String Literal :
---
* String is a predefined class available in java.lang package.

* String is a collection of character which is enclosed with double quotes.

* String is also collection of alpha-numeric character so we can hold the combination of character, number as well as special symbol.

How many ways we can create String object in java :
---
* There are 3 ways to create String object in java :

1) By using String Literal
```
String str = "Hello World";
```

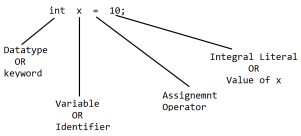
2) By using new keyword
```
String str1 = new String("India");
```

3) By using character array
```
char []ch = {'H','E','L','L','0'};
```

---

Punctuators :
---
* It is also known as separators.
* It is used to inform the compiler how the things will be grouped together.

Example : , ; () {} [] ...




---

What is a local variable ?
---
* If we declare a variable inside the **method body OR block OR Constructor** then it is called local variable.

Example :
---
```
public void acceptData(int x)
{
  int y = 200;
}
```
Parameter Variable

Local variable

* The scope of the local variable is within the same method body, block OR constructor that means we **can't use local variable** outside of method, body or block.

* A local variable must be initialized by the developer before use.

```
public class LocalVariable
{
    public static void main(String[] args)
    {
        int x;
        x = 100;
        System.out.println("x value is  :"+x);
    }
}
```

* We can't apply any kind of modifier on local variable except final.

```
public class LocalVariable
{
    public static void main(String[] args)
    {
        final int x = 10;
        System.out.println("x value is  :"+x);
    }
}
```

* A local variable must be pre-declared before use.

```
public class LocalVariable
{
    public static void main(String[] args)
    {
        System.out.println(x);
        int x = 100;
    }
}
```

* All the local variables are executed as a part of method body and all the methods are executed in special memory called Stack Memory so we can say all the local variables are also executed in the stack memory.

---

Why we can't use local variables outside of the method OR body OR Constructor ?
---
* All the local variables are executed as a part of Stack Memory and Stack memory works on LIFO (Last In First Out) basis.

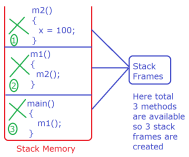
* In this Stack memory whenever we call any method then for that particular method a separate stack frame will be created inside the Stack Memory only as shown in the program below :

```
public class LocalVariableScope
{
    public static void main(String[] args)
    {
        System.out.println("Main method Started!!!");
        m1();
        System.out.println("Main method ended!!!");
    }

    public static void m1()
    {
        System.out.println("M1 method Started!!!");
        m2();
        System.out.println("M1 method ended!!!");
    }

    public static void m2()
    {
        int x = 100;
        System.out.println(x);
    }

}
```





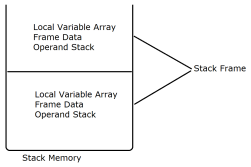


Here total 3 methods are available so 3 stack frames are created

* This Stack frame is responsible to hold local variable data, Once the method execution is completed then that particular stack frame will be automatically deleted from the corresponding stack memory hence all the local variables are also deleted hence we can't reuse outside of the Frame.

* Every Stack frame internally contains 3 parts :

1) Local Variable Array
2) Frame Data
3) Operand Stack




---

Limitation of Commnd line Argument ?
---
* By using Command Line Argument, we can pass value at runtime but the Limitation of Command line Argument is, we can't pass user friendly input message to the end user

* If we want to provide user - friendly input message then we should go with another level of concept

How to read the data from the Client with user-friendly message :
---
1) java.io.DataInputStream class
2) java.io.BufferedReader class
3) System.in.read()
4) java.io.Console class
5) **java.util.Scanner**