Note: Please don't change the package name.

Source Code:

```
q11349/ThreadDemo.java
```

```
package q11349;
public class ThreadDemo {
        public static void main(String[] args) throws Exception {
                Thread t1 = new Thread(new Printer("Good morning", 1, 2));
                Thread t2 = new Thread(new Printer("Hello", 1, 2));
                Thread t3 = new Thread(new Printer("Welcome", 3, 1));
                t1.start();
                t2.start();
                t3.start();
                t1.join();
                t2.join();
                t3.join();
                System.out.println("All the three threads t1, t2 and t3 have completed 
execution.");
    }
}
class Printer implements Runnable {
        String message;
        int delay,count;
        Printer(String a,int b,int c)
        {
                message=a;
                delay=b;
                count=c;
        }
        public void run()
        {
                for(int i=0;i<count;i++)
                        {
                                System.out.println(message);
                                try{
                                        Thread.sleep(delay*1000);
                                }
                                catch(InterruptedException ie)
                                        {
                                                System.out.println(ie);
                                        }
                        }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Good morning
Hello
Welcome
Good morning
Hello
All the three threads t1, t2 and t3 have completed execution.
```

S.No: 26	Exp. Name: ***Program to find and replace pattern in a given file.***	**Date: 2023-12-14**

Aim:

Write a java program to find and replace patterns in a given file. Replace the string "**This is test string 20000**" with the input string.

Note: Please don't change the package name.

Source Code:

q29790/ReplaceFile.java

```
package q29790;
import java.io.*;
import java.util.*;
import java.util.*;
class ReplaceFile {
        public static void main(String args[])
        {
                try
                        {
                                File file = new File("file.txt");
                                        BufferedReader reader = new BufferedReader(new 
FileReader(file));
                                        String line , oldtext=new String();
                                        while((line = reader.readLine()) != null)
                                        {
                                                if(oldtext==null)
                                                oldtext = line + "\r\n";
                                                else
                                                oldtext += line + "\r\n";
                                                }
                                        reader.close();
                                        System.out.print("Previous string: "+oldtext);
                                        // replace a word in a file
                                        //String newtext = oldtext.replaceAll("drink", 
"Love");
                                        //To replace a line in a file
                                        String newtext = oldtext.replaceAll("This is test 
string 20000", "New string");
                                        System.out.print("New String: "+newtext);
                                        }
                                                catch (IOException ioe)
                                                {
                                                        ioe.printStackTrace();
                                                        }
                        }
}
```

file.txt

```
This is test string 20000. The test string is replaced with your input string, check the
string you entered is now visible here.
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
New string
Previous string: This is test string 20000. The test string is replaced with your input
string, check the string you entered is now visible here.
New String: New string. The test string is replaced with your input string, check the
string you entered is now visible here.
```

S.No: 27	Exp. Name: ***A java program to demonstrate that the catch block for type Exception A catches the exception of type Exception B and Exception C.***	**Date: 2023-12-07**

Aim:

Use inheritance to create an exception superclass called Exception A and exception subclasses Exception B and Exception C, where Exception B inherits from Exception A and Exception C inherits from Exception B. Write a java program to demonstrate that the catch block for type Exception A catches the exception of type Exception B and Exception C.

Note: Please don't change the package name.

Source Code:

`q29793/TestException.java`

```
package q29793;
import java.lang.*;
@SuppressWarnings("serial")
class ExceptionA extends Exception {
        String message;
        public ExceptionA(String message) {
                this.message = message;
        }
}
@SuppressWarnings("serial")
class ExceptionB extends ExceptionA {
//Write constructor of class ExceptionB with super()
ExceptionB(String message){
        super(message);
}
}
@SuppressWarnings("serial")
class ExceptionC extends ExceptionB {
//Write constructor of class ExceptionC with super()
ExceptionC(String message){
        super(message);
        }
}
@SuppressWarnings("serial")
public class TestException {
        public static void main(String[] args) {
                try {
                        getExceptionB();
                }
                catch(ExceptionA ea) {
                        System.out.println("Got exception from Exception B");
                }
                try {
                        getExceptionC();
                }
                catch(ExceptionA ea) {
                        System.out.println("Got exception from Exception C");
                }
        }
        public static void getExceptionB() throws ExceptionB {
                throw new ExceptionB("Exception B");
        }
        public static void getExceptionC() throws ExceptionC {
                throw new ExceptionC("Exception C");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Got exception from Exception B
Got exception from Exception C
```

S.No: 28	Exp. Name: ***Stack Implementation***	**Date: 2023-12-14**

Aim:

Create an interface for stack with push and pop operations. Implement the stack in two ways fixed-size stack and Dynamic stack (stack size is increased when the stack is full).

