

S.No: 1

Exp. Name: ***sample programs on operator precedence and associativity***

Date: 2023-09-11

Aim:

Write a java program to demonstrate operator precedence and associativity

Source Code:**OperatorPrecedence.java**

```

import java.util.Scanner;
class OperatorPrecedence
{
    public static void main(String[] args)
    {
        int x, result;
        System.out.print("Enter a num: ");
        Scanner sc=new Scanner(System.in);
        x=sc.nextInt();
        result=x++ +x++*--x/x++ - --x+3>>1|2;
        System.out.println("The operation going is x++ + x++ * --x / x++ - -x + 3 >> 1 | 2");
        System.out.println("result = "+result);
    }
}

```

Execution Results - All test cases have succeeded!**Test Case - 1****User Output**

Enter a num:

4

The operation going is x++ + x++ * --x / x++ - --x + 3 >> 1 | 2

result = 3

Test Case - 2**User Output**

Enter a num:

-3

The operation going is x++ + x++ * --x / x++ - --x + 3 >> 1 | 2

result = 2

S.No: 2

Exp. Name: **Sample program on java to demonstrate Control structures**

Date: 2023-09-11

Aim:

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

Source Code:

Control.java

```
import java.util.Scanner;
class Control
{
    public static void main(String args[])
    {
        int x,y,z;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter first num : ");
        x=sc.nextInt();
        System.out.print("Enter second num : ");
        y=sc.nextInt();
        z=x+y;
        if(z<20)
            System.out.println("x + y is less than 20");
        else
            System.out.println("x + y is greater 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: 3

Exp. Name: **Sample Program to demonstrate constructor**

Date: 2023-09-11

Aim:

Write a program to demonstrate constructor class

Source Code:

Student.java

```
class Student {  
    int id;  
    String name;  
    void display(){  
        System.out.println(id+" "+name);  
    }  
    public static void main(String args[]){  
        Student s1=new Student();  
        Student s2=new Student();  
        s1.display();  
        s2.display();  
    }  
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

0 null

0 null

S.No: 4

Exp. Name: **Sample program to demonstrate
destructor**

Date: 2023-09-11

Aim:

Write a program to demonstrate destructor class

Source Code:

DestructorExample.java

```
public class DestructorExample{
    public static void main(String[] args)
    {
        DestructorExample de=new DestructorExample();
        de.finalize();
        de=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

Aim:

Write a Java program to print Half Pyramid pattern.

Source Code:**HalfPyramid.java**

```
import java.util.Scanner;
public class HalfPyramid{
    public static void main(String[] args){
        int i,j;
        Scanner input=new Scanner(System.in);
        System.out.print("Enter no of rows : ");
        int n=input.nextInt();
        for(i=1;i<=n;i++){
            for(j=1;j<=i;j++)
                System.out.print("* ");
            System.out.print("\n");
        }
    }
}
```

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 pyramid pattern**

Date: 2023-09-11

Aim:

Write a Program to Print Inverted Half Pyramid Pattern

Source Code:

HalfPyramidRev.java

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

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: 2023-09-11

Aim:

Write a Program to Print Hollow Inverted half Pyramid Pattern

Source Code:

HollowHalfPyramidRev.java

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

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

* * *

* *

*

Aim:

Write a Program to Print Pyramid Pattern

Source Code:**Pyramid.java**

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

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: 2023-09-11

Aim:

Write a Program to Print inverted Pyramid Pattern

Source Code:

PyramidRev.java

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

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 a Program to print the Hollow pyramid pattern

Source Code:**PyramidGap.java**

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

