

## Lesson 02 Demo 06

### Using String Methods

**Objective:** To explore methods available with string data type

**Tools required:** Eclipse IDE

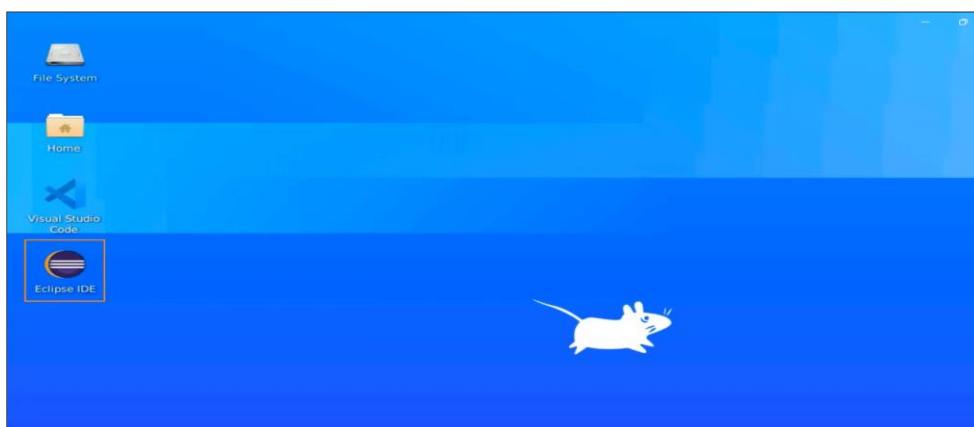
**Prerequisites:** None

Steps to be followed:

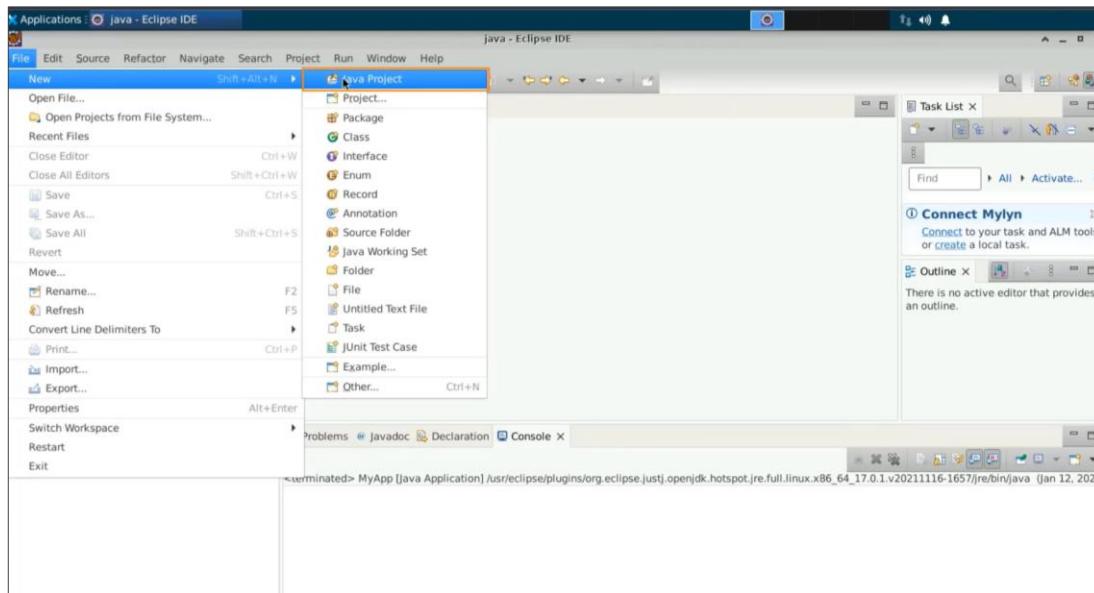
1. Write a string and calculate the number of characters in the string
2. Get the index of characters
3. Use the uppercase method to convert the string to uppercase
4. Implement string slicing and extracting sub-strings
5. Use the trim function on strings
6. Implement the concat method for concatenation of strings
7. Validate the strings using the methods ends, contains ends with, and starts with
8. Implement an algorithm to know the number of times a character appears
9. Implement String comparison and intern strings
10. Use the **method.equals** method

