S.No: 2	Exp. Name: ***Sample program on java to demonstrate Control structures***	**Date:2023-09-16**

Aim:

write a java program that uses if-else control statement and print the result

Source Code:

Control.java

```
import java.util.Scanner;
public class Control
{
public static void main(String args[])
{
Scanner s = new Scanner(System.in);
System.out.print("Enter first num : ");
int x = s.nextInt();
System.out.print("Enter second num : ");
int y = s.nextInt();
if(x+y>20)
{
System.out.println("x + y is greater than 20");
}
else
{
System.out.println("x + y is less than 20");
}
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter first num :  13
Enter second num :  5
x + y is less than 20
```

Test Case - 2

User Output

```
Enter first num :  24
Enter second num :  10
x + y is greater than 20
```

S.No: 4	Exp. Name: ***Sample program to demonstrate destructor***	Date:2023-09-16

Aim:

Write a program to demonstrate destructor class

Source Code:

DestructorExample.java

```
public class DestructorExample
{
   public static void main(String args[])
   {
      DestructorExample m=new DestructorExample();
      m.finalize();
      m=null;
      System.gc();
      System.out.println("Inside the main() method");
   }
   protected void finalize(){
      System.out.println("Object is destroyed by the Garbage Collector");
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Object is destroyed by the Garbage Collector
Inside the main() method
Object is destroyed by the Garbage Collector
```

S.No: 5	Exp. Name: ***A program to print Half pyramid pattern***	Date:2023-09-16

Aim:

Write a Java program to print Half Pyramid pattern.

Source Code:

HalfPyramid.java

```
import java.util.*;
class HalfPyramid
{
   public static void main(String args[])
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter no of rows : ");
      int rows=sc.nextInt();
      for(int i=0;i<rows;i++)
      {
         for(int j=0;j<=i;j++)
         {
            System.out.print("* ");
         }
         System.out.println();
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows :  5
*
* *
* * *
* * * *
* * * * *
```

Test Case - 2

User Output

```
Enter no of rows :  3
*
* *
* * *
```

Test Case - 3

User Output

```
Enter no of rows :  10
*
* *
```

* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *

S.No: 6	Exp. Name: ***A program to print Inverted Half pyramin pattern***	Date:2023-09-16

Aim:

Write a Program to Print Inverted Half Pyramid Pattern

Source Code:

HalfPyramidRev.java

```
import java.util.Scanner;
 class HalfPyramidRev
 {
   public static void main(String args[])
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter no of rows : ");
      int rows=sc.nextInt();
      for(int i=rows;i>0;i--)
      {
         for(int j=i;j>0;j--)
         {
            System.out.print("* ");
         }
         System.out.println();
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows :  5
* * * * *
* * * *
* * *
* *
*
```

Test Case - 2

User Output

```
Enter no of rows :  3
* * *
* *
*
```

S.No: 7	Exp. Name: ***A program to print Hollow Inverted Half Pyramid Pattern***	Date:2023-09-16

Aim:

Write a Program to Print Hollow Inverted half Pyramid Pattern

Source Code:

HollowHalfPyramidRev.java

```
import java.util.*;
class HollowHalfPyramidRev
{
   public static void main(String args[])
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter no of rows : ");
      int rows=sc.nextInt();
      for(int i=rows;i>0;i--)
      {
         for(int j=1;j<=i;j++)
         {
            if(j==1||i==rows||j==i)
            {
               System.out.print("* ");
            }
            else{
               System.out.print("  ");
            }
         }
         System.out.println("");
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows :  5
* * * * *
*     *
*   *
* *
*
```

Test Case - 2

User Output

```
Enter no of rows :  3
* * *
* *
*
```

S.No: 8	Exp. Name: ***A program to print Pyramid Pattern***	Date:2023-09-16

Aim:

Write a Program to Print Pyramid Pattern

Source Code:

Pyramid.java

```java
import java.util.*;
class Pyramid{
   public static void main(String args[]){
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter no of rows : ");
      int rows=sc.nextInt();
      for(int i=1;i<=rows;i++){
         for(int j=rows-i;j>0;j--){
            System.out.print(" ");
         }
         for(int k=i;k>0;k--){
            System.out.print("* ");
         }
         System.out.println();
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows :  5
    * 
   * * 
  * * * 
 * * * * 
* * * * * 
```