```

class Vehicle {
    String color;
    int speed, size;
    public void setVehicleAttributes(String col, int s, int sz) {
        color = col;
        speed = s;
        size = sz;
    }
}
class Car extends Vehicle {
    int CC, gears;
    public void setCarAttributes(int c, int g){
        CC = c;
        gears = g;
    }
    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 b1 = new Car();
        b1.setVehicleAttributes(args[0], Integer.parseInt(args[1]),
        Integer.parseInt(args[2]));
        b1.setCarAttributes(Integer.parseInt(args[3]),
        Integer.parseInt(args[4]));
        b1.displayCarAttributes();
    }
}

```

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: 2023-11-09

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**

```
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

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 obj = new Demo();
        Human obj2 = new Human();
        obj.walk();
        obj2.walk();
    }
}
```

Execution Results - All test cases have succeeded!**Test Case - 1****User Output**

Boy walks

Human walks

Aim:

Write a program on method overloading

Source Code:**Sample.java**

```
class DisplayOverloading {  
    public void test(char n){  
        System.out.println(n);  
    }  
    public void test(char n,int i){  
        System.out.println(n + " " + i);  
    }  
}  
class Sample{  
    public static void main(String args[]){  
        DisplayOverloading s=new DisplayOverloading();  
        s.test('a');  
        s.test('a',10);  
    }  
}
```

Execution Results - All test cases have succeeded!**Test Case - 1****User Output**

a
a 10

Aim:

Write a program on method overriding

Source Code:**Bike.java**

```
class vehicle {  
    void run() {  
        System.out.println("Bike is running safely");  
    }  
}  
class Bike extends vehicle{  
    public static void main(String args[])  
    {  
        Bike obj=new Bike();  
        obj.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(car1,
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: 2023-11-09

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(ArithmaticException 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

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: 2023-11-09

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(ArithmecticException 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: 2023-11-09

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: 2023-12-24

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[]={};
        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: 2023-12-24

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;  
    private 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);
```

```
// Let the thread sleep for a while.
```

```
Thread.sleep(200);
```

```
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(300);
            R1.suspend();
        }

        System.out.println("Suspending First Thread");
        Thread.sleep(300);
        R1.resume();

        System.out.println("Resuming First Thread");
        R2.suspend();

        System.out.println("Suspending thread Two");
        Thread.sleep(300);
        R2.resume();

        System.out.println("Resuming thread Two");
        } catch (InterruptedException e) {
            System.out.println("Main
thread Interrupted");
        }
    try {
}

        System.out.println("Waiting for threads to finish.");
    }
}

```

```

        } catch (InterruptedException e) {
            System.out.println("Main
thread Interrupted");
        }
        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.

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
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.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: 2023-11-13

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 subclasses 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();
    }
}

```

```
    }  
}
```

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: 2023-12-24

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 {
    private String message;
    private int delay;
    private int count;
    public Printer(String message, int delay, int count) {
        this.message = message;
        this.delay = delay;
        this.count = count;
    }
    public void run() {
        for (int i = 0; i < count; i++) {
            try {
                System.out.println(message);
                Thread.sleep(delay * 500);
            } 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: 2023-12-24

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
FileReader(file));
            String line = "", oldtext = "";
            while((line = reader.readLine()) != null) {
                oldtext += line + "\r\n";
            }
            reader.close();
            String newtext = oldtext.replaceAll("This is test string
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 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: 2023-11-13

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: 2023-12-24

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 Stack {
    void push(int item);
    int pop();
}

class FixedSizeStack implements Stack {
    private int stck[];
    private int tos;

    FixedSizeStack(int size) {
        stck = new int[size];
        tos = -1;
    }
    // Push an item onto the stack
    public 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
    public int pop() {
        if(tos < 0) {
            System.out.println("Stack underflow");
            return 0;
        } else {
            return stck[tos--];
        }
    }
}
class DynamicStack {
    private int stck[];
    private int tos;

    DynamicStack(int size) {
        stck = new int[size];
        tos = -1;
    }
    // Push an item onto the stack
    public void push(int item) {
        if(tos == stck.length-1) { // use length member
            System.out.println("Stack is full and increased");
            stck=doublesize(stck);
        } else {
            stck[++tos] = item;
        }
    }
    // Pop an item from the stack
    public int pop() {
        if(tos < 0) {
            System.out.println("Stack underflow");
            return 0;
        } else {
            return stck[tos--];
        }
    }
}