#### Step 1: Write a string and calculate the number of characters in the string

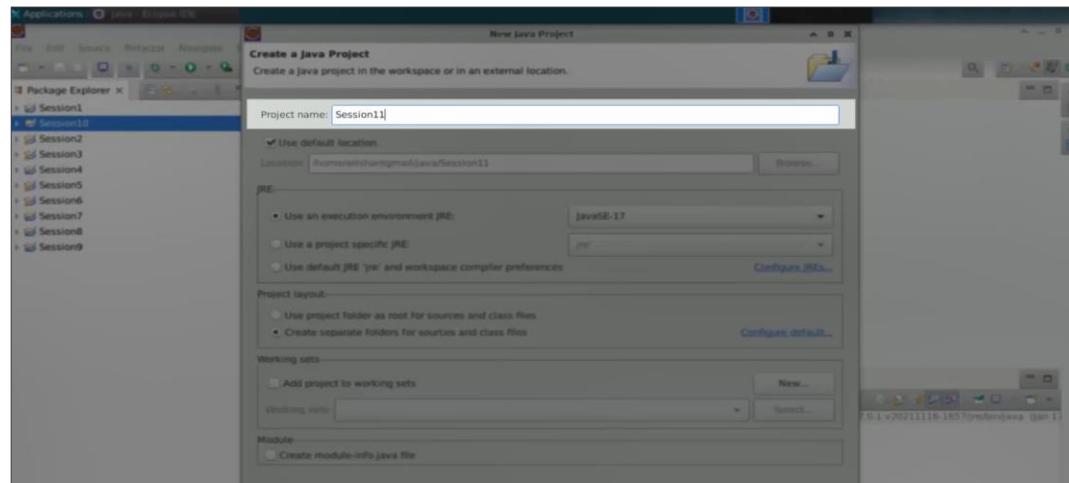
##### 1.1 Open the **Eclipse IDE**



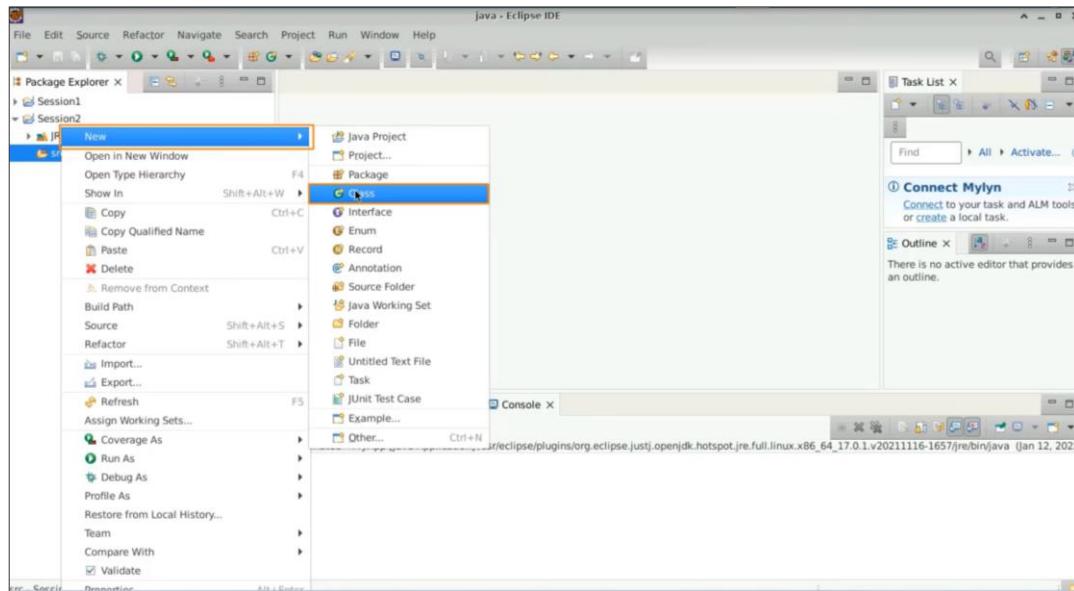
## 1.2. Select File, then New, and then Java project



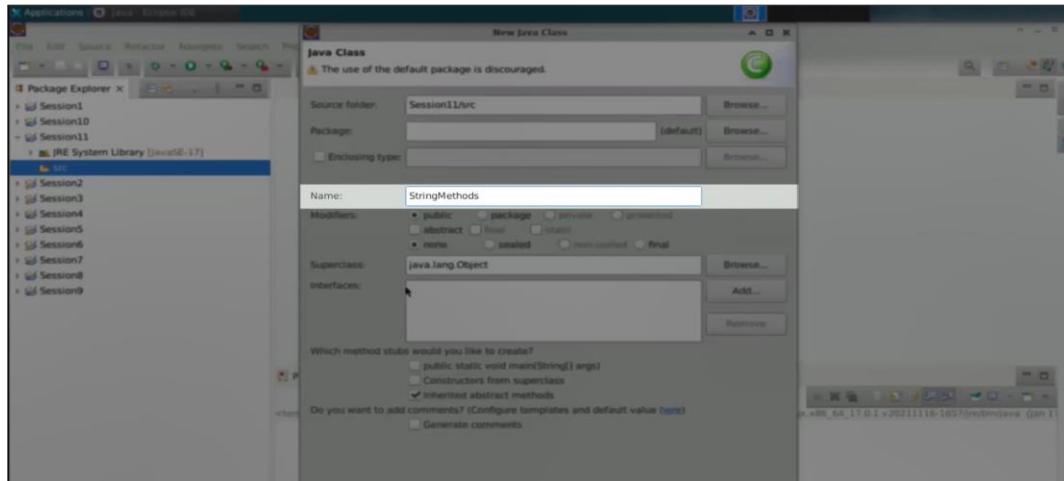
## 1.3 Name the project “Session11”, uncheck “Create a module info dot Java file”, and press Finish



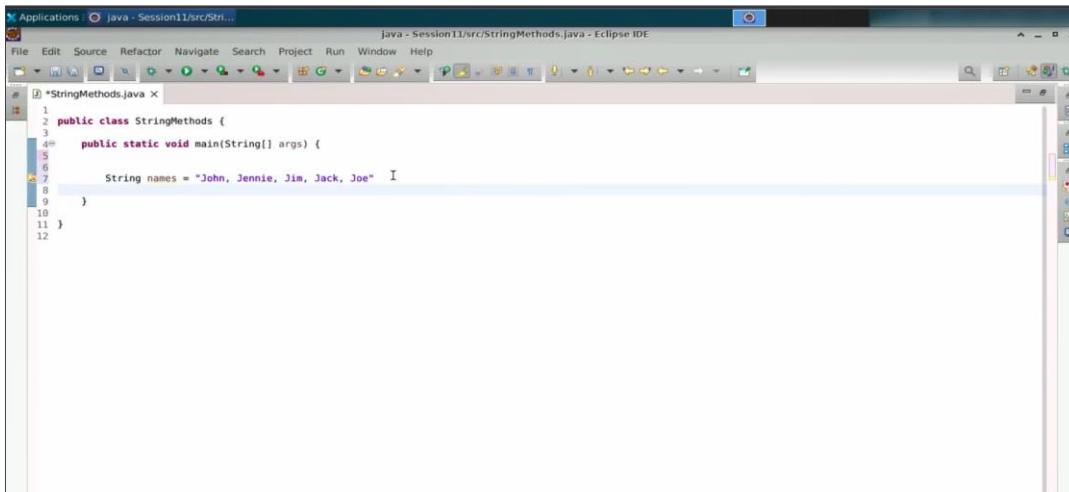
#### 1.4 With a **Session11** on the src, do a right-click and create a new class