Test Case - 2

User Output

```
Enter no of rows :  6
     * 
    * * 
   * * * 
  * * * * 
 * * * * * 
* * * * * * 
```

S.No: 9	Exp. Name: ***A program to print Inverted Pyramid Pattern***	**Date:2023-09-16**

Aim:

Write a Program to Print inverted Pyramid Pattern

Source Code:

PyramidRev.java

```
import java.util.*;
class PyramidRev{
   public static void main(String args[]){
      Scanner sc=new Scanner(System.in);
      int j,k;
      System.out.print("Enter no of rows : ");
      int rows=sc.nextInt();
      for(int i=rows;i>=1;i--){
         for(j=rows-i;j>=1;j--){
            System.out.print(" ");
         }
         for(k=i;k>=1;k--){
            System.out.print("* ");
         }
         System.out.println();
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows :  5
* * * * * 
 * * * * 
  * * * 
   * * 
    * 
```

Test Case - 2

User Output

```
Enter no of rows :  6
* * * * * * 
 * * * * * 
  * * * * 
   * * * 
    * * 
     * 
```

S.No: 10	Exp. Name: ***A program to print Hollow Pyramid Pattern***	Date:2023-09-16

Aim:

Write a Program to print the Hollow pyramid pattern

Source Code:

PyramidGap.java

```
import java.util.*;
class PyramidGap{
   public static void main(String args[]){
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter no of rows : ");
      int n=sc.nextInt();
      for(int i=1;i<=n;i++){
         for(int j=1;j<=n-i;j++)
            System.out.print(" ");
            for(int j=1;j<=i;j++){
               if(j==1||j==i||i==n){
               System.out.print("* ");
            }
            else{
               System.out.print("  ");
            }
         }
         System.out.println();
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows :  5
    *
   * *
  *   *
 *     *
* * * * *
```

Test Case - 2

User Output

```
Enter no of rows :  6
     *
    * *
   *   *
  *     *
 *       *
* * * * * *
```

Aim:

Write Java program on use of Inheritance.

Create a class Vehicle

- contains the data members **color** of String type and **speed** and **size** of integer data type.
- write a method **setVehicleAttributes()** to initialize the data members

Create another class Car which is derived from the class Vehicle

- contains the data members **cc** and **gears** of **integer** data type
- write a method **setCarAttributes()** to initialize the data members
- write a method **displayCarAttributes()** which will display all the attributes.

Write another class InheritanceDemo with **main()** it receives five arguments **color**, **speed**, **size**, **cc** and **gears**.

Source Code:

InheritanceDemo.java

```
import java.util.*;
class Vehicle
{
  String color;
  int speed;
  int size;
  void setVehicleAttributes(String c,String sp,String s)
  {
   color=c;
   speed=Integer.parseInt(sp);
   size=Integer.parseInt(s);
  }
}
class Car extends Vehicle
{
   int cc;
   int gears;
   void setCarAttributes(String c,String sp,String s,String x,String y)
   {
      setVehicleAttributes(c,sp,s);
      cc=Integer.parseInt(x);
      gears=Integer.parseInt(y);
      displayCarAttributes();
   }
   void displayCarAttributes(){
      System.out.println("Color of Car : "+color);
      System.out.println("Speed of Car : "+speed);
      System.out.println("Size of Car : "+size);
      System.out.println("CC of Car : "+cc);
      System.out.println("No of gears of Car : "+gears);
   }
}
public class InheritanceDemo
{
    public static void main(String args[])
    {
       Car z=new Car();
       z.setCarAttributes(args[0],args[1],args[2],args[3],args[4]);
    }
 }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`Color of Car : Blue`
`Speed of Car : 100`
`Size of Car : 20`
`CC of Car : 1000`
`No of gears of Car : 5`

Test Case - 2
User Output
`Color of Car : Orange`
`Speed of Car : 120`
`Size of Car : 25`
`CC of Car : 900`
`No of gears of Car : 5`

S.No: 12	Exp. Name: ***write a java program to prevent inheritance using abstract class.***	Date:2023-10-15

Aim:

write a java program to prevent inheritance using abstract class.

- Create an abstract class `Shape`
- Create a class `Rectangle` which extends the class `Shape`
- Class Rectangle contains a method **draw** whcih prints **drawing rectangle**
- Create another class `circle1` which extends `Shape`
- Class circle1 contains a method **draw** whcih prints **drawing circle**
- Create a main class `TestAbstraction1`
- Create object for the class circle1 and called the method draw

Source Code:

TestAbstraction1.java

