## 1. Mathematical Functions

Java Math class provides several methods to work on math calculations like min(), max(), avg(), sin(), cos(), tan(), round(), ceil(), floor(), abs() etc.

```java
public class JavaMathExample1
{
    public static void main(String[] args)
    {
        double x = 28;
        double y = 4;

        // return the maximum of two numbers
        System.out.println("Maximum number of x and y is: " +Math.max(x, y));

        // return the square root of y
        System.out.println("Square root of y is: " + Math.sqrt(y));

        //returns 28 power of 4 i.e. 28*28*28*28
        System.out.println("Power of x and y is: " + Math.pow(x, y));

        // return the logarithm of given value
        System.out.println("Logarithm of x is: " + Math.log(x));
        System.out.println("Logarithm of y is: " + Math.log(y));

        // return the logarithm of given value when base is 10
        System.out.println("log10 of x is: " + Math.log10(x));
        System.out.println("log10 of y is: " + Math.log10(y));

        // return the log of x + 1
        System.out.println("log1p of x is: " +Math.log1p(x));

        // return a power of 2
        System.out.println("exp of a is: " +Math.exp(x));

        // return (a power of 2)-1
        System.out.println("expm1 of a is: " +Math.expm1(x));
    }
}
```

**Output:**

```
Maximum number of x and y is: 28.0
Square root of y is: 2.0
Power of x and y is: 614656.0
Logarithm of x is: 3.332204510175204
Logarithm of y is: 1.3862943611198906
log10 of x is: 1.4471580313422192
log10 of y is: 0.6020599913279624
log1p of x is: 3.367295829986474
exp of a is: 1.446257064291475E12
expm1 of a is: 1.446257064290475E12
```

---------------------------------------------------------

```java
public class JavaMathExample2
{
    public static void main(String[] args)
    {
        double a = 30;

        // converting values to radian
        double b = Math.toRadians(a);

        // return the trigonometric sine of a
        System.out.println("Sine value of a is: " +Math.sin(a));

        // return the trigonometric cosine value of a
        System.out.println("Cosine value of a is: " +Math.cos(a));

        // return the trigonometric tangent value of a
        System.out.println("Tangent value of a is: " +Math.tan(a));

        // return the trigonometric arc sine of a
        System.out.println("Sine value of a is: " +Math.asin(a));

        // return the trigonometric arc cosine value of a
        System.out.println("Cosine value of a is: " +Math.acos(a));

        // return the trigonometric arc tangent value of a
        System.out.println("Tangent value of a is: " +Math.atan(a));

        // return the hyperbolic sine of a
        System.out.println("Sine value of a is: " +Math.sinh(a));

        // return the hyperbolic cosine value of a
        System.out.println("Cosine value of a is: " +Math.cosh(a));
        // return the hyperbolic tangent value of a
        System.out.println("Tangent value of a is: " +Math.tanh(a));
    }
}
```

**Output:**

```
Sine value of a is: -0.9880316240928618
Cosine value of a is: 0.15425144988758405
Tangent value of a is: -6.405331196646276
Sine value of a is: NaN
Cosine value of a is: NaN
Tangent value of a is: 1.5374753309166493
Sine value of a is: 5.343237290762231E12
Cosine value of a is: 5.343237290762231E12
Tangent value of a is: 1.0
```

------------------------------***************************---------------------------

What is **NaN** argument ?

A constant holding a Not-a-Number (NaN) value of type double. It is equivalent to the value returned by Double.longBitsToDouble(0x7ff8000000000000L).

---------------*******************----------------****************------------

- **Java code explaining abs(), acos(), toRadians() method in Math class.**

```
// Java program explaining Math class methods
// abs(), acos(), toRadians()

import java.math.*;
public class NewClass
{

    public static void main(String[] args)
    {
        // Declaring the variables
        int Vali = -1;
        float Valf = .5f;

        // Printing the values
```

```
                System.out.println("Initial value of int  : "+Vali);
                System.out.println("Initial value of int  : "+Valf);


                // Use of .abs() method to get the absoluteValue
                int Absi = Math.abs(Vali);
                float Absf = Math.abs(Valf);

                System.out.println("Absolute value of int : "+Absi);
                System.out.println("Absolute value of int : "+Absf);
                System.out.println("");

                // Use of acos() method
                // Value greater than 1, so passing NaN
                double Acosi = Math.acos(60);
                System.out.println("acos value of Acosi : "+Acosi);
                double x = Math.PI;

                // Use of toRadian() method
                x = Math.toRadians(x);
                double Acosj = Math.acos(x);
                System.out.println("acos value of Acosj : "+Acosj);

            }
        }
```

