

NAME: HIBA N
TO: PUNITH SIR

ASSIGNMENT

Math.pow():

The [java.lang.Math.pow\(\)](#) is used to calculate a number raised to the power of some other number. This function accepts two parameters and returns the value of first parameter raised to the second parameter.

There are some special cases as listed below:

- If the second parameter is positive or negative zero then the result will be 1.0.
- If the second parameter is 1.0 then the result will be same as that of the first parameter.
- If the second parameter is NaN then the result will also be NaN.
- The function **java.lang.Math.pow()** always returns a double datatype.

SYNTAX:

```
public static double pow(double a, double b)
```

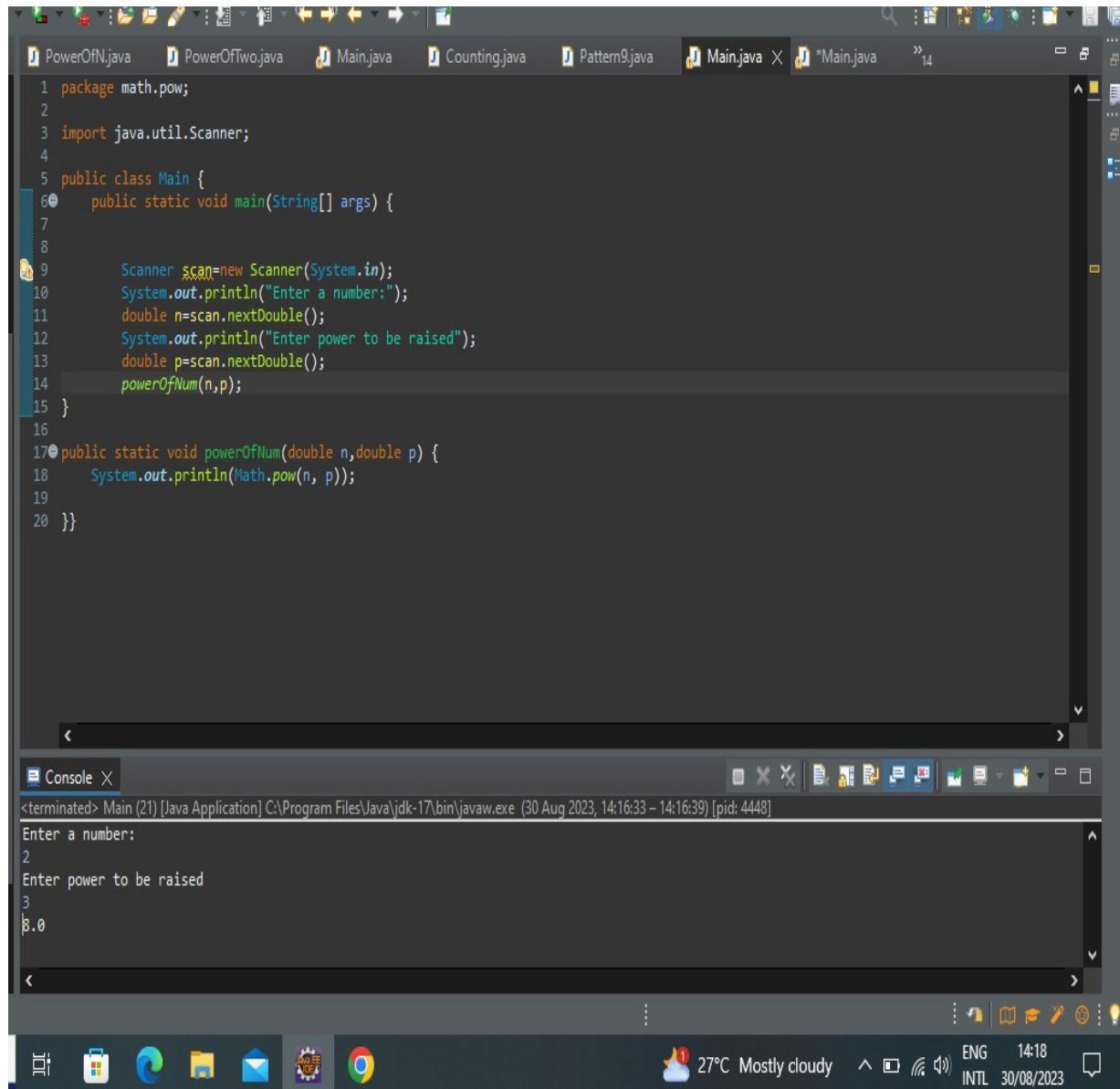
a: this parameter is the base

b: this parameter is the exponent.