```
import java.util.*;
abstract class Shape{
   abstract void draw();
}
class Rectangle extends Shape{
   void draw(){
   System.out.println("drawing rectangle");
}
}
class Circle1 extends Shape{
   void draw() {
      System.out.println("drawing circle");
   }
}
class TestAbstraction1{
   public static void main(String args[]){
      Shape s=new Circle1();
      s.draw();
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`drawing circle`

S.No: 13	Exp. Name: ***program on dynamic binding***	Date:2023-10-15

Aim:

write a program on dynamic binding

Source Code:

Demo.java

```
class Human{
   public void walk(){
      System.out.println("Human walks");
   }
}
class Demo extends Human{
   public void walk(){
      System.out.println("Boy walks");
   }
   public static void main(String args[]){
      Human x=new Demo();
      Human y=new Human();
      x.walk();
      y.walk();
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Boy walks
Human walks

S.No: 14	Exp. Name: ***Sample program on method overloading***	Date:2023-10-15

Aim:

Write a program on method overloading

Source Code:

Sample.java

```java
class DisplayOverLoading{
   void disp(char c){
      System.out.println(c);
   }
   void disp(char c,int num){
      System.out.println(c+" "+num);
   }
}
class Sample{
   public static void main(String args[]){
      DisplayOverLoading obj=new DisplayOverLoading();
      obj.disp('a');
      obj.disp('a',10);
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
a
a 10
```

S.No: 15	Exp. Name: ***Sample program on method overriding***	Date:2023-10-15

Aim:

Write a program on method overriding

Source Code:

Bike.java

```
class Vehicle {
   void run(){
      System.out.println("Bike");
   }
}
class Vehicle2 extends Vehicle {
   void run(){
      System.out.println("Bike is running");
   }
}
class Bike extends Vehicle2{
   void run(){
      System.out.println("Bike is running safely");
   }
   public static void main(String args[]){
      Bike b=new Bike();
      b.run();
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`Bike is running safely`

Aim:

Write a Java program that implements an **interface**.

Create an interface called `Car` with two abstract methods `String getName()` and `int getMaxSpeed()`. Also declare one **default** method `void applyBreak()` which has the code snippet

```
System.out.println("Applying break on " + getName());
```

In the same interface include a **static** method `Car getFastestCar(Car car1, Car car2)`, which returns **car1** if the **maxSpeed** of **car1** is greater than or equal to that of **car2**, else should return **car2**.

Create a class called `BMW` which implements the interface `Car` and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).

Similarly, create a class called `Audi` which implements the interface `Car` and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).

Create a **public** class called `MainApp` with the **main()** method.
Take the input from the command line arguments. Create objects for the classes `BMW` and `Audi` then print the fastest car.

Note:
Java 8 introduced a new feature called `default` methods or `defender` methods, which allow developers to add new methods to the interfaces without breaking the existing implementation of these interface. These **default** methods can also be overridden in the implementing classes or made abstract in the extending interfaces. If they are not overridden, their implementation will be shared by all the implementing classes or sub interfaces.

Below is the syntax for declaring a `default` method in an **interface** :

```
public default void methodName() {
    System.out.println("This is a default method in interface");
}
```

Similarly, **Java 8** also introduced `static` methods inside interfaces, which act as regular static methods in classes. These allow developers group the utility functions along with the interfaces instead of defining them in a separate helper class.

Below is the syntax for declaring a `static` method in an **interface** :

```
public static void methodName() {
    System.out.println("This is a static method in interface");
}
```

Note: Please don't change the package name.