```

```

        int[] doublesize(int []arr) {
            int[] newArray = new int[stck.length * 2];
            for(int i = 0; i<stck.length; i++) {
                newArray[i] = stck[i];
            }
            return newArray;
        }
    }

public class StaticAndDynamicStack {
    public static void main(String args[]) {
        FixedSizeStack mystack1 = new FixedSizeStack(5);
        DynamicStack mystack2 = new DynamicStack(5);
        // push some numbers onto the stack
        for(int i=0; i<5; i++)
            mystack1.push(i);
        for(int i=0; i<10; i++)
            mystack2.push(i);
        // pop those numbers off the stack
        System.out.println("Stack in mystack1:");
        for(int i=0; i<5; i++) {
            System.out.println(mystack1.pop());
        }
        System.out.println ("Stack in mystack2 :");
        for (int i=0; i<10; i++)
            System.out.println(mystack2.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: 2023-12-24

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;
    }
}

```

```

        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

S.No: 30

Exp. Name: ***Write java program(s) that use collection framework classes.(TreeMap class)***

Date: 2023-12-03

Aim:

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

Source Code:

Treemap.java

```

// Java Program to Iterate Over Entries 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 getKey() 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

S.No: 31

Exp. Name: **Write java program(s) that use collection framework classes.(TreeSet class)**

Date: 2023-12-04

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[]){
        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);
        }
        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

S.No: 32

Exp. Name: ***Write java program(s) that use collection framework classes.(LinkedHashMap class)***

Date: 2023-12-04

Aim:

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

Source Code:**LinkedHashMapclass.java**

```

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){
        LinkedHashMap<String, String> linkedHashMap= new
        LinkedHashMap<String, String>();
        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);
        }
        Set entrySet = linkedHashMap.entrySet();
        Iterator it = entrySet.iterator();
        System.out.println("LinkedHashMap entries : ");
        while (it.hasNext())
            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

S.No: 33

Exp. Name: ***Write java program(s) that use collection framework classes.(HashMap class)***

Date: 2023-12-04

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);
        }
        for (Map.Entry<String, Integer> entry : map.entrySet())
            System.out.println("Key = " + entry.getKey()
+", Value = " + entry.getValue());
        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**

```
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();
        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);
        }
        System.out.println("LinkedList:" + ll);
        ListIterator list_Iter = ll.listIterator(0);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

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: 2023-12-04

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<Integer>();for (int i=1;
i<=n; i++){
            arrli.add(i);
        }
        Iterator itr=arrli.iterator();
        System.out.println("ArrayList printing by using Iterator: ");
        while(itr.hasNext()){
            System.out.println(itr.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

S.No: 36

Exp. Name: ***Write java program(s) that use collection framework classes.(HashTable class)***

Date: 2023-12-03

Aim:

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

Source Code:

HashTableclass.java

```

// Java Program to Iterate through HashTable
// using enumeration interface

// Importing required packages
import java.util.*;
import java.util.Enumeration;

// MAin Class
class HashTableclass {

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

        // Creating Hashtable object where key is of Integer
        // type and value is of String type
        Hashtable<Integer, String> ht = new Hashtable<>();

        // Putting key-value pairs to HashTable object
        // Custom input entries
        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);
        }

        // Creating Enumeration interface
        // and get keys() from Hashtable
        Enumeration<Integer> e = ht.keys();

        // Iterating through the Hashtable
        // object

        // Checking for next element in Hashtable object
        // with the help of hasMoreElements() method
        while (e.hasMoreElements()) {

            // Getting the key of a particular entry
            int key = e.nextElement();

            // Print and display the Rank and Name
            System.out.println("Rank : " + key
                + "\t\t Name : "
                + ht.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