#### 1.5 Name this class as an **StringMethods**, then select the **main method**, and then select **finish**



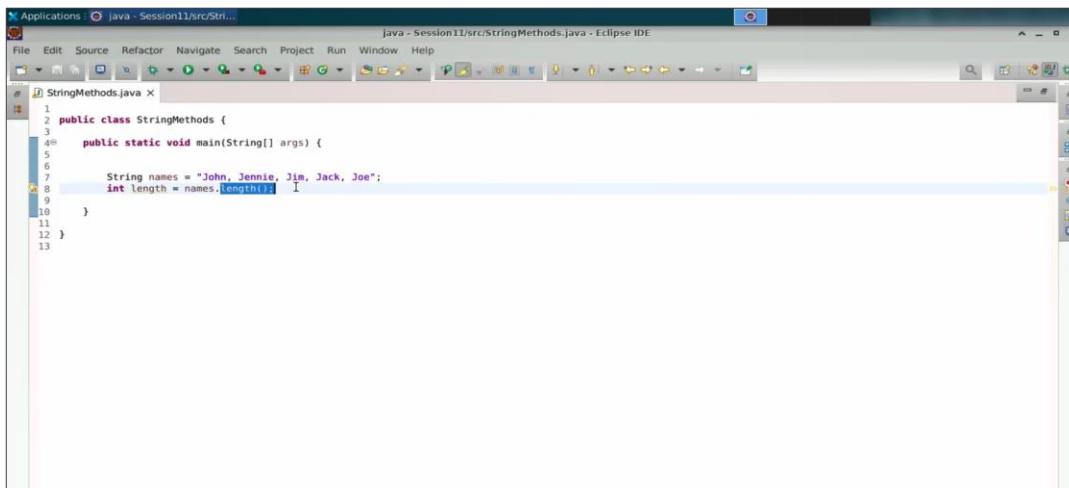
1.6 Let us write one string called names and these names are John, Jenny, Jim, Jack, and Joe. This is altogether a single string



The screenshot shows the Eclipse IDE interface with a Java file named "StringMethods.java" open. The code defines a class "StringMethods" with a main method that initializes a string variable "names" with the value "John, Jenny, Jim, Jack, Joe".

```
1 public class StringMethods {  
2     public static void main(String[] args) {  
3         String names = "John, Jenny, Jim, Jack, Joe"  
4     }  
5 }  
6  
7  
8  
9  
10  
11 }  
12
```

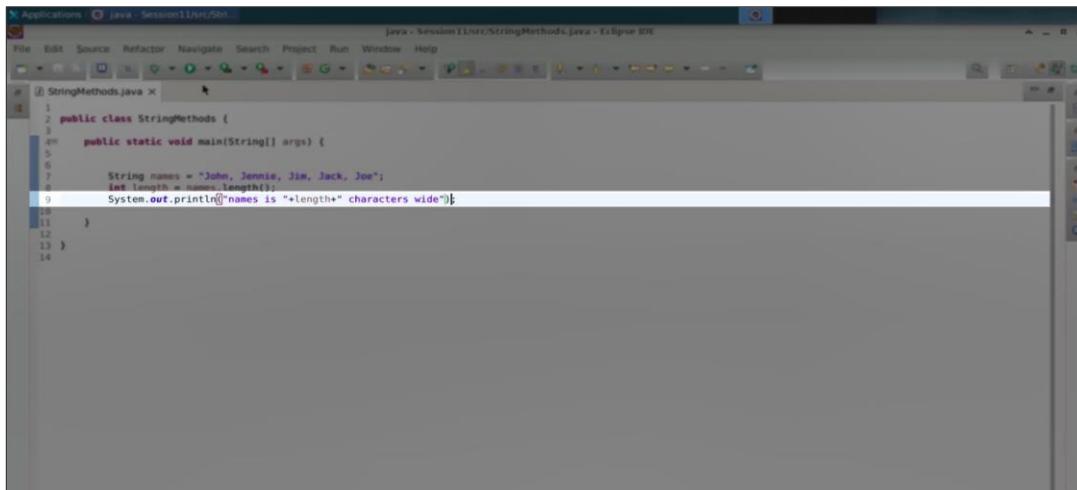
1.7 Add int length = names.length(); to execute a function called length. The length method is available as a built-in method



The screenshot shows the Eclipse IDE interface with the same Java file "StringMethods.java" open. In addition to the previous code, it includes an assignment statement where an integer variable "length" is assigned the value of the "length" method call on the "names" string variable.

```
1 public class StringMethods {  
2     public static void main(String[] args) {  
3         String names = "John, Jenny, Jim, Jack, Joe";  
4         int length = names.length();  
5     }  
6 }  
7  
8  
9  
10  
11 }  
12  
13
```

