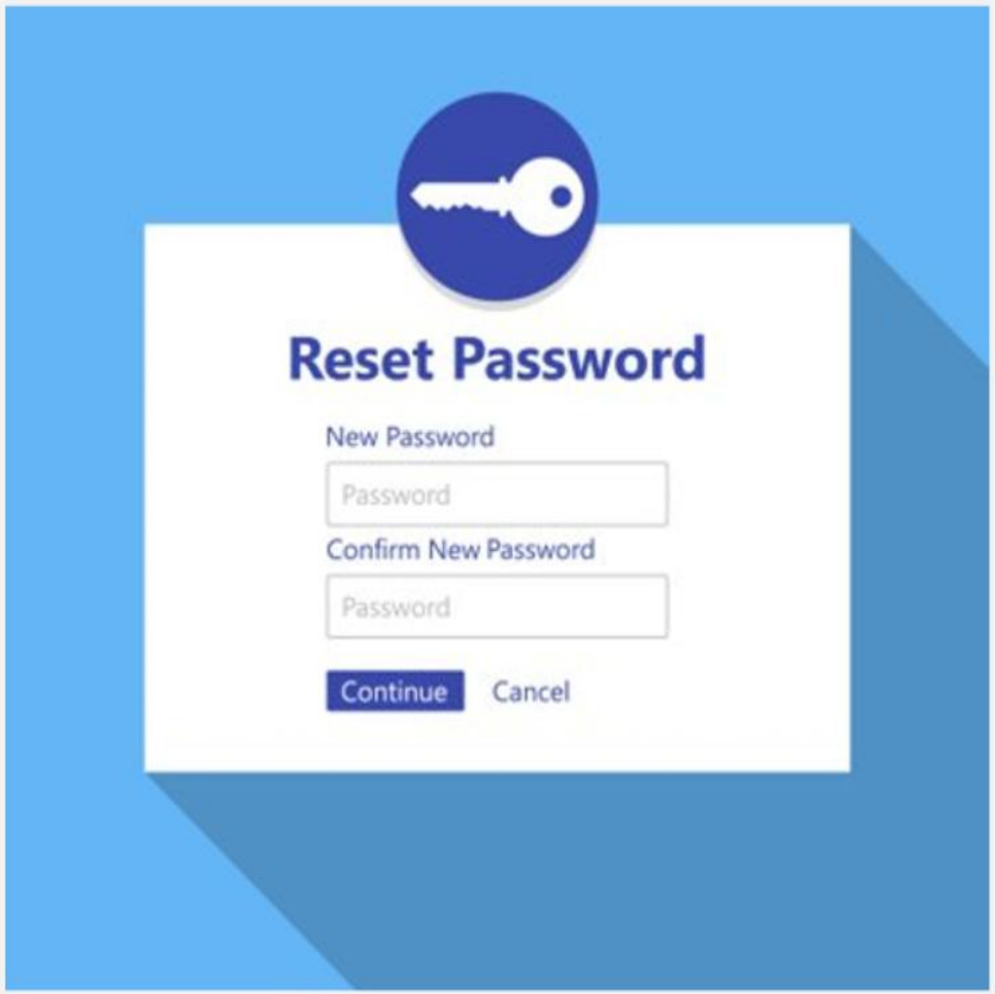


Think and Tell

- What type of data is used to change a password?
- What do you think happens when a string is entered as a password?

Employee Identity Document

- What is common between all the employee Ids?
- How can you prefix employee Ids with “EMP”?

	A	B	C
1	**Employee ID**	**Location**	**Name**
2	E001	Chicago, IL	Steven Adams
3	E002	Chicago, IL	Aaron Love
4	E003	Chicago, IL	Gary Buxton
5	E004	Chicago, IL	Michael Stafford
6	E005	Chicago, IL	Patricia Spain
7	E006	Chicago, IL	Dharma Milner
8	E007	Chicago, IL	Thanh Nguyen
9	E008	Chicago, IL	Myron Duarte
10	E009	Chicago, IL	Cheryl Perez
11	E010	Oakbrook, IL	Calvin Russell
12	E011	Oakbrook, IL	Virginia Thiesse
13	E012	Oakbrook, IL	Wenquao Li
14	E013	Oakbrook, IL	Jo Ann Cortelloni
15	W001	Chicago, IL	Chicago Workgroup
16	W002	Oakbrook, IL	Oakbrook Workgroup

Finding the Country Code

Imagine that you have been assigned the task of finding a particular country code from a long list of countries.