Note: Please don't change the package name.

Source Code:

```
q29794/StaticAndDynamicStack.java
```

```
package q29794;
interface IntStack{
        void push(int item);
        int pop();
}
class FixedStack implements IntStack{
                        private int stck[];
        private int tos;
        FixedStack(int size)
        {
        stck = new int[size];
                tos = -1;
                }
        public void push(int item)
        {
        if(tos == stck.length-1)
                System.out.println("Stack is full and increased");
        else
                stck[++tos]=item;
}
public int pop()
        {
        if (tos<0)
                        {
                        System.out.println("Stack underflow");
                        return 0;
                        }
                                else
                                return stck[tos--];
                }
        }
        class StaticAndDynamicStack{
        public static void main(String args[])
        {
                        FixedStack mystack = new FixedStack(0);
                        FixedStack mystack1 = new FixedStack(5);
                        FixedStack mystack2 = new FixedStack(10);
                        for(int i=0;i<1;i++)
                                mystack.push(i);
                        for(int i=0;i<5;i++)
                                mystack1.push(i);
                        for(int i=0;i<10;i++)
                                mystack2.push(i);
                        System.out.println("Stack in mystack1:");
                        for(int i=0;i<5;i++)
                        System.out.println(mystack1.pop());
                System.out.print("Stack in mystack2 :\n");
                for(int i=0;i<4;i++)
                        System.out.println(mystack2.pop());
                mystack2.pop();
                for(int i=1;i<6;i++)
                        System.out.println(mystack2.pop());
                System.out.println(mystack.pop());
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Stack is full and increased
Stack in mystack1:
4
3
2
1
0
Stack in mystack2 :
9
8
7
6
4
3
2
1
0
Stack underflow
0
```

S.No: 29	Exp. Name: ***Create multiple threads to access the contents of a stack***	**Date: 2023-12-14**

Aim:

Create multiple threads to access the contents of a stack. Synchronize thread to prevent simultaneous access to push and pop operations.

Note: Please don't change the package name.

Source Code:

q29795/StackThreads.java

```
package q29795;
import java.util.*;
class NewThread implements Runnable{
        Thread t;
        int n;
        Stack<Integer> STACK=new Stack<Integer>();
NewThread(int size){
        n=size;
        t=new Thread(this);
        t.start();
}
synchronized public void run(){
        STACK.push(n);
        System.out.println(STACK.pop());
 }
}
class StackThreads{
        public static void main(String args[]){
                System.out.println("Enter the size of the stack");
                Scanner sc=new Scanner(System.in);
                        int k=sc.nextInt();
                for(int i=1;i<=k;i++){
                        NewThread ob=new NewThread(i);
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the size of the stack
4
1
2
3
4
```

Test Case - 2

User Output

```
Enter the size of the stack
9
1
2
3
4
5
6
7
8
9
```

S.No: 30	Exp. Name: ***Write java program(s) that use collection framework classes.(TreeMap class)***	**Date: 2023-12-07**

Aim:

Write a java program(s) that use collection framework classes.(TreeMap class)

Source Code:

Treemap.java

```
import java.util.*;
public class Treemap{
        public static void main(String[] args) {
                Scanner inp = new Scanner(System.in);
        TreeMap<Integer,String> treeMap = new TreeMap<Integer,String>();
        System.out.print("No.Of Mapping Elements in TreeMap:");
        int num = inp.nextInt();
        for(int i=0;i<num;i++){
                System.out.print("Integer:");
                int key = inp.nextInt();
                inp.nextLine();
                System.out.print("String:");
                String value = inp.nextLine();
                treeMap.put(key,value);
        }
        for(Map.Entry m: treeMap.entrySet()) {
                System.out.println(m.getKey()+"->"+m.getValue());
        }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
No.Of Mapping Elements in TreeMap:
2
Integer:
1
String:
HELLO
Integer:
2
String:
WORLD
1->HELLO
2->WORLD
```