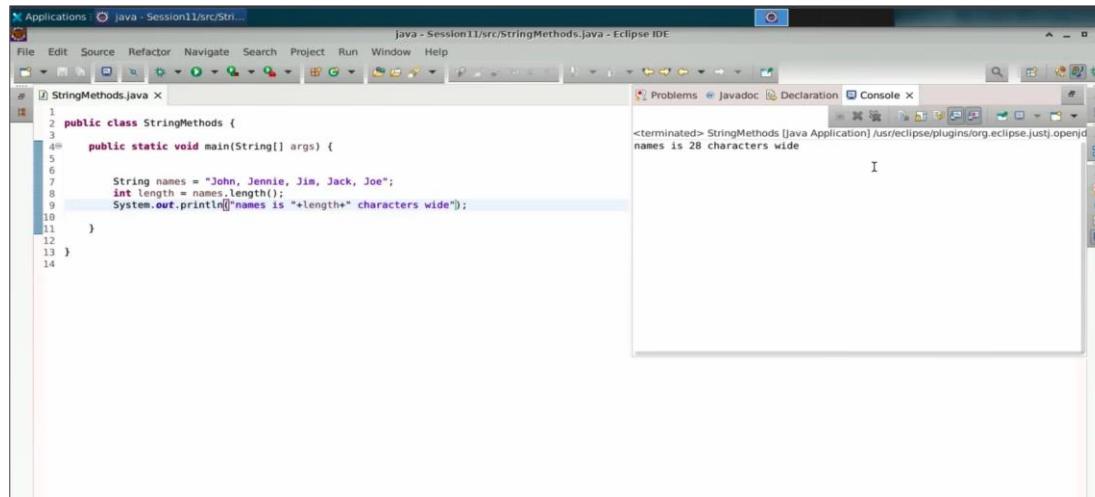
1.8 Add `System.out.println("Names are " + length + " characters wide.");`



A screenshot of the Eclipse IDE interface. The title bar says "Applications java - Session11/src/StringMethods.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The main editor window shows the following Java code:

```
StringMethods.java
1 public class StringMethods {
2     public static void main(String[] args) {
3         String names = "John, Jennie, Jim, Jack, Joe";
4         int length = names.length();
5         System.out.println("names is "+length+" characters wide");
6     }
7 }
8
9
10
11
12
13
14
```

1.9 Run the code. It says that name is 28 characters wide

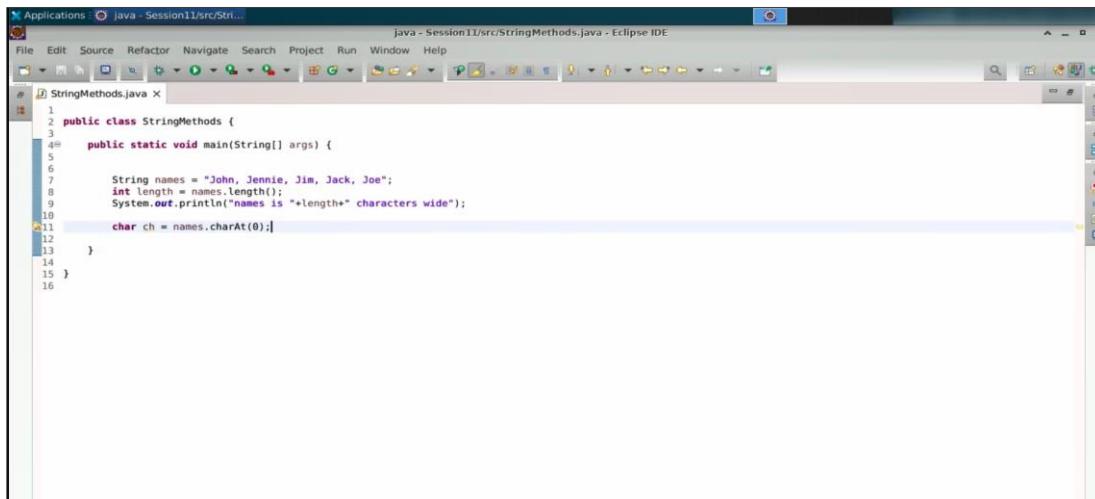


A screenshot of the Eclipse IDE interface. The title bar says "Applications java - Session11/src/StringMethods.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The main editor window shows the same Java code as the previous screenshot. To the right, the "Console" tab is selected in the view switcher, showing the output of the run:

```
<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdt
names is 28 characters wide
```

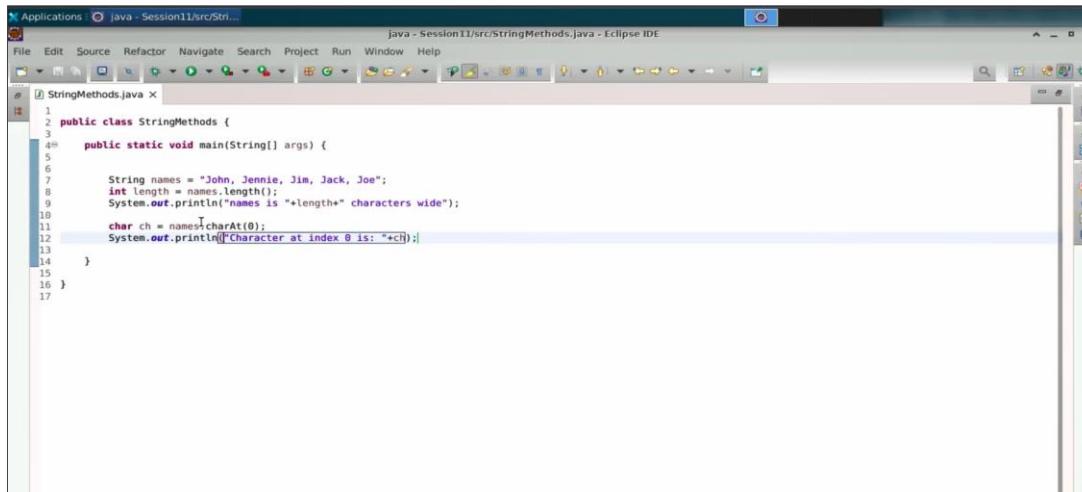
## Step 2: Get the index of characters

2.1 A char called Ch is going to be named dot character at the index number 0



```
StringMethods.java X
1 public class StringMethods {
2     public static void main(String[] args) {
3
4         String names = "John, Jennie, Jim, Jack, Joe";
5         int length = names.length();
6         System.out.println("names is "+length+" characters wide");
7
8         char ch = names.charAt(0);
9
10    }
11
12 }
13
14
15 }
16 }
```