Source Code:

q11284/MainApp.java

```
package q11284;
interface Car{
   String getName();
   int getMaxSpeed();
   default void applyBreak()
   {
      System.out.println("Applying break on " + getName());
   }
   static Car getFastestCar(Car car1, Car car2){
      return (car1.getMaxSpeed() >= car2.getMaxSpeed()) ? car1 : car2;
}
}
class BMW implements Car {
      private String name;
      private int maxSpeed;
      public BMW(String name, int maxSpeed) {
               this.name = name;
               this.maxSpeed = maxSpeed;
      }
      public int getMaxSpeed() {
               return maxSpeed;
      }
      public String getName() {
               return name;
      }
}
class Audi implements Car {
      private String name;
      private int maxSpeed;
      public Audi(String name, int maxSpeed) {
               this.name = name;
               this.maxSpeed = maxSpeed;
      }
      public int getMaxSpeed() {
               return maxSpeed;
      }
      public String getName() {
               return name;
      }
}
public class MainApp {
      public static void main(String args[]) {
               String name = args[0];
               int speed = Integer.parseInt(args[1]);
               String name1 = args[2];
               int speed1 = Integer.parseInt(args[3]);
               Car car1 = new BMW(name, speed);
               Car car2 = new Audi(name1, speed1);
               System.out.println("Fastest car is : " + Car.getFastestCar(car
1, car2).getName());
      }
}
```

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:2023-11-19

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:2023-11-19

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:2023-11-19

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:2023-11-19

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`

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");
         R2.suspend();
         Thread.sleep(100);
         System.out.println("Resuming thread Two");
         R2.resume();
      }
      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:2024-01-03

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.Scanner;
public class split1 {
      public static void main(String args[]) {
               try{
                           String inputfile = "test.txt";
                           double nol = 5.0;
                           File file = new File(inputfile);
                           Scanner scanner = new Scanner(file);
                           int count = 0;
                           while (scanner.hasNextLine()) {
                                          scanner.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 FileRead
er(inputfile));
                           String strLine;
                           for (int j = 1; j <= nof; j++) {
                                          FileWriter fw= new FileWriter("Fil
e"+j+".txt");
                                          for (int i = 1; i <= nol; i++) {
                                                            strLine = br.readL
ine();
                                                            if (strLine!= nul
l) {

strLine=strLine+"\r\n";

fw.write(strLine);
                                                            }
                                          }
                                          fw.close();
                           }
```

```
                        br.close();
                }
                catch (Exception e) {
                        System.err.println("Error: " + e.getMessage());
                }
        }
}
```

test.txt

```
Insert text here : 1614065200486
hi
hello
```

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:2023-11-19

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;
   double dim2;
   double dim3;
   double 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:2024-01-04

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 e
xecution.");
    }
}
class Printer implements Runnable {
   public String name;
   public int rep;
   public int delay;
   public Printer(String name,int delay,int rep){
      this.name=name;
      this.delay=delay;
   this.rep=rep;
}
public void run(){
   for(int i=0;i<rep;i++){
      System.out.println(name);
      try{
         Thread.sleep(delay*1000);
      }catch(Exception e){
         e.printStackTrace();
      }
      }
   }
}
```

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:2024-01-03

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.*;
public class ReplaceFile {
      public static void main(String args[]) {
               try {
                           Scanner sc = new Scanner(System.in);
                           String input = sc.nextLine();
                           File file = new File("file.txt");
                           BufferedReader reader = new BufferedReader(new File
Reader(file));
                           String line = "", oldtext = "";
                           while((line = reader.readLine()) != null) {
                                          oldtext += line + "\r\n";
                           }
                           reader.close();
                           String newtext = oldtext.replaceAll("This is test s
tring 20000", input);
                           FileWriter writer = new FileWriter("file.txt");