- How can you extract this data from the long list?

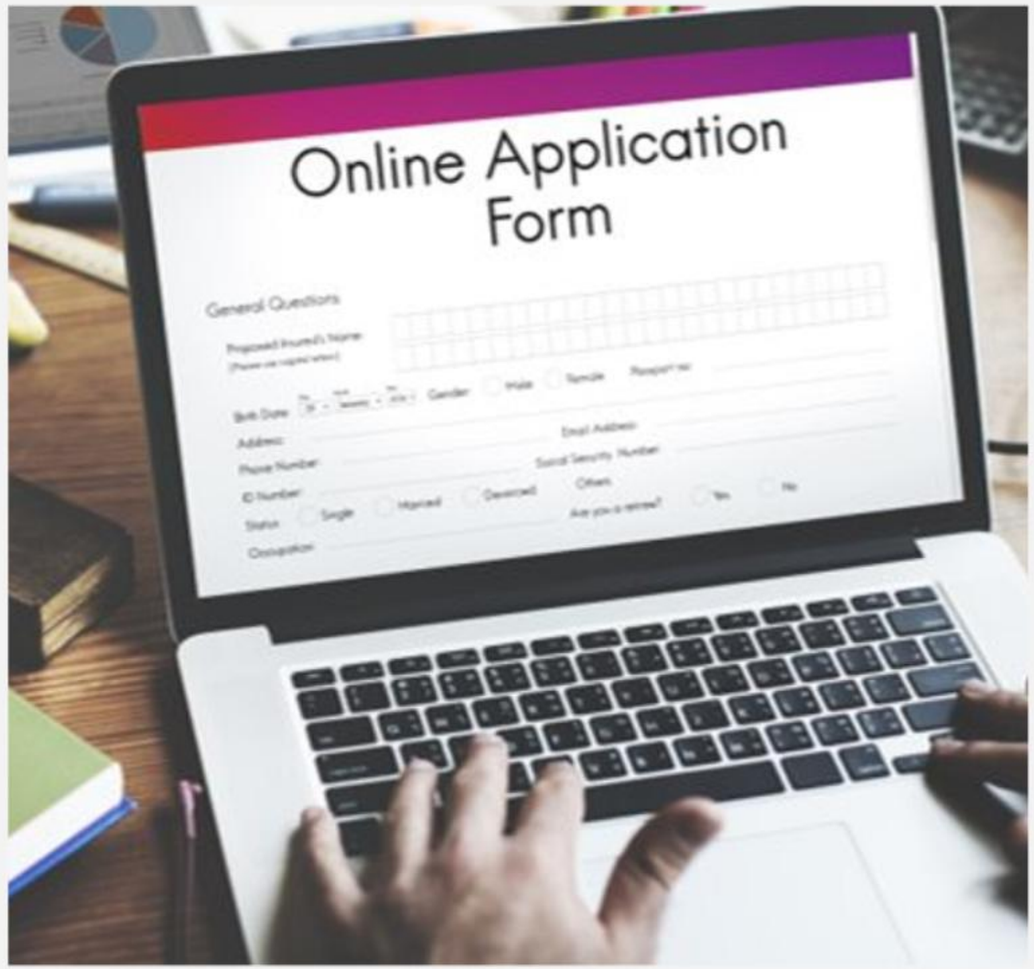
France	+33 6 34 55 66 77
USA	+44 20 8366 1177
Algeria	+213 634 55 66 77
Swiss	+41 44 668 18 00
Italy	+39 365 478 9145
Portugal	+351 22 040 4000

Registration Form

A registration form usually requires the following data:

- Name in sentence case
- DOB in the correct format (DD/DMM/YYYY)
- Phone number
- Gender
- Exact address details with house and street number, city, state, country and ZIP Code

How do you handle the string data entered by a user in this form?