2.2 Now the character at index number 0, is very much clear, let us say character at Index zero is plus Ch. When you say char, this is a method with the help of which you can get the character from a string



```
StringMethods.java X
1 public class StringMethods {
2     public static void main(String[] args) {
3
4         String names = "John, Jennie, Jim, Jack, Joe";
5         int length = names.length();
6         System.out.println("names is "+length+" characters wide");
7
8         char ch = names.charAt(0);
9         System.out.println("Character at index 0 is: "+ch);
10
11    }
12
13 }
14
15 }
16 }
```

2.3 The 0 character is this upper-case J and when you run the program here you get to see it gives the value as J

```

StringMethods.java
1 public class StringMethods {
2     public static void main(String[] args) {
3
4         String names = "John, Jennie, Jim, Jack, Joe";
5         int length = names.length();
6         System.out.println("names is "+length+" characters wide");
7
8         char ch = names.charAt(0);
9         System.out.println("Character at index 0 is: "+ch);
10    }
11
12 }
13
14
15
16
17

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.vml/names is 28 characters wide  
Character at index 0 is: J

2.4 Let us also say ch is `names.charAt(length - 1)`. Then add `System.out.println("Character at length - 1 is " + ch);`

```

StringMethods.java
1 public class StringMethods {
2     public static void main(String[] args) {
3
4         String names = "John, Jennie, Jim, Jack, Joe";
5         int length = names.length();
6         System.out.println("names is "+length+" characters wide");
7
8         char ch = names.charAt(0);
9         System.out.println("Character at index 0 is: "+ch);
10        ch = names.charAt(length-1);
11        System.out.println("Character at length-1 is: "+ch);
12    }
13
14 }
15
16
17
18
19
20

```

2.5 When you run the code, you will see that the character at **length - 1** is e

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays `StringMethods.java` with the following content:

```

1 public class StringMethods {
2     public static void main(String[] args) {
3
4         String names = "John, Jennie, Jim, Jack, Joe";
5         int length = names.length();
6         System.out.println("names is "+length+" characters wide");
7
8         char ch = names.charAt(0);
9         System.out.println("Character at index 0 is: "+ch);
10
11        ch = names.charAt(length-1);
12        System.out.println("Character at length-1 is: "+ch);
13    }
14
15 }

```

On the right, the **Console** view shows the output of the program:

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/openFiles/StringMethods.java:12:20: characters wide
names is 20 characters wide
Character at index 0 is: J
Character at length-1 is: e

```

2.6 Type **idx = name.indexOf('o');**. Let us see the index of a character called 'o'. The character 'o' is available at the first index as well as the second-to-last index. The **indexOf** method will give you the index of the character from the beginning. So, when you type **indexOf('o')**, it will give you the value 1

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays `StringMethods.java` with the following content:

```

1 public class StringMethods {
2     public static void main(String[] args) {
3
4         String names = "John, Jennie, Jim, Jack, Joe";
5         int length = names.length();
6         System.out.println("names is "+length+" characters wide");
7
8         char ch = names.charAt(0);
9         System.out.println("Character at index 0 is: "+ch);
10
11        ch = names.charAt(length-1);
12        System.out.println("Character at length-1 is: "+ch);
13
14        int idx = names.indexOf('o');
15        System.out.println("Index of o is: "+idx);
16    }
17
18 }

```

On the right, the **Console** view shows the output of the program:

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/openFiles/StringMethods.java:15:20: characters wide
names is 20 characters wide
Character at index 0 is: J
Character at length-1 is: e
Index of o is: 1

```

## 2.7 Run the code and you can see an index of o is 1

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays a Java file named StringMethods.java. The code defines a class StringMethods with a main method that prints the length of a string and its characters at index 0 and length-1. It also prints the index of the character 'o'. On the right, the Console view shows the output of the program:

```
<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.java-11-ea-elemosnips/jdk-11.0.2+9!/bin/java -Dfile.encoding=UTF-8 -cp /usr/eclipse/plugins/org.eclipse.justj.openjdk.java-11-ea-elemosnips/jdk-11.0.2+9!/bin:/usr/eclipse/plugins/org.eclipse.justj.openjdk.java-11-ea-elemosnips/jdk-11.0.2+9!/lib/* StringMethods
names is 28 characters wide
Character at index 0 is: J
Character at length-1 is: e
Index of o is: 1
```

## 2.8 You can also add `idx = names.lastIndexOf('o');`. There is another method called `lastIndexOf`, which will give you the index of the character from the end of the string

The screenshot shows the Eclipse IDE interface. The code editor displays the same Java file StringMethods.java. The code has been modified to include the `lastIndexOf` method. The new line of code is:

```
idx = names.lastIndexOf('o');
```

2.9 Run this code: **length - 2**. The length is 28, so the index of 'O' is **28 - 2**, which is **26**

```

StringMethods.java Run StringMethods
1
2  public class StringMethods {
3
4    public static void main(String[] args) {
5
6      String names = "John, Jennie, Jim, Jack, Joe";
7      int length = names.length();
8      System.out.println("names is "+length+" characters wide");
9
10     char ch = names.charAt(0);
11     System.out.println("Character at index 0 is: "+ch);
12
13     ch = names.charAt(length-1);
14     System.out.println("Character at length-1 is: "+ch);
15
16     int idx = names.indexOf('o');
17     System.out.println("Index of o is: "+idx);
18
19     idx = names.lastIndexOf('o');
20     System.out.println("Last Index of o is: "+idx);
21
22
23   }
24
25 }
26

```

2.10 Use **idx = name.indexOf("Jim")**; instead of a character. You can pass a string, such as "Jim". The index will be where "Jim" starts. You are going to add the index of "Jim" to the existing index