Test Case - 2

User Output

```
No.Of Mapping Elements in TreeMap:
3
Integer:
25
String:
UNIVERSITY
Integer:
26
String:
KNOWLEDGE
Integer:
27
String:
TECHNOLOGIES
25->UNIVERSITY
26->KNOWLEDGE
27->TECHNOLOGIES
```

S.No: 31	Exp. Name: ***Write java program(s) that use collection framework classes.(TreeSet class)***	**Date: 2023-12-07**

Aim:

Write java program(s) that use collection framework classes.(TreeSet class)

Source Code:

TreeSetclass.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.TreeSet;
public class TreeSetclass{
        public static void main(String[] args) throws Exception {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.print("No.Of Elements in TreeSet:");
                int size = Integer.parseInt(br.readLine());
                TreeSet<String>strings = new TreeSet<>();
                for(int i=0;i<size;++i){
                        System.out.print("String:");
                        strings.add(br.readLine());
                }
                System.out.println("TreeSet Elements by Iterating:");
                for(String s1 : strings)
System.out.println(s1);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
No.Of Elements in TreeSet:
3
String:
Never
String:
Give
String:
Up
TreeSet Elements by Iterating:
Give
Never
Up
```

Test Case - 2

User Output

```
No.Of Elements in TreeSet:
```

```
2
String:
Hello
String:
There
TreeSet Elements by Iterating:
Hello
There
```

S.No: 32	Exp. Name: ***Write java program(s) that use collection framework classes.(LinkedHashMap class)***	**Date: 2023-12-07**

Aim:

Write a java program(s) that use collection framework classes.(LinkedHashMap class)

Source Code:

LinkedHashMapclass.java

```
import java.util.*;
public class LinkedHashMapclass{
        public static void main(String[] args){
Scanner inp = new Scanner(System.in);
                LinkedHashMap<String,String>linkedHashMap = new LinkedHashMap<String,String>
();
                System.out.print("No.Of Mapping Elements in LinkedHashMap:");
                int num = inp.nextInt();
                inp.nextLine();
                for(int i=0;i<num;i++){
System.out.print("String:");
                        String Key = inp.nextLine();
                        System.out.print("Corresponding String:");
                        String value = inp.nextLine();
                        linkedHashMap.put(Key,value);
                }
System.out.println("LinkedHashMap entries : ");
                for(Map.Entry m: linkedHashMap.entrySet()){
                        System.out.println(m.getKey()+"="+m.getValue());
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
No.Of Mapping Elements in LinkedHashMap:
3
String:
ONE
Corresponding String:
hi
String:
TWO
Corresponding String:
hello
String:
THREE
Corresponding String:
```

```
everyone
LinkedHashMap entries :
ONE=hi
TWO=hello
THREE=everyone
```

Test Case - 2

User Output

```
No.Of Mapping Elements in LinkedHashMap:
4
String:
1x1
Corresponding String:
1
String:
1x2
Corresponding String:
2
String:
1x3
Corresponding String:
3
String:
1x4
Corresponding String:
4
LinkedHashMap entries :
1x1=1
1x2=2
1x3=3
1x4=4
```

S.No: 33	Exp. Name: ***Write java program(s) that use collection framework classes.(HashMap class)***	**Date: 2023-12-07**

Aim:

Write a java program(s) that use collection framework classes.(HashMap class)

Source Code:

HashMapclass.java

```
import java.util.*;
public class HashMapclass{
        public static void main(String[] args){
Scanner inp = new Scanner(System.in);
                HashMap<String, Integer> hashMap = new HashMap<String,Integer>();
                System.out.print("No.Of Mapping Elements in HashMap:");
                int num = inp.nextInt();
                for(int i=0;i<num;i++){
inp.nextLine();
                        System.out.print("String:");
                        String key = inp.nextLine();
                        System.out.print("Integer:");
                        int value = inp.nextInt();
                hashMap.put(key, value);
                }
for(Map.Entry m : hashMap. entrySet()){
        System.out.println("Key = "+m.getKey()+", Value = "+m.getValue());
}
                System.out.println(hashMap);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
No.Of Mapping Elements in HashMap:
3
String:
hi
Integer:
1
String:
hello
Integer:
2
String:
world
Integer:
3
Key = hi, Value = 1
Key = world, Value = 3
Key = hello, Value = 2
{hi=1, world=3, hello=2}
```

Test Case - 2

User Output

