**1.Java primitive data types**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data type** | **Wrapper class** | **Size** | **Range** |
| byte | Byte | 1 byte | -128 – 127 |
| short | Short | 2 bytes | -32,768 – 32,767 |
| int | Integer | 4 bytes | -2,147,438,648 – 2,147,438,647 |
| long | Long | 8 bytes | -9,223,372,036,854,775,808 - 9,223,372,036,854,775,807 |
| float | Float | 4 bytes | 1.4E-45 - 3.4028235E38 |
| double | Double | 8 bytes | 4.9E-324 - 1.7976931348623157E308 |
| boolean | Boolean | 1 bit | True or False |
| char | Character | 2 bytes | Single character ascii or unicode |

**2. Examples of Increment and decrement operators**

**Increment (++):**

**Pre increment (++a):** int x = 10;

int y = ++x;

System.out.println("y value is: " + y); // y is 11 , x is 11

**Post increment (a++):** int x = 10;

int y = x++;

System.out.println("y value is: " + y); // y is 10 , x is 11

**Decrement (- -):**

**Pre decrement (--a):** int x = 10;

int y = --x;

System.out.println("y value is: " + y); // y is 9, x is 9

**Post decrement (a--):** int x = 10;

int y = x--;

System.out.println("y value is: " + y); // y is 10, x is 9

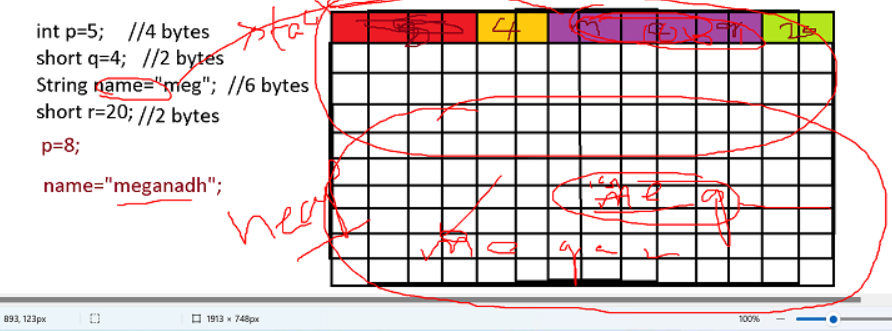
**3. Java code for Factorial of a number**

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) {  
 // write your code here  
 System.*out*.println(*factorial*(0));  
  
 }  
 public static int factorial(int num){  
 int fact = 1;  
 while(num>0){  
 fact \*= num;  
 num--;  
 }  
 return fact;  
 }  
  
  
}

**3. Java code for nCr and nPr**

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) {  
 // write your code here  
 System.*out*.println(*factorial*(0)); //1  
 System.*out*.println(*nCr*(12,5)); //792  
 System.*out*.println(*nPr*(5,3)); // 60  
  
 }  
 public static int factorial(int num){  
 int fact = 1;  
 while(num>0){  
 fact \*= num;  
 num--;  
 }  
 return fact;  
 }  
  
 public static int nCr(int n, int r){  
 return *factorial*(n)/(*factorial*(r)\**factorial*(n-r));  
 }  
  
 public static int nPr(int n, int p){  
 return *factorial*(n)/(*factorial*(n-p));  
 }  
  
  
}

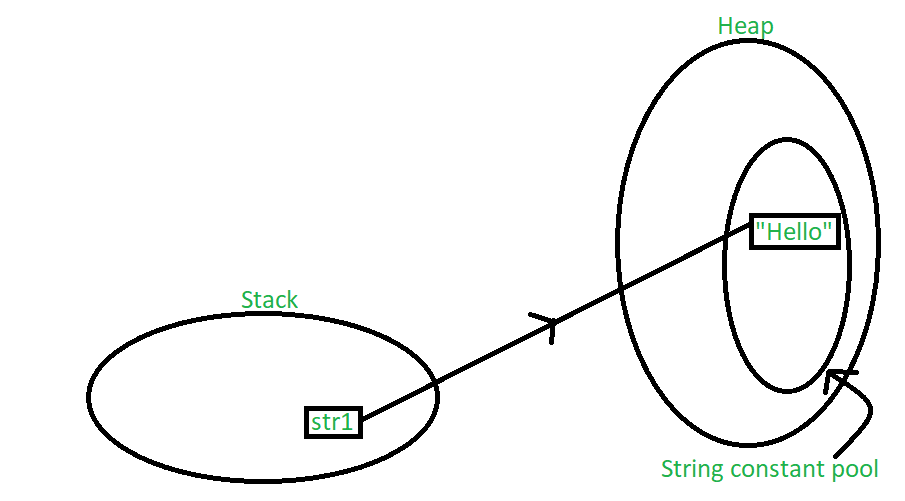
**5. How are strings stored in memory?**

****

* The variable **name which will be stored in stack memory** only store address of the first cell of memory location where “meg” is stored in heap memory.
* And when we change the **name** to “Pranav” java will create a **new string object in heap memory,** because some other variable might be stored after “meg”. And **now the variable name will point to that new memory location**.
* The old memory will be cleared.
* This is why the strings are called immutable.