```

StringMethods.java X
1
2  public class StringMethods {
3
4    public static void main(String[] args) {
5
6      String names = "John, Jennie, Jim, Jack, Joe";
7      int length = names.length();
8      System.out.println("names is "+length+" characters wide");
9
10     char ch = names.charAt(0);
11     System.out.println("Character at index 0 is: "+ch);
12
13     ch = names.charAt(length-1);
14     System.out.println("Character at length-1 is: "+ch);
15
16     int idx = names.indexOf('o');
17     System.out.println("Index of o is: "+idx);
18
19     idx = names.lastIndexOf('o');
20     System.out.println("Last Index of o is: "+idx);
21
22     idx = names.indexOf("Jim");
23     System.out.println("Idx of Jim is: "+idx);
24
25   }
26
27
28 }
29

```

2.11 Run the code. Now, this is where Jim's J starts, so that's like 14.

```

StringMethods.java
1
2 public class StringMethods {
3
4     public static void main(String[] args) {
5
6         String names = "John, Jennie, Jim, Jack, Joe";
7         int length = names.length();
8         System.out.println("names is "+length+" characters wide");
9
10        char ch = names.charAt(0);
11        System.out.println("Character at index 0 is: "+ch);
12
13        ch = names.charAt(length-1);
14        System.out.println("Character at length-1 is: "+ch);
15
16        int idx = names.indexOf('o');
17        System.out.println("Index of o is: "+idx);
18
19        idx = names.lastIndexOf('o');
20        System.out.println("Last Index of o is: "+idx);
21
22        idx = names.indexOf("Jim");
23        System.out.println("Idx of Jim is: "+idx);
24
25
26    }
27
28 }

```

### Step 3: Use the uppercase method to convert the string to uppercase

3.1 Type **names.toUpperCase()**. The method **toUpperCase** is very clear in its function; it converts the string to uppercase. Use the print command to print the original names variable followed by the uppercase version

```

StringMethods.java
1
2 public class StringMethods {
3
4     public static void main(String[] args) {
5
6         String names = "John, Jennie, Jim, Jack, Joe";
7         int length = names.length();
8         System.out.println("names is "+length+" characters wide");
9
10        char ch = names.charAt(0);
11        System.out.println("Character at index 0 is: "+ch);
12
13        ch = names.charAt(length-1);
14        System.out.println("Character at length-1 is: "+ch);
15
16        int idx = names.indexOf('o');
17        System.out.println("Index of o is: "+idx);
18
19        idx = names.lastIndexOf('o');
20        System.out.println("Last Index of o is: "+idx);
21
22        idx = names.indexOf("Jim");
23        System.out.println("Idx of Jim is: "+idx);
24
25        names.toUpperCase();
26        System.out.println("names: "+names);
27
28    }
29

```

3.2 Run the program here, you will get to see the names are still the same. There are some methods that are meant to do the manipulation on those strings

```

StringMethods.java
1 public class StringMethods {
2     public static void main(String[] args) {
3         String names = "John, Jennie, Jim, Jack, Joe";
4         int length = names.length();
5         System.out.println("names is "+length+" characters wide");
6         char ch = names.charAt(0);
7         System.out.println("Character at index 0 is: "+ch);
8         ch = names.charAt(length-1);
9         System.out.println("Character at length-1 is: "+ch);
10        int idx = names.indexOf('o');
11        System.out.println("Index of o is: "+idx);
12        idx = names.lastIndexOf('o');
13        System.out.println("Last Index of o is: "+idx);
14        idx = names.indexOf("Jim");
15        System.out.println("idx of Jim is: "+idx);
16        names.toUpperCase();
17        System.out.println("names: "+names);
18    }
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core:/Applications/java - Session11/src/StringMethods.java - Eclipse IDE  
names is 20 characters wide  
Character at index 0 is: J  
Character at length-1 is: e  
Index of o is: 1  
Last Index of o is: 20  
idx of Jim is: 14  
names: John, Jennie, Jim, Jack, Joe

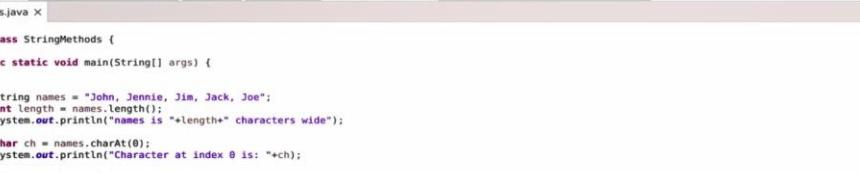
3.3 Now here you need to understand the fundamental, which is strings are immutable. Immutable means that strings cannot be modified

```

StringMethods.java
1 public class StringMethods {
2     public static void main(String[] args) {
3         String names = "John, Jennie, Jim, Jack, Joe";
4         int length = names.length();
5         System.out.println("names is "+length+" characters wide");
6         char ch = names.charAt(0);
7         System.out.println("Character at index 0 is: "+ch);
8         ch = names.charAt(length-1);
9         System.out.println("Character at length-1 is: "+ch);
10        int idx = names.indexOf('o');
11        System.out.println("Index of o is: "+idx);
12        idx = names.lastIndexOf('o');
13        System.out.println("Last Index of o is: "+idx);
14        idx = names.indexOf("Jim");
15        System.out.println("idx of Jim is: "+idx);
16        // Strings are IMMUTABLE
17        // Strings cannot be modified
18        names.toUpperCase();
19        System.out.println("names: "+names);
20    }
21 }
22
23
24
25
26
27
28
29
30
31
32
33