Java String Manipulations

Learning Objective

- Explore String Class
- Describe how Strings are Immutable
- Compare String with (==) equality operator and with equals method
- Manipulate String Objects

String Class

What Is a String?

- A collection or a sequence of characters is known as a string.
- String is a predefined class of Java.
- Once a String is declared and initialized it cannot be changed. Stings are Immutable.
- Characters of a String are numbered with numerical index.
- String name = "John Miller"

Characters	J	o	h	n		M	i	l	l	e	r
Index	0	1	2	3	4	5	6	7	8	9	10

- First Character is at 0^{th} index.
- Last Character is at String length -1 index.

Declare and Initialize a String

- A String is a non-primitive data type which is created as an Object.
- A String object can be created in two different ways:
  - Declared as a new object using the String class.
    - `String name = new String ("John");`
    - The above String is equivalent to:

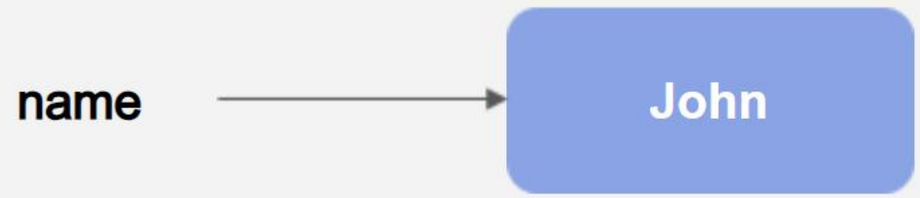
```
char name [] = {'J','o','h','n'};
```

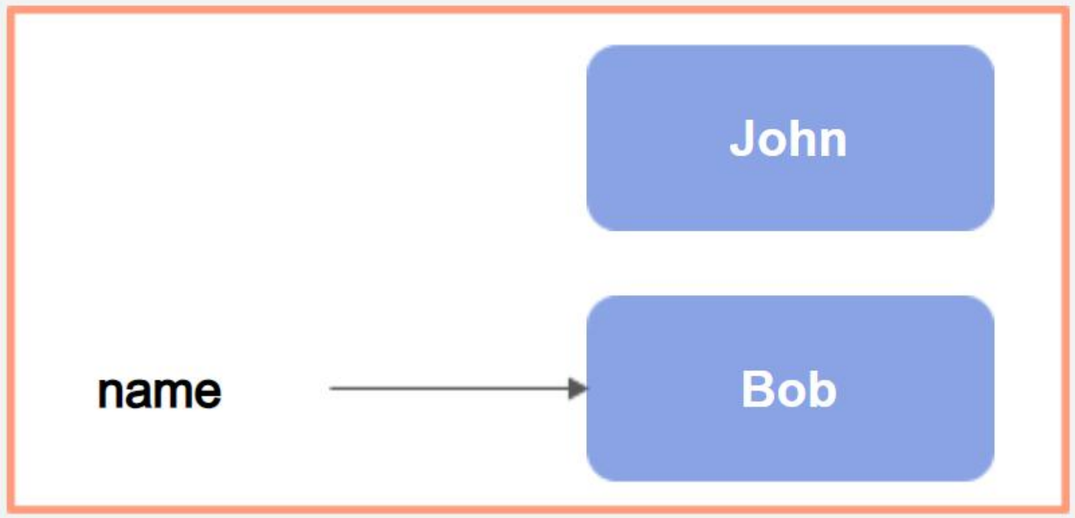
  - Declared as a literal like primitive datatypes without the new keyword.
    - `String name = "John"`
- *Note: Object will be discussed later in the course.*

Strings Are Immutable

Immutability of Strings

- In Java, String is an immutable class. This means that once a string object is created, its value cannot be changed.
- Let's create a variable of type String called `name` and assign a value `John` to it.
- `String name = "John"`
- The `name` variable will be pointing to `John`
- Assign another value to variable `name` called `Bob`.
- `String name = "Bob"`
- The `name` variable will point to Bob but will not overwrite `John`.
- The value `John` object is still present in the memory without any variable referring to it.
- Hence, String is called immutable classes.