**6. String Pool in Java**

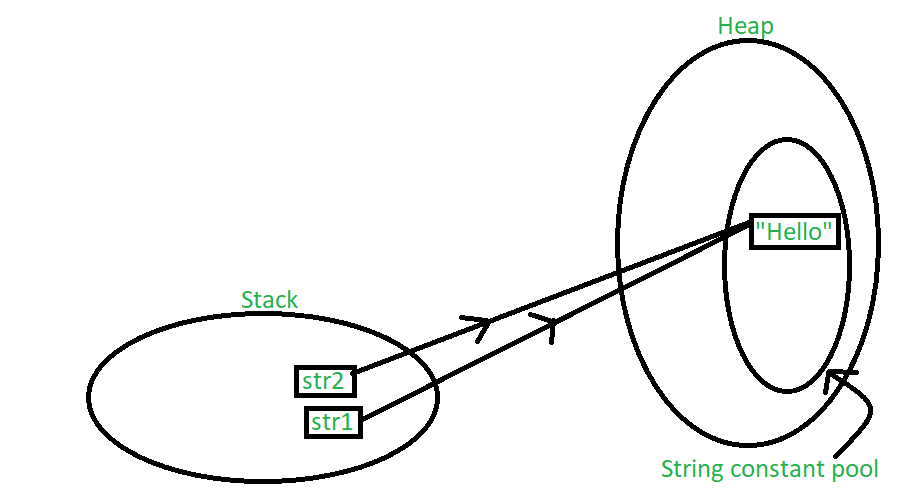
* + Strings in java are immutable. This immutability is achieved through the use of a specia string constant pool in the heap.
  + A string constant pool is a separate place in the heap memory where the values of all the strings which are defined in the program are stored.
  + When we declare a string , an object of the type string is created in the stack, while an instance with the value of string is created in the heap.
  + On the standard assignment of a value to a string variable ( string literal), the variable is allocated in stack, while the value is stored in the heap in the string constant pool.
  + Ex: String str1 = “Hello”;. The object is created in stack and the vale is created and stored in the heap, in constant pool area.



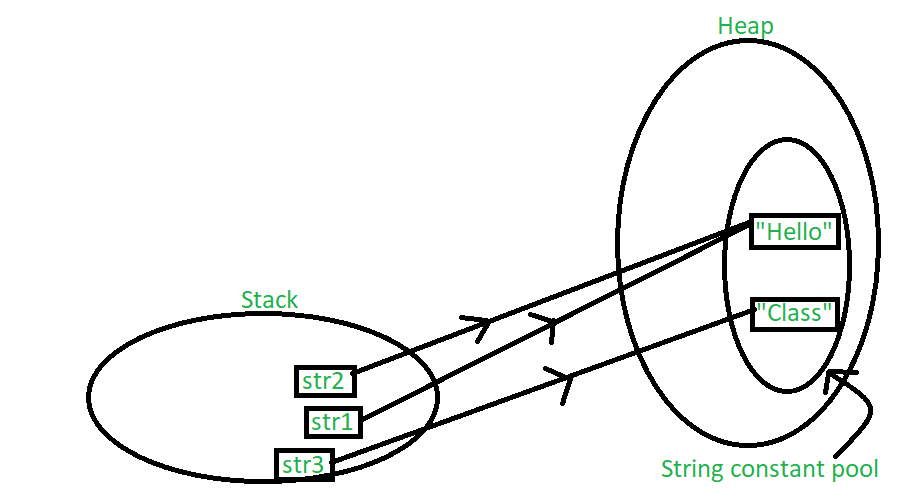
* + if the we declare multiple values with the same string literal:

ex: String str1 = "Hello";