```

3.4 When you type `names.toUpperCase()`, it creates a new string in memory and returns it. Hence, you will have another string called `uppercaseNames` (or any name of your choice). If you print the string called `uppercaseNames`, it will display the converted string



The screenshot shows the Eclipse IDE interface with the title bar "java - Session11/src/StringMethods.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The left sidebar shows the package structure with "StringMethods.java X" selected. The main editor area contains the following Java code:

```
StringMethods.java X

1
2 public class StringMethods {
3
4     public static void main(String[] args) {
5
6         String names = "John, Jennie, Jim, Jack, Joe";
7         int length = names.length();
8         System.out.println("names is "+length+" characters wide");
9
10        char ch = names.charAt(0);
11        System.out.println("Character at index 0 is: "+ch);
12
13        ch = names.charAt(length-1);
14        System.out.println("Character at length-1 is: "+ch);
15
16        int idx = names.indexOf('o');
17        System.out.println("Index of o is: "+idx);
18
19        idx = names.lastIndexOf('o');
20        System.out.println("Last Index of o is: "+idx);
21
22        idx = names.indexOf("Jim");
23        System.out.println("Idx of Jim is: "+idx);
24
25        // Strings are IMMUTABLE
26        // Strings cannot be modified
27        String upperCaseNames = names.toUpperCase();
28        System.out.println("names: "+names);
29        System.out.println("upperCaseNames: "+upperCaseNames);
30
31    }
32
33
34 }
```

3.5 Run this code what you see is that the upper-case names are basically containing the characters in the upper case, so this is a return string through to the uppercase method

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session11/src/StringMethods.java
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Panel:** Shows the current file is "StringMethods.java". The code is as follows:

```
1  public class StringMethods {  
2      public static void main(String[] args) {  
3          String names = "John, Jennie, Jim, Jack, Joe";  
4          int length = names.length();  
5          System.out.println("names is "+length+" characters wide");  
6  
7          char ch = names.charAt(0);  
8          System.out.println("Character at index 0 is: "+ch);  
9  
10         ch = names.charAt(length-1);  
11         System.out.println("Character at length-1 is: "+ch);  
12  
13         int id = names.indexOf('o');  
14         System.out.println("Index of o is: "+id);  
15  
16         idx = names.lastIndexOf('o');  
17         System.out.println("Last Index of o is: "+idx);  
18  
19         idx = names.indexOf("Jim");  
20         System.out.println("Idx of Jim is: "+idx);  
21  
22         // Strings are IMMUTABLE  
23         // Strings cannot be modified  
24         String upperCaseNames = names.toUpperCase();  
25         System.out.println("names: "+names);  
26         System.out.println("upperCaseNames: "+upperCaseNames);  
27  
28     }  
29  
30 }
```

- Right Panel:** Shows the output of the code execution in the "Console" view:

```
<terminated> StringMethods [java Application] /usr/eclipse/plugins/org.eclipse.justopened.com  
names is 28 characters wide  
Character at index 0 is: J  
Character at length-1 is: e  
Index of o is: 1  
Last Index of o is: 26  
Idx of Jim is: 14  
names: John, Jennie, Jim, Jack, Joe  
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
```

3.6 For a given context, you can even say, spring replaced is going to be named ‘.replace’. Let the old character j be replaced with a new character k. If you print out the names. You will find that names have no change. And when you add, replaced names, it prints down the replace string

```

1  public class StringMethods {
2
3     public static void main(String[] args) {
4
5         String names = "John, Jennie, Jim, Jack, Joe";
6         int length = names.length();
7         System.out.println("names is "+length+" characters wide");
8
9         char ch = names.charAt(0);
10        System.out.println("Character at index 0 is: "+ch);
11
12        ch = names.charAt(length-1);
13        System.out.println("Character at length-1 is: "+ch);
14
15        int idx = names.indexOf('o');
16        System.out.println("Index of o is: "+idx);
17
18        idx = names.lastIndexOf('o');
19        System.out.println("Last Index of o is: "+idx);
20
21        idx = names.indexOf("Jim");
22        System.out.println("Idx of Jim is: "+idx);
23
24        // Strings are IMMUTABLE
25        // Strings cannot be modified
26        String upperCaseNames = names.toUpperCase();
27        System.out.println("names: "+names);
28        System.out.println("upperCaseNames: "+upperCaseNames);
29
30        String replaced = names.replace('J', 'K');
31        System.out.println("names: "+names);
32        System.out.println("replaced names: "+replaced);
33
34

```

3.7 Run this program, and you get to see the data is manipulated. But the names are not at all changed

```

1  public class StringMethods {
2
3     public static void main(String[] args) {
4
5         String names = "John, Jennie, Jim, Jack, Joe";
6         int length = names.length();
7         System.out.println("names is "+length+" characters wide");
8
9         char ch = names.charAt(0);
10        System.out.println("Character at index 0 is: "+ch);
11
12        ch = names.charAt(length-1);
13        System.out.println("Character at length-1 is: "+ch);
14
15        int idx = names.indexOf('o');
16        System.out.println("Index of o is: "+idx);
17
18        idx = names.lastIndexOf('o');
19        System.out.println("Last Index of o is: "+idx);
20
21        idx = names.indexOf("Jim");
22        System.out.println("Idx of Jim is: "+idx);
23
24        // Strings are IMMUTABLE
25        // Strings cannot be modified
26        String upperCaseNames = names.toUpperCase();
27        System.out.println("names: "+names);
28        System.out.println("upperCaseNames: "+upperCaseNames);
29
30        String replaced = names.replace('J', 'K');
31        System.out.println("names: "+names);
32
33

```

The Console output shows:

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openj9
names is 28 characters wide
Character at index 0 is: J
Character at length-1 is: e
Index of o is: 1
Last Index of o is: 26
Idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jennie, Jim, Jack, Joe
replaced names: Kohn, Kennie, Kim, Kack, Koe

```

## **Step 4: Implement string slicing and extracting sub-strings**

4.1 Now you are going to extract a substring from a string, which is called string slicing. Type `names.substring` and pass the starting index of 'Jim'. This will create a new string in memory. Let us name this substring '**substring**' and store it at the specified index. Print out the substring by using `System.out.println("Substring is " + substring);`



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** java - Session11/src/StringMethods.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Includes icons for Save, Undo, Redo, Cut, Copy, Paste, Find, and others.
- Left Sidebar:** Shows the package structure (src) and a tree view of files.
- Code Editor:** Displays the Java code for `StringMethods.java`. The code demonstrates various string manipulation methods like `length()`, `charAt()`, `indexOf()`, `lastIndexOf()`, `replace()`, and `substring()`.

