USAGE OF POW IN MATH CLASS

The **java.lang.Math.pow()** is used to return the value of first argument raised to the power of the second argument. The return type of pow() method is double.

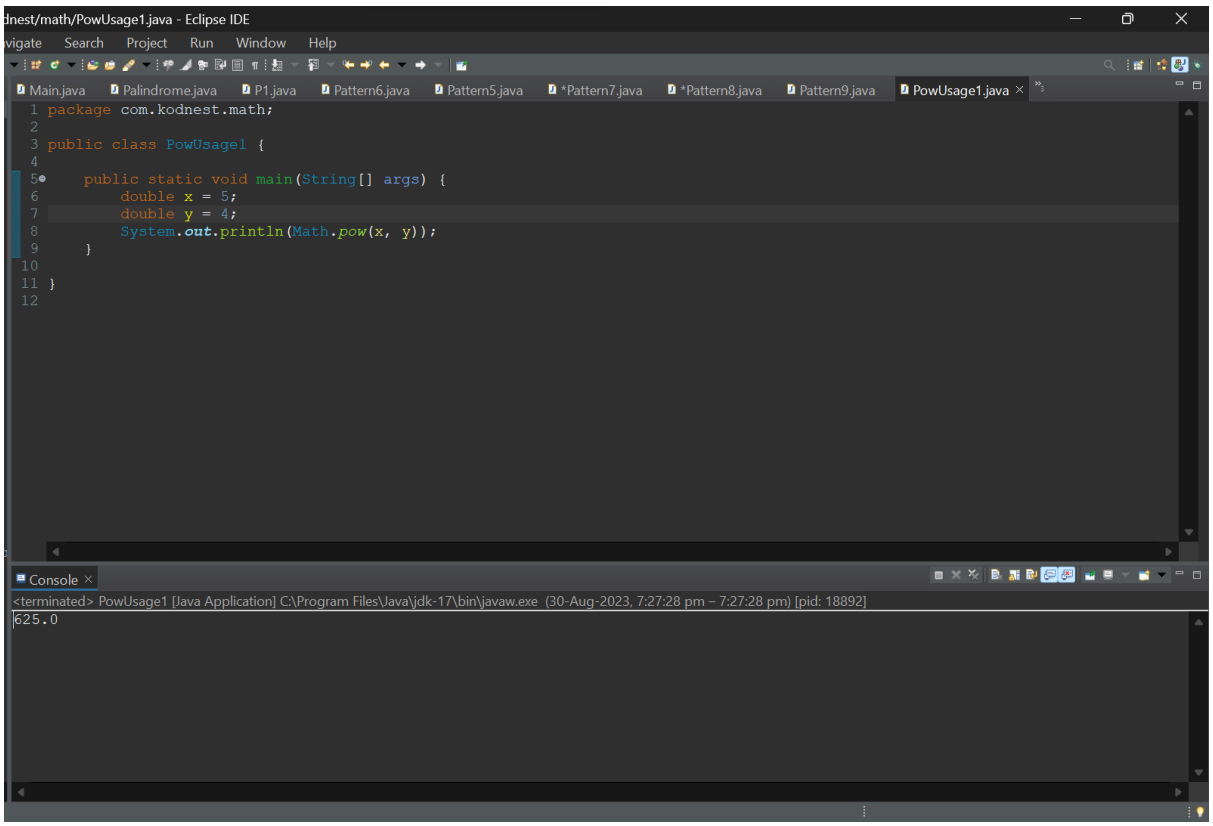
SYNTAX

public static double pow(double a, double b)
a= base b= exponent

This method returns the value of a^b

- If the second argument is positive or negative **Zero**, this method will return **1.0**.
- If the second argument is not a number **(NaN)**, this method will return **NaN**.
- If the second argument is **1**, this method will return the result same as the **first argument**.

Example:

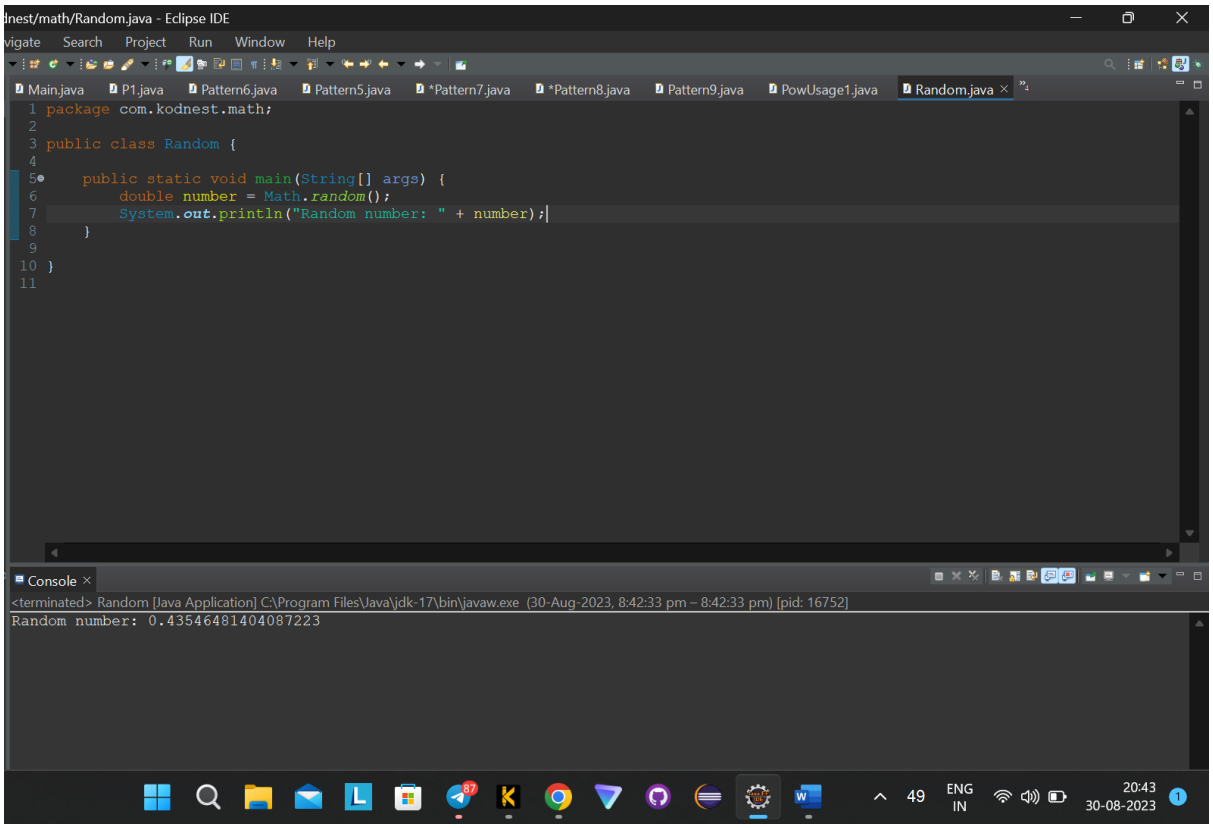


RANDOM METHOD IN MATH CLASS:

The Java `Math.random()` method is used to generate a pseudorandom number, which is a number created with a formula that simulates randomness. The pseudorandom number will be greater than or equal to 0.0 and less than 1.0. In other words, the number generated by `Math.random` is always between 0 and 1, and is a floating-point number.

The random method returns a random double, which is the data type used to store floating-point values.

Example:



RANDOM CLASS

Random class is used to generate pseudo-random numbers in java. An instance of this class is thread-safe. The instance of this class is however cryptographically insecure. This class provides various method calls to generate different random data types such as float, double, int.

Methods:

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

   Syntax:

   ```
   public DoubleStream doubles()
   ```

   Returns:

   a stream of pseudorandom double values

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

   Syntax:

   ```
   public IntStream ints()
   ```

   Returns:

   a stream of pseudorandom int values

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

   Syntax:

   ```
   public LongStream longs()
   ```

   Returns:

   a stream of pseudorandom long values

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

   Syntax:

   ```
   protected int next(int bits)
   ```

   Parameters:

   bits - random bits

```
Returns:
the next pseudo random value from this
random number generator's sequence
```

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

   Syntax:

   ```
   public boolean nextBoolean()

   Returns:
   the next pseudorandom, uniformly distributed boolean value
   from this random number generator's sequence
   ```

6. **java.util.Random.nextBytes(byte[] bytes) :**Generates random bytes and places them into a user-supplied byte array

   Syntax:

   ```
   public void nextBytes(byte[] bytes)
   Parameters:
   bytes - the byte array to fill with random bytes
   Throws:
   NullPointerException - if the byte array is null
   ```

7. **java.util.Random.nextDouble():** Returns the next pseudo random, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence

   Syntax:

   ```
   public double nextDouble()
   Returns:
   the next pseudo random, uniformly distributed double
   value between 0.0 and 1.0 from this
   random number generator's sequence
   ```

8. **java.util.Random.nextFloat():** Returns the next pseudo random, uniformly distributed float value between 0.0 and 1.0 from this random number generator's sequence

   Syntax:

   ```
   public float nextFloat()
   Returns:
   the next pseudorandom, uniformly distributed float value
   between 0.0 and 1.0 from this
   random number generator's sequence
   ```

9. **java.util.Random.nextGaussian():** Returns the next pseudo random, Gaussian ("normally") distributed double value with mean

0.0 and standard deviation 1.0 from this random number generator's sequence

Syntax:

```
public double nextGaussian()
```

Returns:

```
the next pseudorandom, Gaussian ("normally") distributed
double value with mean 0.0 and standard deviation 1.0 from
this random number generator's sequence
```

10.java.util.Random.nextInt()**:** Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence

Syntax:

```
public int nextInt()
```

Returns:

```
the next pseudorandom, uniformly distributed int value from
this random number generator's sequence
```

11.java.util.Random.nextInt(int bound): Returns a pseudo random, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence

Syntax:

```
public int nextInt(int bound)
Parameters:
bound - the upper bound (exclusive). Must be positive.
Returns:
the next pseudorandom, uniformly distributed int value
between zero (inclusive) and bound (exclusive) from this
random number generator's sequence
Throws:
IllegalArgumentException - if bound is not positive
```

12.java.util.Random.nextLong(): Returns the next pseudorandom, uniformly distributed long value from this random number generator's sequence

Syntax:

```
public long nextLong()
Returns:
the next pseudorandom, uniformly distributed long value
from this random number generator's sequence
```