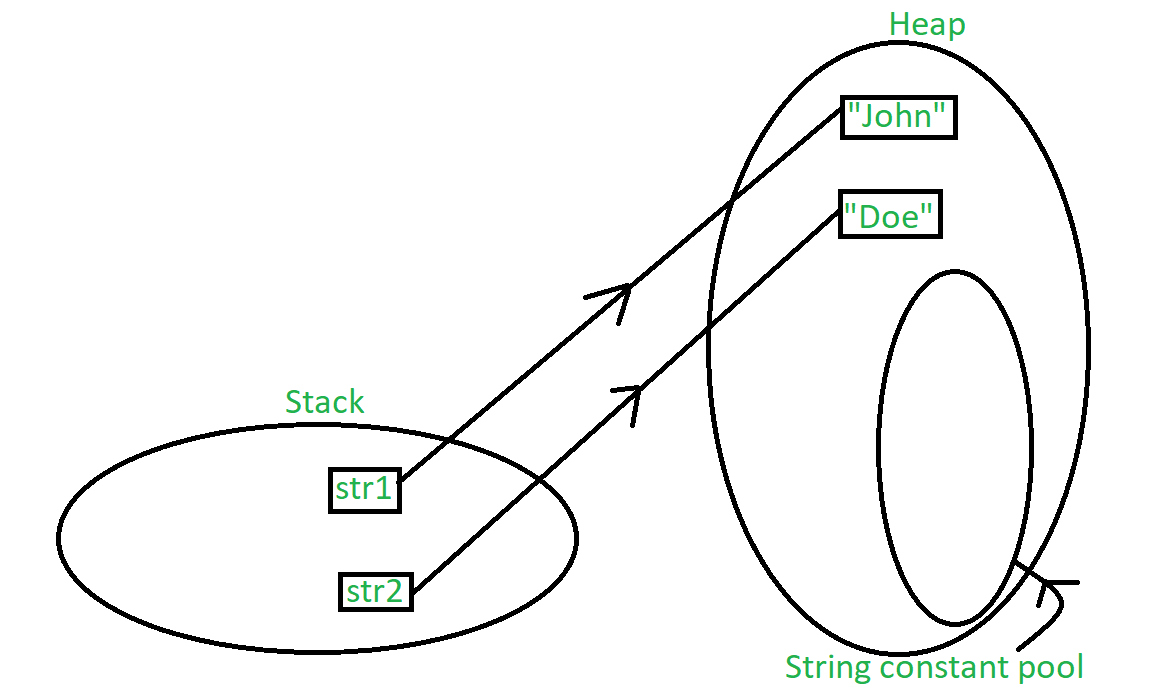
String str2 = "Hello";



* + In this case, both the string objects get created in the stack, but another instance of the value “Hello” is not created in the heap. Instead, the previous instance of “Hello” is re-used. The *string constant pool* is a small cache that resides within the heap. Java stores all the *values* inside the string constant pool on direct allocation. This way, if a similar value needs to be accessed again, a new string object created in the stack can reference it directly with the help of a pointer.
  + When a string object is assigned to a different value, the new value will be registered in the string constant pool as a separate instance.



* + One way to skip this memory allocation is to use the [**new keyword**](https://www.geeksforgeeks.org/new-operator-java/) while creating a new string object. The ‘new’ keyword forces a new instance to always be created regardless of whether the same value was used previously or not. Using ‘new’ forces the instance to be created in the heap outside the string constant pool
    - String str1 = new String("John");
    - String str2 = new String("Doe");



* + This is why both of the program below will produce different result.
* Program I

public static void main(String[] args)

    {

        String s1 = "abc";

        String s2 = "abc";

        // Note that this == compares whether

        // s1 and s2 refer to same object or not

        if (s1 == s2)

           System.out.println("Yes"); // YES

        else

           System.out.println("No");

    }

* Program II

 public static void main(String[] args)

    {

        String s1 = new String("abc");

        String s2 = new String("abc");

        // Note that this == compares whether

        // s1 and s2 refer to same object or not

        if (s1 == s2)

           System.out.println("Yes");

        else

           System.out.println("No"); // NO

    }

**7. Write Java code to read number from user and print multiplication table of that number.**

package com.company;  
  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Scanner scn = new Scanner(System.*in*);  
 System.*out*.println("Enter a number to get it's multiplication table:");  
 int number = scn.nextInt();  
 int i=0;  
 scn.nextLine();  
 while(i < 11){  
 System.*out*.println(number+" X "+ i+" = "+number\*i);  
 i++;  
 }  
 scn.close();  
  
 }  
  
  
}

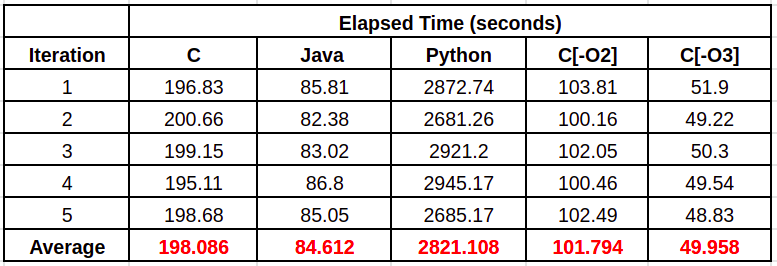
**8. Java vs Python which is faster.**

In terms of speed, Java is **faster** than Python as it is a compiled language. It takes less time to execute a code.

Python is an interpreted language and it determines the type of data at run time which makes it **slower** comparatively.

**The Experiment**

Matrix multiplication using all three languages. The matrices are of size 2048 x 2048 (i.e. 8,589,934,592 multiplication and addition operations each) and populate them with random values between 0.0 and 1.0 (the impact of using random values rather than using the exact same matrices for all three languages is negligible). Run each experiment five times and calculated the average running time.



**Python is 33.34 times** **slower** than Java.