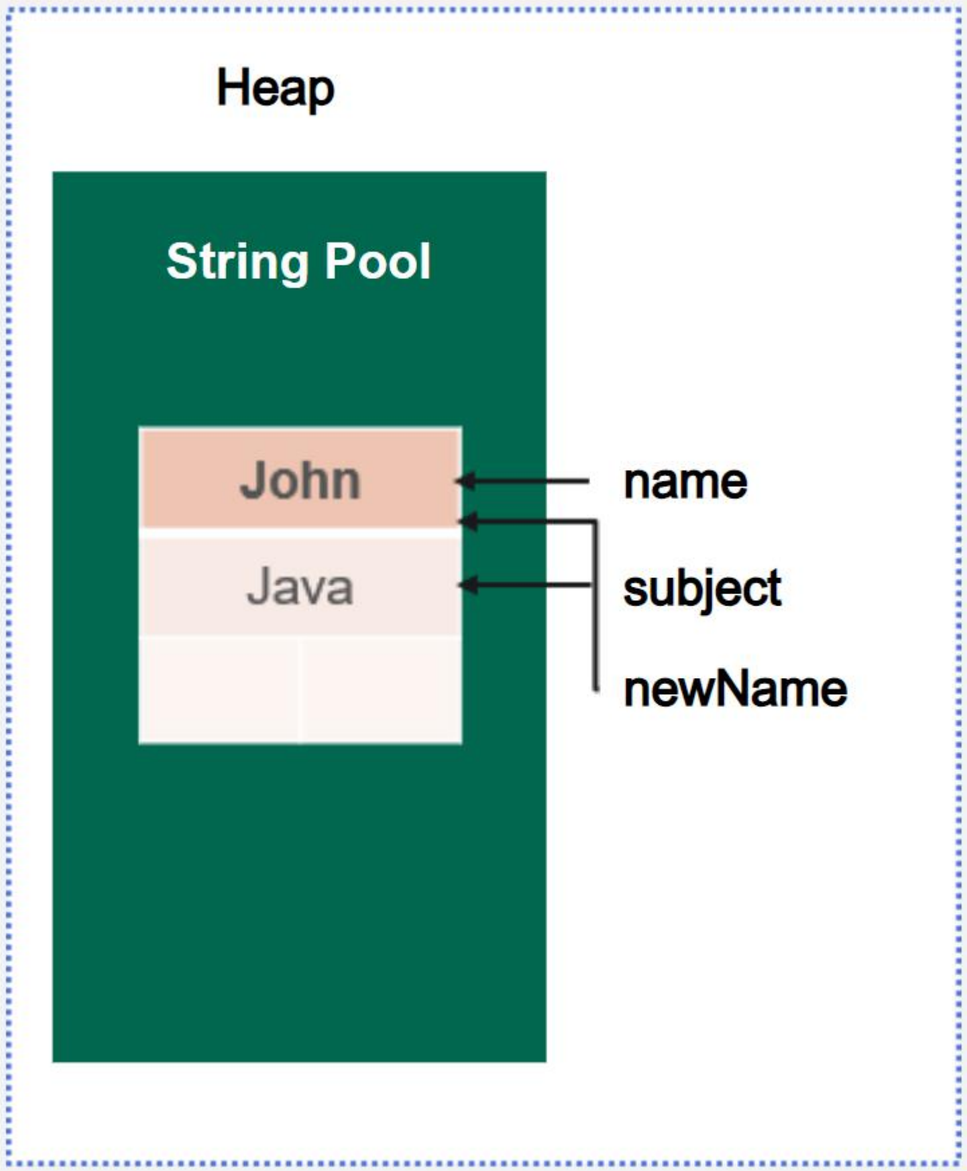




What Is a String Pool?