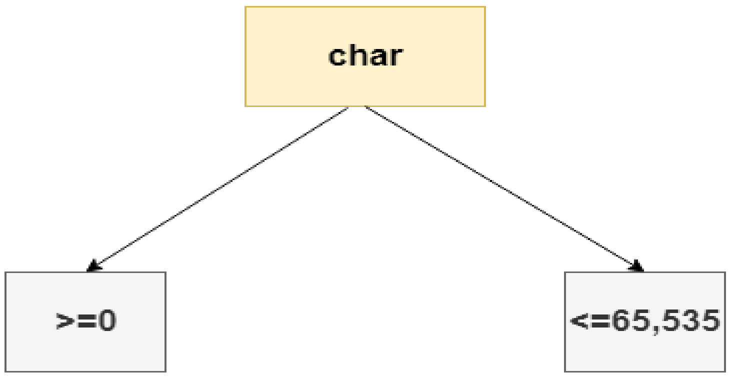
**Output**:

```
Initial value of int  : -1
Initial value of int  : 0.5
Absolute value of int : 1
Absolute value of int : 0.5
acos value of Acosi : NaN
acos value of Acosj : 1.5159376794536454
```

## 2. Character Data Type in Java

The Java char keyword is a primitive data type. It is used to declare the character-type variables and methods. It is capable of holding the unsigned 16-bit Unicode characters.

- The char range lies between 0 to 65,535 (inclusive).
- Its default value is '\u0000'.
- Its default size is 2 byte.
- It is used to store characters.



- **Why char uses 2 bytes in java?**

It is because Java uses Unicode system not ASCII code system.

- **What is \u0000?**

The \u0000 is the lowest range of the Unicode system.

-------------------*********************-----------------------------

## Example 1

**Let's see a simple example of displaying characters.**

```
public class CharExample1 {

    public static void main(String[] args) {

        char char1='a';
        char char2='A';


        System.out.println("char1: "+char1);
        System.out.println("char2: "+char2);
    }
}
```

**Output:**

```
char1: a
char2: A
```

## Example 2

**//You don't have to know the Unicode of each character; these are just as examples to show you how Java could deal with these Unicode**

In this example, we provide integer value to char variable. Here, compiler implicitly typecast integer to char and display the corresponding ASCII value.

```java
public class CharExample2 {

    public static void main(String[] args) {

        char char1=65;
        char char2=97;

        System.out.println("char1: "+char1);
        System.out.println("char2: "+char2);

    }
}
```

**Output:**

```
char1: A
char2: a
```

## Example 3

In this example, we typecast the integer value to char explicitly.

```java
public class CharExample3 {

    public static void main(String[] args) {

        int num1=97;
        char char1=(char)num1;

        int num2=65;
        char char2=(char)num2;

        System.out.println("char1: "+char1);
        System.out.println("char2: "+char2);

    }

}
```

**Output:**

```
char1: a
char2: A
```

## Example 4

In this example, we increment the provided char value by 1.

```
public class CharExample5 {

    public static void main(String[] args) {

        char char1='A';
        char1=(char)(char1+1);

        System.out.println("char: "+char1);

    }
}
```

**Output:**

```
char: B
```

## Example 5

**//We can understand this example after we know how array works in Java**

Let's see an example to break the string in the form of characters.

```
import java.util.Arrays;

public class CharExample6 {

```

```java
    public static void main(String[] args) {

        String str="javatpoint";
        char[] ch=str.toCharArray();

        System.out.println("String: "+str);
        System.out.println("char: "+Arrays.toString(ch));

    }

}
```

**Output:**

```
String: javatpoint
char: [j, a, v, a, t, p, o, i, n, t]
```

## 3. String in Java

In Java, a string is a sequence of characters. For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'.

We use **double quotes** to represent a string in Java. For example,

```java
// create a string
String type = "Java programming";
```

Here, we have created a string variable named `type`. The variable is initialized with the string `Java Programming`.

**Note**: Strings in Java are not primitive types (like `int`, `char`, etc). Instead, all strings are objects of a predefined class named `String`.
And, all string variables are objects of the `String` class.

### Example: Create a String in Java

```
class Main {
  public static void main(String[] args) {

    // create strings
    String first = "Java";
    String second = "Python";
    String third = "JavaScript";

    // print strings
    System.out.println(first);   // print Java
    System.out.println(second);  // print Python
    System.out.println(third);   // print JavaScript
  }
}
```

In the above example, we have created three strings named first, second, and third. Here, we are directly creating strings like primitive types.
However, there is another way of creating Java strings (using the new keyword).