                           writer.write(newtext);writer.close();
                           System.out.print("Previous string: "+oldtext);
                           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 i
New String: New string. The test string is replaced with your input string, check t
```

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:2023-11-19

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 {
ExceptionB(String message){
   super(message);
}


}
@SuppressWarnings("serial")
class ExceptionC extends ExceptionB {
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

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

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 Stack {
      int tos;
      int stck[];
      int size;
      Stack(int size) {
              this.size=size;
              tos=-1;
              stck=new int[this.size];
      }
      synchronized void push(int item) {
              if(tos==stck.length-1) {
                          // use length member
                          System.out.println("Stack is full");
              }
              else {
                          stck[++tos] = item;
              }
      }
      // Pop an item from the stack
      synchronized int pop() {
              if(tos < 0) {

                          System.out.println("Stack underflow");
                          return 0;
              }
              else
                          return stck[tos--];
      }
}

class PushThread extends Thread {
      Stack s;
      PushThread(Stack s) {
              this.s=s;
      }
      public void run() {
              for(int i=1;i<=s.size;i++) {
                          s.push(i);
                          try {
```

```
                                        Thread.sleep(100);
                            }
                            catch(Exception e) {
                                        System.out.println(e);
                            }
                  }
      }
}
class PopThread extends Thread {

      Stack s;
      PopThread(Stack s){
         this.s=s;
      }
      public void run() {
                  for(int i=1;i<=s.size;i++) {
                            System.out.println(s.pop());
                            try {
                                        Thread.sleep(100);
                            }
                            catch(Exception e) {
                                        System.out.println(e);
                            }
                  }
      }
}

public class StackThreads {
      public static void main(String args[]) {
                  int size;
                  Scanner sc =new Scanner(System.in);
                  System.out.println("Enter the size of the stack");
                  size=sc.nextInt();
                  Stack s = new Stack(size);//only one object
                  PushThread t1=new PushThread(s);
                  PopThread t2=new PopThread(s);
                  t1.start();
                  t2.start();
                  t2.setPriority(9);

      }
}
```

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
```

Aim:

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

Source Code:

Treemap.java

```
// Java Program to Iterate Over Enter in a TreeMap
import java.util.*;
//Importing required
//Main class
class Treemap{
   //Main driver method
   public static void main(String[] args)
   {
      //Creating a TreeMap class Object
      //Objects are of Key-value pairs (integer and // string type)
      TreeMap<Integer, String> tm= new TreeMap<Integer, String>();
      // Customly adding elements
      System.out.print("No.Of Mapping Elements in TreeMap:");
      Scanner sc=new Scanner(System.in);
      int n =sc.nextInt();
      for(int i=0;i<n;i++){
         System.out.print("Integer:");
         Scanner b = new Scanner(System.in);
         int num=b.nextInt();
         System.out.print("String:");
         Scanner a=new Scanner(System.in);
         String str = a.nextLine();
         tm.put(num,str);
      }
      // Get all entries usign the entrySet() method
      Set<Map.Entry<Integer, String> > entries=tm.entrySet();
      // Getting an iterator
      Iterator<Map.Entry<Integer,String> > iterator= entries.iterator();
      // Additional step here
      // To Initialize object holding for
      // Key-value pairs to null
      Map.Entry<Integer, String> entry = null;
      // Holds true till there is no element remaining in
      // the object using hasNExt() method
      while (iterator.hasNext()){
         // Moving onto next pairs using next() method
         entry = iterator.next();
         // Printing the key-value pairs
         // using getKet() and getValue() methods
         System.out.println(entry.getKey() + "->"  + entry.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`

Aim:

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

Source Code:

TreeSetclass.java

```
import java.util.*;
class TreeSetclass{
	public static void main(String args[]){
		  //Creating and adding elements
		TreeSet<String> al=new TreeSet<String>();
		System.out.print("No.Of Elements in TreeSet:");
		Scanner sc= new Scanner(System.in);
		int n = sc.nextInt();
		for (int i=0; i<n; i++){
				System.out.print("String:");
				Scanner b = new Scanner(System.in);
				String str= b.nextLine();
				al.add(str);
		}

		  //Traversing elements
		  System.out.println("TreeSet Elements by Iterating:");
		  Iterator<String> itr=al.iterator();
		  while(itr.hasNext()){
					System.out.println(itr.next());
		 }
	}
}
```

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
```

Aim:

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

Source Code:

LinkedHashMapclass.java

```
   // Java Program to iterate over linkedHashMap using
   // entrySet() and iterator
   import java.util.Scanner;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Set;

public class LinkedHashMapclass {

      public static void main(String[] args){

              // Create a LinkedHashMap and populate it with
              // elements
              LinkedHashMap<String,String> linkedHashMap
                          = new LinkedHashMap<String,String>();

              // adding the elements to the linkedHashMap
              System.out.print("No.Of Mapping Elements in LinkedHashMap:");
              Scanner sc= new Scanner(System.in);
              int n = sc.nextInt();
              for (int i=0; i<n; i++){
                          System.out.print("String:");
                          Scanner b = new Scanner(System.in);
                          String str1= b.nextLine();
                          System.out.print("Corresponding String:");
                          String str2= b.nextLine();
                          linkedHashMap.put(str1,str2);
              }