- A JVM has two memory spaces called as stack and heap.
- String pool is a storage area in memory inside the heap where string literals are stored.
- It is also called as the String Constant Pool.
- It is empty by default and maintained by the Java String class.
- Whenever a String object is created it occupies memory.
- Creating several String objects may increase cost and memory usage resulting to performance issues.
- So, to decrease the memory load and increase the performance, the JVM stores all the Strings created by Java programs in the String Constant Pool.

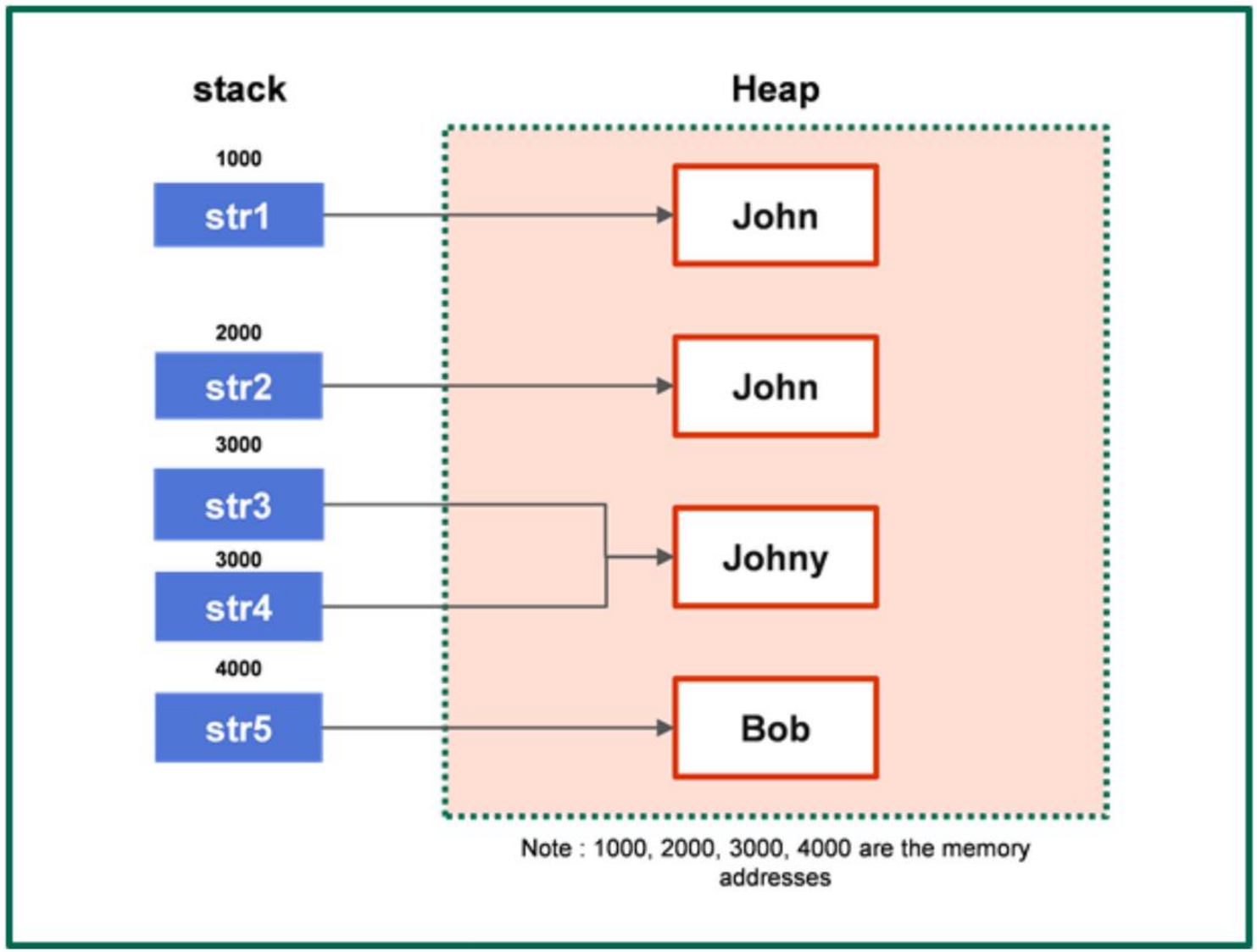
- *Note – More about JVM memory will be discussed in upcoming courses*



Literals in String Pool

- `String name = "John" -`
- `String subject = "Java"`
- `String newName = "John"`
- When the above literals are created, JVM first checks the literals in the String pool.
- If a literal is present in the pool, it returns a reference of it.
- If the literal is not present a new String object is created in the pool.
- Here, as John is already present in the String pool, a new object is not getting created but `newname` is pointing at the same object.