```
No.Of Mapping Elements in HashMap:
3
String:
Students
Integer:
200
String:
Teachers
Integer:
5
String:
Principal
Integer:
1
Key = Teachers, Value = 5
Key = Students, Value = 200
Key = Principal, Value = 1
{Teachers=5, Students=200, Principal=1}
```

S.No: 34	Exp. Name: ***Write java program(s) that use collection framework classes.(LinkedList class)***	**Date: 2023-12-07**

Aim:

Write a java program(s) that use collection framework classes.(LinkedList class)

Source Code:

Linkedlist.java

```
import java.util.*;
import java.io.*;
public class Linkedlist{
        public static void main(String[] args){
try{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("No.Of Strings in LinkedList:");
        int size=Integer.parseInt(br.readLine());
        LinkedList<String>stringList=new LinkedList<>();
        for(int i=1;i<=size;++i){
                System.out.println("Enter the String:");
                stringList.addLast(br.readLine());
        }
        System.out.println("LinkedList:"+stringList);
        System.out.println("The List is as follows:");
        for(String word:stringList)
                System.out.println(word);
}
                catch(IOException e){
e.printStackTrace();
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
No.Of Strings in LinkedList:
3
Enter the String:
Hi
Enter the String:
Hello
Enter the String:
World
LinkedList:[Hi, Hello, World]
The List is as follows:
Hi
Hello
World
```

Test Case - 2

User Output

```
No.Of Strings in LinkedList:
2
Enter the String:
Human
Enter the String:
Being
LinkedList:[Human, Being]
The List is as follows:
Human
Being
```

S.No: 35	Exp. Name: ***Write java program(s) that use collection framework classes.(ArrayList class)***	**Date: 2023-12-07**

Aim:

Write a java program(s) that use collection framework classes.(ArrayList class)

Source Code:

ArraylistExample.java

```
import java.io.*;
import java.util.*;
class ArraylistExample{
        public static void main(String[] args){
int n;
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter ArrayList length: ");
                n=sc.nextInt();
                ArrayList<Integer>arrli=new ArrayList<Integer>(n);
                System.out.println("ArrayList printing by using Iterator: ");
                for(int i=1;i<=n;i++){
                        arrli.add(i);
                }
                for(int i: arrli)
                        {
                                System.out.println(i);
                        }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter ArrayList length: 
5
ArrayList printing by using Iterator: 
1
2
3
4
5
```

Test Case - 2

User Output

```
Enter ArrayList length: 
3
ArrayList printing by using Iterator: 
1
2
```

```
3
```

S.No: 36	Exp. Name: ***Write java program(s) that use collection framework classes.(HashTable class)***	**Date: 2023-12-07**

Aim:

Write a java program(s) that use collection framework classes.(HashTable class)

Source Code:

HashTableclass.java

```
import java.util.*;
public class HashTableclass {
        public static void main(String[] args){
                Scanner inp = new Scanner(System.in);
                Hashtable<Integer,String> hashTable = new Hashtable<Integer,String>();
                System.out.print("No.Of Mapping Elements in HashTable:");
int num = inp.nextInt();
                for(int i=0;i<num;++i) {
                        System.out.print("Rank:");
                        int key = inp.nextInt();
                        inp.nextLine();
                        System.out.print("Name:");
                        String value = inp.nextLine();
                        hashTable.put(key,value);
                }
                for(Map.Entry<Integer,String> m : hashTable.entrySet()) {
                        System.out.println("Rank : "+m.getKey()+"\t\t Name :
"+m.getValue());
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
No.Of Mapping Elements in HashTable:
3
Rank:
4
Name:
Robert
Rank:
5
Name:
John
Rank:
6
Name:
Jennifer
Rank : 6                 Name : Jennifer
Rank : 5            Name : John
Rank : 4            Name : Robert
```

Test Case - 2

User Output

```
No.Of Mapping Elements in HashTable:
3
Rank:
1
Name:
Jon
Rank:
2
Name:
Robert
Rank:
3
Name:
Jennifer
Rank : 3            Name : Jennifer
Rank : 2            Name : Robert
Rank : 1            Name : Jon
```