Return:

This method returns a^b .

EXAMPLE :



The screenshot shows an IDE with a Java file named `Main.java` open. The code defines a `math.pow` package with a `Main` class. The `main` method prompts the user for a number and a power, then calls `powerOfNum` to calculate the result. The `powerOfNum` method uses `Math.pow` to perform the calculation and prints the result. The console output shows the program running successfully, with the user inputting '2' for the number and '3' for the power, resulting in '8.0'.

```
1 package math.pow;
2
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) {
7
8
9         Scanner scan=new Scanner(System.in);
10        System.out.println("Enter a number:");
11        double n=scan.nextDouble();
12        System.out.println("Enter power to be raised");
13        double p=scan.nextDouble();
14        powerOfNum(n,p);
15    }
16
17    public static void powerOfNum(double n,double p) {
18        System.out.println(Math.pow(n, p));
19    }
20 }
```

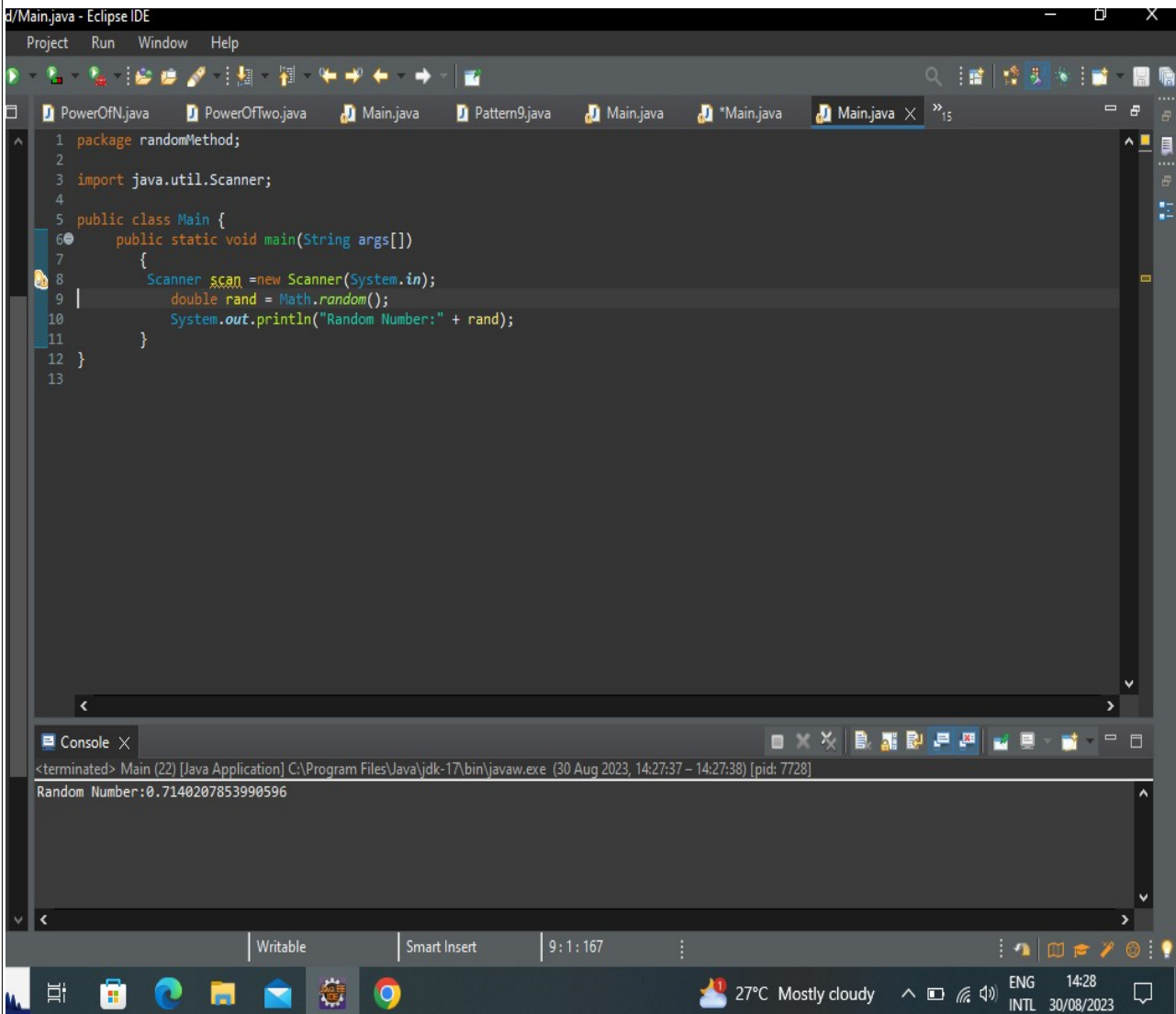
Console Output:

```
<terminated> Main (21) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (30 Aug 2023, 14:16:33 - 14:16:39) [pid: 4448]
Enter a number:
2
Enter power to be raised
3
8.0
```

