S.No: 1	Exp. Name: ***sample programs on operator precedence and associativity***	**Date: 2023-09-27**

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: 2023-09-28**

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: 2023-09-28

Aim:
Write a program to demonstrate constructor class
Source Code:

Student.java

```
class Student{
        int num;
        String name;
        void display(){
                System.out.println(num+" "+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: 2023-09-28**

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
```

S.No: 5	Exp. Name: ***A program to print Half pyramid pattern***	**Date: 2023-09-28**

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[])
{
    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=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 pyramin pattern***	**Date: 2023-09-28**

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 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;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: 2023-09-28**

Aim:

Write a Program to Print Hollow Inverted half Pyramid Pattern

Source Code:

HollowHalfPyramidRev.java

```java
import java.util.Scanner;
public class HollowHalfPyramidRev
{
        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=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
* * *
* *
*

S.No: 8	Exp. Name: ***A program to print Pyramid Pattern***	Date: 2023-09-28

Aim:
Write a Program to Print Pyramid Pattern
Source Code:

Pyramid.java

```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: 2023-09-28**

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
* * * * * * 
 * * * * * 
  * * * * 
   * * * 
    * * 
     * 
```

S.No: 10	Exp. Name: ***A program to print Hollow Pyramid Pattern***	**Date: 2023-09-28**

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,j,n;
                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
```

```
     *
    * *
   *   *
  *     *
 *       *
* * * * * *
```

S.No: 11	Exp. Name: ***A program to illustrate Inheritance***	**Date: 2023-10-14**

Aim:

Write Java program on use of Inheritance.

Create a classVehicle

- 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 classCarwhich is derived from the classVehicle

- 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.Scanner;
class Vehicle {
        String color;
        int speed;
        int size;
        void setVehicleAttributes(String c,String s,String sp){
                color = c;
                speed = Integer.parseInt(s);
                size = Integer.parseInt(sp);
        }
}
class Car extends Vehicle{
        int CC;
        int gears;
        void setCarAttributes(String c,String s,String sp,String
        cce,String gear){
                setVehicleAttributes(c,s,sp);
                CC = Integer.parseInt (cce);
                gears = Integer.parseInt(gear);
                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 b1 = new Car();

b1.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

```
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 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
```

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

```
class DisplayOverloading
{
        public void dis(char c)
        {
                System.out.println(c);
        }
        public void dis(char c,int num)
        {
                System.out.println(c+" "+num);
        }
}
class Sample
{
        public static void main(String args[])
        {
                DisplayOverloading obj=new DisplayOverloading();
                obj. dis('a');
                obj.dis('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 is good");
        }
}
class safe extends Vehicle
{
        void run()
        {
                System.out.println("Bike is running safely");
        }
}
class Bike
{
        public static void main(String args[])
        {
                Vehicle obj=new safe();
                obj.run();
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Bike is running safely
```

S.No: 16	Exp. Name: ***Write a Java program to implement Interface***	Date: 2023-11-06

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{
        public String getName();
        public int getMaxSpeed();
        public default void applyBreak(){
                System.out.println("applying Break on "+getName());
        }
        public static Car getFastestCar(Car a,Car b)
        {
                if(a.getMaxSpeed()>b.getMaxSpeed())
                return a;
                else
                return b;
        }
}
class BMW implements Car{
        String name;
        int speed;
        public BMW(String n,String s){
                speed=Integer.parseInt (s);
                name=n;
        }
        public String getName(){
                return name;
        }
        public int getMaxSpeed(){
                return speed;
        }
}
class Audi implements Car {
        String name;
        int speed;
        public Audi(String  n,String s){
                speed=Integer.parseInt(s);
                name=n;
        }
        public String getName(){
                return name;
        }
        public int getMaxSpeed(){
                return speed;
        }
}
public class MainApp{
        public static void main(String args[]){
                BMW bmw=new BMW(args[0],args[1]);
                Audi audi=new Audi(args[2],args[3]);
                Car max=Car.getFastestCar(bmw,audi);
                System.out.println("Fastest car is : "+max.getName());
        }
}
```