## Java String Operations

Java String provides various methods to perform different operations on strings. We will look into some of the commonly used string operations.

### 1. Get Length of a String

To find the length of a string, we use the length() method of the String. For example,

```
class Main {
  public static void main(String[] args) {

    // create a string
    String greet = "Hello! World";
    System.out.println("String: " + greet);
    // get the length of greet
    int length = greet.length();
    System.out.println("Length: " + length);
  }
}
```

**Output**

```
String: Hello! World
Length: 12
```

In the above example, the `length()` method calculates the total number of characters in the string and returns it.

## 2. Join two Strings

We can join two strings in Java using the `concat()` method. For example,

```
class Main {
  public static void main(String[] args) {

    // create first string
    String first = "Java ";
    System.out.println("First String: " + first);

    // create second
    String second = "Programming";
    System.out.println("Second String: " + second);

    // join two strings
    String joinedString = first.concat(second);
    System.out.println("Joined String: " + joinedString);
  }
}
```

**Output**

```
First String: Java
Second String: Programming
Joined String: Java Programming
```

In the above example, we have created two strings named `first` and `second`. Notice the statement,

```
String joinedString = first.concat(second);
```

Here, we the `concat()` method joins `first` and `second` and assigns it to the `joinedString`variable. We can also join two strings using the `+` operator in Java.

## 3. Compare two Strings

In Java, we can make comparisons between two strings using the `equals()` method. For example,

```java
class Main {
  public static void main(String[] args) {

    // create 3 strings
    String first = "java programming";
    String second = "java programming";
    String third = "python programming";

    // compare first and second strings
    boolean result1 = first.equals(second);
    System.out.println("Strings first and second are equal: " + result1);

    // compare first and third strings
    boolean result2 = first.equals(third);
    System.out.println("Strings first and third are equal: " + result2);
  }
}
```

**Output**

```
Strings first and second are equal: true
Strings first and third are equal: false
```

In the above example, we have created 3 strings named `first`, `second`, and `third`. Here, we are using the `equal()` method to check if one string is equal to another.
The `equals()` method checks the content of strings while comparing them.

**Note**: We can also compare two strings using the `==` operator in Java. However, this approach is different than the `equals()` method.

➢ **Watch this video to get more ideas about compareTo() method**

https://www.youtube.com/watch?v=iTC43mLZG38&t=176s

## Escape character in Java Strings

The escape character is used to escape some of the characters present inside a string.

Suppose we need to include double quotes inside a string.

```
// include double quote
String example = "This is the "String" class";
```

Since strings are represented by **double quotes**, the compiler will treat "This is the " as the string. Hence, the above code will cause an error.
To solve this issue, we use the escape character \ in Java. For example,

```
// use the escape character
String example = "This is the \"String\" class.";
```

Now escape characters tell the compiler to escape **double quotes** and read the whole text.

## Java Strings are Immutable

In Java, strings are **immutable**. This means, once we create a string, we cannot change that string.
To understand it more deeply, consider an example:

```
// create a string
String example = "Hello! ";
```

Here, we have created a string variable named `example`. The variable holds the string `"Hello! "`.
Now suppose we want to change the string.

```
// add another string "World"
// to the previous tring example
```

```java
example = example.concat(" World");
```

Here, we are using the `concat()` method to add another string `World` to the previous string. It looks like we are able to change the value of the previous string. However, this is not `true`.

Let's see what has happened here,

1. JVM takes the first string `"Hello! "`
2. creates a new string by adding `"World"` to the first string
3. assign the new string `"Hello! World"` to the `example` variable
4. the first string `"Hello! "` remains unchanged

## Creating strings using the new keyword

So far, we have created strings like primitive types in Java.

Since strings in Java are objects, we can create strings using the `new` keyword as well. For example,

```java
// create a string using the new keyword
String name = new String("Java String");
```

In the above example, we have created a string `name` using the `new` keyword. Here, when we create a string object, the `String()` constructor is invoked.

### Example: Create Java Strings using the new keyword

```java
class Main {
  public static void main(String[] args) {

    // create a string using new
    String name = new String("Java String");

    System.out.println(name);  // print Java String
  }
}
```

If we create the string using **new** keyword, we will not be able to use = = to compare between two strings

```
class Main {
  public static void main(String[] args) {

    // create a string using new
    String a = new String("car");
    String b = new String("car");

    if (a == b)
       System.out.println("True");
    else
      System.out.println("False");


  }
}
```

The output will be **False**, even though both strings equal "car".