Math.random():

The **java.lang.Math.random()** method returns a pseudorandom double type number greater than or equal to 0.0 and less than 1.0. When this method is first called, it creates a single new pseudorandom-number generator, exactly as if by the expression `new java.util.Random`.

EXAMPLE :



The screenshot shows the Eclipse IDE interface. The top menu bar includes 'Project', 'Run', 'Window', and 'Help'. Below the menu is a toolbar with various icons. The main editor window displays a Java file named 'Main.java' with the following code:

```
1 package randomMethod;
2
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String args[])
7     {
8         Scanner scan = new Scanner(System.in);
9         double rand = Math.random();
10        System.out.println("Random Number:" + rand);
11    }
12 }
13
```

Below the editor is a 'Console' window showing the output of the program:

```
<terminated> Main (22) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (30 Aug 2023, 14:27:37 - 14:27:38) [pid: 7728]
Random Number:0.7140207853990596
```

The bottom status bar shows 'Writable', 'Smart Insert', and the time '9:1:167'. The Windows taskbar at the very bottom displays the system clock as '14:28' on '30/08/2023' with weather information '27°C Mostly cloudy'.

Random class:

Random class is part of java. util package. An instance of java Random class is **used to generate random numbers**. This class provides several methods to generate random numbers of type integer, double, long, float etc.

1)**Java.util.Random.doubles()**: Returns an effectively unlimited stream of pseudo random double values, each between zero (inclusive) and one (exclusive)