String in Memory



- Stack is used to store order of method execution and local variables and object reference.
- Heap stores Objects that are created using new keyword like String object, Array object etc.
- String str1= new String("John");
- String str2 = new String("John");
- String str3 = "Johny";
- String str4 = "Johny";
- String str5 = new String("Bob");
- Here str3 and str4 point towards the same object in heap and have same address.

Comparing Strings

String Equality Using (==) Operator

```java
public static void main(String[] args) {
  String name1= new String( original: "john");
  String name2=new String( original: "john");
  if(name1 == name2)
  {
      System.out.println("Both the name are same");
  }
  else{
      System.out.println("Name are not same");
  }

}
```

- The equality operator or "==" compares two objects based on memory reference

- Both the Objects have the same value, but the memory address are different as they are getting created using a new keyword.

String Equality Using (==) Operator (cont'd)

```
public static void main(String[] args) {
    String name = "John";
    String name1 = "John";
     if(name == name1){
        System.out.println("Both the name are same");
    }
     else{
        System.out.println("Name are not same");
     }
}
```

- The output will be "Both the name are same" .
- The equality operator or "==" compares two objects based on memory reference.
- As objects are getting stored in the String pool, both the objects are pointing at the same memory location.

String Equality Using Equals Method

- `public boolean equals(Object anObject)`
  this method checks the values of the String.
- To check the equality of two String the equals method should be used.
- The equals method compares the String value of the two string objects and returns a boolean value, true if both the string values are equal, false if both are different.
- Since both the Strings are having same value "John“ hence, output is "Names are equal".
- Even the Strings in the String Pool will return true if both are having same value.

```
public static void main(String[] args) {
    String name = new String( original: "John");
    String name1 = new String( original: "John");
    if(name.equals(name1)){
        System.out.println("Names are equal");
    }
    else{
        System.out.println("Names are not equal");
    }
}
```

```
String name = "John";
String name1 = "John";
if(name.equals(name1)){
    System.out.println("Names are equal");
}
else{
    System.out.println("Names are not equal");
}
```

```
Names are equal
```

Quick Check

Predict the output of the following code:

```
String name = "John";
String name1 = "John";
String name2 = new String( original: "John");
if (name.equals(name2)) {
    System.out.println("They are equal");
} else {
    System.out.println("They are not equal");
}
}
```

1. They are equal
2. They are not equal.
3. Run time error
4. They are equal,
   They are not equal

Quick Check: Solution

Predict the output of the following code:

```
String name = "John";
String name1 = "John";
String name2 = new String( original: "John");
if (name.equals(name2)) {
    System.out.println("They are equal");
} else {
    System.out.println("They are not equal");
}
}
```

1. They are equal
2. They are not equal.
3. Run time error
4. They are equal
   They are not equal

Check Equality of String

Write a program to check the equality of two String using (==) operator and equals method.
Check equality by creating String with new keyword and String as literal.

Click on link for the solution.
The workbench must be used for the demonstration. Execute the test cases provided in the test folder

DEMO

Manipulating Strings

- How do we perform different operations on strings?
- What happens when users enter their username and password in different cases?
- Can String methods change a case to uppercase or lowercase?
- If you want to know the index position of some characters in String how to do that?