              // Get a set of all the entries (key - value pairs)
              // contained in the LinkesHashMap
              Set entrySet = linkedHashMap.entrySet();

              // Obtain an Iterator for the entries Set
              Iterator it = entrySet.iterator();

              // Iterate through LinkedHashMap entries
        System.out.println("LinkedHashMap entries : ");
        while(it.hasNext())
           // iterating over each element using it next()
           // method
           System.out.println(it.next());
              }
      }
```

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
```

Aim:

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

Source Code:

HashMapclass.java

```
import java.util.HashMap;
import java.util.*;
// Main class
public class HashMapclass{

      // Main driver method
      public static void main(String[] args){

               HashMap<String,Integer> map = new HashMap<>();
               System.out.print("No.Of Mapping Elements in HashMap:");
               Scanner sc= new Scanner(System.in);
               int n = sc.nextInt();
               for (int i=0; i<n; i++){
                           System.out.print("String:");
                           Scanner b = new Scanner(System.in);
                           String str= b.nextLine();
                           System.out.print("Integer:");
                           int num= b.nextInt();
                           map.put(str,num);
               }

               // using for-each loop for iteration over Map.entrySet()
                       for (Map.Entry<String,Integer> entry : map.entrySet())
                                                            System.out.println("Key = "
+ entry.getKey() +", Value = "

 + entry.getValue());

               // Printing elements in object of Map
               System.out.println(map);
      }
}
```

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}
```

Aim:

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

Source Code:

Linkedlist.java

```
// Java code to illustrate listIterator()
import java.io.*;
import java.util.*;
import java.util.LinkedList;
import java.util.ListIterator;

public class Linkedlist{
      public static void main(String args[])
   {
            System.out.println("No.Of Strings in LinkedList:");
            Scanner sc= new Scanner(System.in);
            int n = sc.nextInt();
            // Creating an empty LinkedList
            LinkedList<String> ll = new LinkedList<String>();

            for (int i=0; i<n; i++){
                        System.out.println("Enter the String:");
                        Scanner b = new Scanner(System.in);
                        String str= b.nextLine();
                        ll.add(str);
            }

            // Displaying the linkedlist
            System.out.println("LinkedList:" + ll);

            // Setting the ListIterator at a specified position
            ListIterator list_Iter = ll.listIterator(0);

            // Iterating through the created list from the position
            System.out.println("The List is as follows:");
            while(list_Iter.hasNext()){
                        System.out.println(list_Iter.next());
            }
   }
}
```

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:2023-12-10

Aim:

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

Source Code:

ArraylistExample.java

```
    import java.util.*;
    class ArraylistExample{
                public static void main(String args[]){
                                  Scanner sc = new Scanner(System.in);
                              System.out.println("Enter ArrayList length: ");
                              int n = sc.nextInt();
                                  ArrayList<Integer> arrli=new ArrayList<Integ
er>();//Creating arraylist
                                  for (int i=1; i<=n; i++){

                                                      arrli.add(i);
                                 }
                                  //Traversing list through Iterator
                                  Iterator itr=arrli.iterator();
                                  System.out.println("ArrayList printing by us
ing Iterator: ");
                                  while(itr.hasNext()){
                                                          System.out.println(i
tr.next());
                                 }
                }
   }
```

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
```

Aim:

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

Source Code:

HashTableclass.java

```
import java.util.*;
import java.util.Enumeration;
class HashTableclass {
      public static void main(String[] args)
   {
            Hashtable<Integer, String> ht = new Hashtable<>();
            System.out.print("No.Of Mapping Elements in HashTable:");
            Scanner sc= new Scanner(System.in);
            int n = sc.nextInt();
            for (int i=0; i<n; i++){
                        System.out.print("Rank:");
                        Scanner b = new Scanner(System.in);
                        int num= b.nextInt();
                        System.out.print("Name:");
                        Scanner a = new Scanner(System.in);
                        String str= a.nextLine();
                        ht.put(num,str);
            }
            Enumeration<Integer> e = ht.keys();
            while (e.hasMoreElements()) {
                        int key = e.nextElement();
                        System.out.println("Rank : " + key+ "\t\t Name : "+ h
t.get(key));
            }
   }
}
```

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`