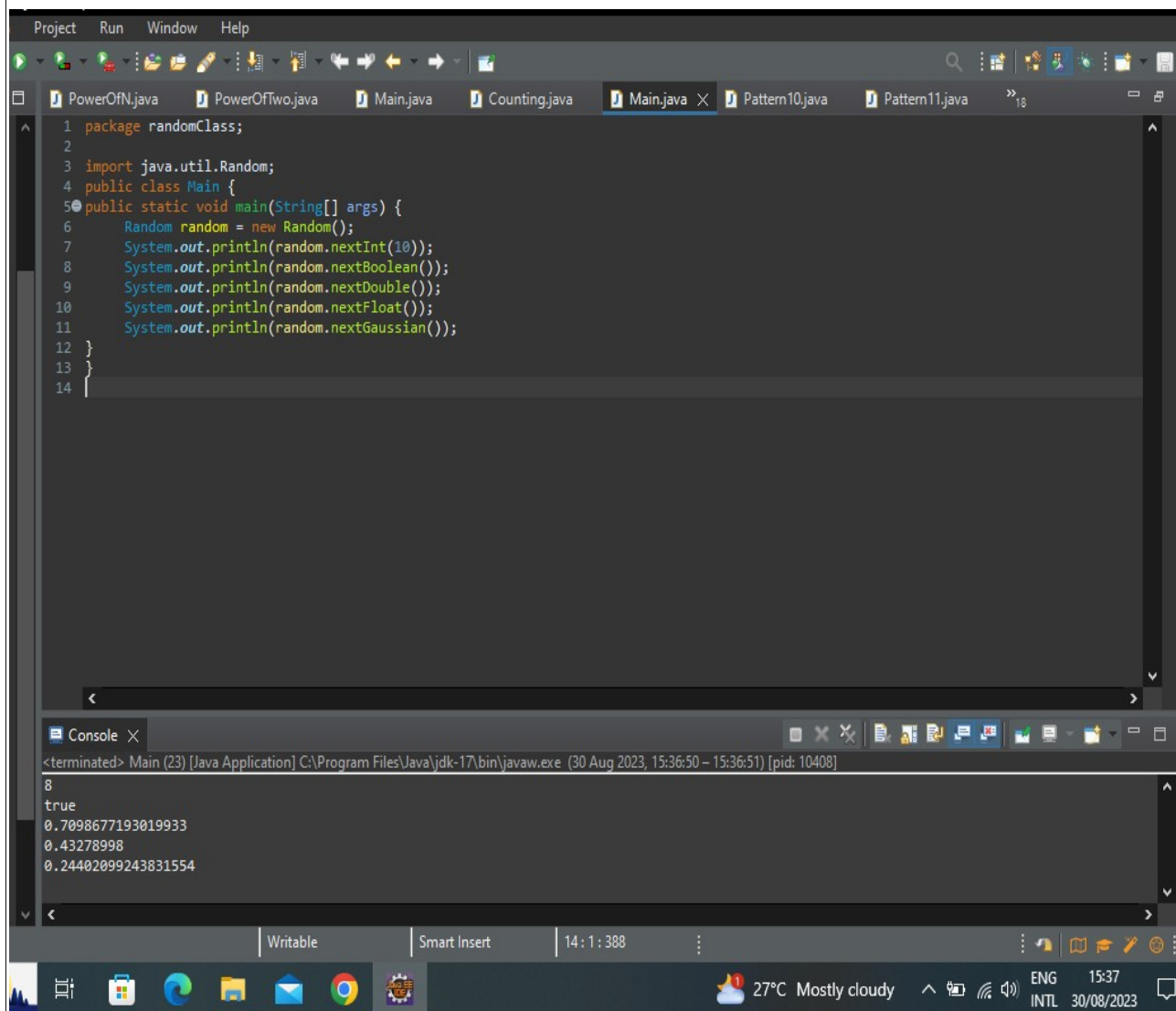
2)**java.util.Random.ints()**: Returns an effectively unlimited stream of pseudo random int values.

3)**java.util.Random.longs()**: Returns an effectively unlimited stream of pseudo random long values.

4)**next(int bits): java.util.Random.next(int bits)** Generates the next pseudo random number.

5)**java.util.Random.nextBoolean()**: Returns the next pseudo random, uniformly distributed boolean value from this random number generator's sequence.

EXAMPLE:



The screenshot shows an IDE with a Java file named `Main.java` containing the following code:

```
1 package randomClass;
2
3 import java.util.Random;
4 public class Main {
5     public static void main(String[] args) {
6         Random random = new Random();
7         System.out.println(random.nextInt(10));
8         System.out.println(random.nextBoolean());
9         System.out.println(random.nextDouble());
10        System.out.println(random.nextFloat());
11        System.out.println(random.nextGaussian());
12    }
13 }
14
```

The console output shows the results of the program execution:

```
<terminated> Main (23) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (30 Aug 2023, 15:36:50 - 15:36:51) [pid: 10408]
8
true
0.7098677193019933
0.43278998
0.24402099243831554
```

The IDE interface includes a menu bar (Project, Run, Window, Help), a toolbar, and a status bar at the bottom showing the temperature (27°C) and date (30/08/2023).