Manipulating String Object

Finding the Length of a String

- `public int length()` **method is used to determine the length of a String.**
- **The methods counts the number of characters in the string variable and returns the** `int` **count.**

```
String subject = "Java Programming Language";
System.out.println("Length of String : " + subject.length());
```

```
Length of String : 25
```

Changing String Case

```
public void changeCase(){
    String message = "Java PrograMMing";
     System.out.println(message.toLowerCase());
     System.out.println(message.toUpperCase());
}
```

```
java programming
JAVA PROGRAMMING
```

- To change the case of a string, we use the toLowerCase() and toUpper Case() methods of the String class.
- toLowerCase() - Converts all of the characters in this String to lower case.
- toUpperCase() – Converts all of the characters in this String to uppercase.
- These methods return a new String and cannot modify the existing string as they are immutable.

Joining Strings

- **To join one or multiple Strings `concat` method is used.**
- `public String concat(String str)` – **Concatenates the specified string to the end of this string.**

```
public void joinString(String fName, String lName, String msg){
        msg = "Hello";
        fName = "James";
        lName = "Smith";
        String name = msg.concat( str: " ").concat(fName).concat( str: " ").concat(lName);
        System.out.println(name);
}
```

```
Hello James Smith
```

String Manipulation

Write a program that accepts the username and password and performs the following tasks:

Task1: Create a password of less than 15 characters.
Task2: Ensure that the username is “James” and the password entered is "password@123“.
Task3: Display the welcome message with the username in upper case.

Click on link for the solution.
The workbench must be used for the demonstration.

DEMO

Finding the Position of Characters in a String

- `public char charAt(int index)` - **This method checks the character at the specified index.**

```java
public void characterAt(){
     String string = "Java Programming";
     char char1 = string.charAt(0);
     char char2 = string.charAt(5);
    System.out.println("Character at first position : " + char1);
    System.out.println("Character at sixth position : " + char2);

}
```

```
Character at first position : J
Character at sixth position : P
```

Substring of Strings

- `public String substring(int beginIndex)` - **This method Returns a string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.**

```
public void subStringDemo(){
    String msg = "Java Programming";
    String newString = msg.substring( beginIndex: 3);
    System.out.println(newString);
    String newString1 = msg.substring(3,6);
    System.out.println(newString1);

}
```

This substring returns a new String that starts from the 3rd index position till the end.

This substring method returns a new String that starts from the 3rd index position till the 6th index position.

```
New String :a Programming
New String :a P
```

Converting a String to Character Array

- `public char[] toCharArray()` **- This method converts this string to a new character array.**

```
public void stringToChar(String message){
    message = "Java Programming";
    char result[] = message.toCharArray();
        for (int i=0;i<result.length;i++){
            System.out.println(result[i]);
        }
}
```

- **After getting characters in the result variable use `for` loop to iterate all the characters.**

Quick Check

Predict the output of the following code:

```
public static void main(String[] args) {
    String str1 = new String( original: "Java Programming");
    str1.concat( str: "language");
    System.out.println(str1);
}
```

1. Java Programming language
2. Java Programming
3. Language
4. Compile time error

Quick Check: Solution

Predict the output of the following code:

```
public static void main(String[] args) {
    String str1 = new String( original: "Java Programming");
    str1.concat( str: "language");
    System.out.println(str1);
}
```

1. Java Programming language
2. Java Programming
3. Language
4. Compile time error

String Class Methods

String method	Description
public char charAt(int index)	Returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0.
public boolean startsWith(String prefix)	Tests if this string starts with the specified prefix.
public boolean endsWith(String suffix)	Tests if this string ends with the specified suffix.
public int indexOf(String str)	Returns the index within this string of the first occurrence of the specified substring.
public String replace(char oldChar, char newChar)	Returns a string resulting from replacing all occurrences of oldChar in this string with the newChar.
public String trim()	Returns a string whose value is this string, with any leading and trailing whitespace removed.
public String toString()	This object (which is already a string!) is itself returned.

String Methods

Write a Java program that will create an array of Strings and will perform the following task.

- Find the length of all the String in an array.
- Make all the String in the array in Uppercase.
- Find all the String that startwith "J".
- Count all the Vowels of all the Strings present in the array.

Click on link for the solution.
The workbench must be used for the demonstration. Execute the test cases provided in the test folder.

DEMO