```
1 public class StringMethods {  
2     public static void main(String[] args) {  
3         String names = "John, Jennie, Jim, Jack, Joe";  
4         int length = names.length();  
5         System.out.println("names is "+length+" characters wide");  
6         char ch = names.charAt(0);  
7         System.out.println("Character at index 0 is: "+ch);  
8         ch = names.charAt(length-1);  
9         System.out.println("Character at length-1 is: "+ch);  
10        int idx = names.indexOf('o');  
11        System.out.println("Index of o is: "+idx);  
12        idx = names.lastIndexOf('o');  
13        System.out.println("Last Index of o is: "+idx);  
14        idx = names.indexOf("Jim");  
15        System.out.println("Idx of Jim is: "+idx);  
16        // Strings are IMMUTABLE  
17        // Strings cannot be modified  
18        String upperCaseNames = names.toUpperCase();  
19        System.out.println("names: "+names);  
20        System.out.println("upperCaseNames: "+upperCaseNames);  
21        String replaced = names.replace('J', 'K');  
22        System.out.println("names: "+names);  
23        System.out.println("replaced: names: "+replaced);  
24        String subString = names.substring(idx);  
25        System.out.println("subString is: "+subString);  
26    }  
27}
```

4.2 Run the code. A substring with one single input will simply slice your string from that index till the end

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session11/src/StringMethods.java
- Menu Bar:** File Edit Source Refactor Navigate Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Margin:** Shows line numbers from 1 to 34.
- Code Editor:** Displays the Java code for `StringMethods.java`. The code prints various string manipulations to the console.
- Console View:** Shows the output of the program:

```
<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.just/jopenjd
names is 28 characters wide
Character at index 0 is: J
Character at length-1 is: e
End of string
Last Index of o is: 26
idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jennie, Jim, Jack, Joe
replacedNames: Kohn, Kennie, Kim, Kack, Koe
subString is: Jim, Jack, Joe
```

4.3 Now try to write one more usage of this substring method, which is an overloaded method, you can pass the index, and then you can pass a number called 3

```

1  package com.simplilearn;
2
3  public class StringMethods {
4      public static void main(String[] args) {
5          String names = "John, Jennie, Jim, Jack, Joe";
6          int length = names.length();
7          System.out.println("names is "+length+" characters wide");
8
9          char ch = names.charAt(0);
10         System.out.println("Character at index 0 is: "+ch);
11
12         ch = names.charAt(length-1);
13         System.out.println("Character at length-1 is: "+ch);
14
15         int idx = names.indexOf('o');
16         System.out.println("Index of o is: "+idx);
17
18         idx = names.lastIndexOf('o');
19         System.out.println("Last Index of o is: "+idx);
20
21         idx = names.indexOf("Jim");
22         System.out.println("Idx of Jim is: "+idx);
23
24         // Strings are IMMUTABLE
25         // Strings cannot be modified
26         String upperCaseNames = names.toUpperCase();
27         System.out.println("names: "+names);
28         System.out.println("upperCaseNames: "+upperCaseNames);
29
30         String replaced = names.replace('J', 'K');
31         System.out.println("names: "+names);
32         System.out.println("replaced names: "+replaced);
33
34         /*String subString = names.substring(idx);
35         String subString = names.substring(idx, 3);
36         System.out.println("subString is: "+subString);
37 */
38     }
39 }

```

4.4 When you run this code. You get to see that there is an error. Now the index is the beginning point and the second input is the ending point

```

1  package com.simplilearn;
2
3  public class StringMethods {
4      public static void main(String[] args) {
5          String names = "John, Jennie, Jim, Jack, Joe";
6          int length = names.length();
7          System.out.println("names is "+length+" characters wide");
8
9          char ch = names.charAt(0);
10         System.out.println("Character at index 0 is: "+ch);
11
12         ch = names.charAt(length-1);
13         System.out.println("Character at length-1 is: "+ch);
14
15         int idx = names.indexOf('o');
16         System.out.println("Index of o is: "+idx);
17
18         idx = names.lastIndexOf('Jim');
19         System.out.println("Idx of Jim is: "+idx);
20
21         // Strings are IMMUTABLE
22         // Strings cannot be modified
23         String upperCaseNames = names.toUpperCase();
24         System.out.println("names: "+names);
25         System.out.println("upperCaseNames: "+upperCaseNames);
26
27         String replaced = names.replace('J', 'K');
28         System.out.println("names: "+names);
29
30         /*String subString = names.substring(idx);
31         String subString = names.substring(idx, 3);
32         System.out.println("subString is: "+subString);
33 */
34     }
35 }

```

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin index 31 is greater than end index 3 at java.base/java.lang.String.checkBoundsBeginEnd(String.java:411) at java.base/java.lang.String.substring(String.java:2704) at StringMethods.main(StringMethods.java:37)

4.5 The index of Jim over gives the output as 14, which means the next index cannot be less than 14. It must be larger than. You're going to add 17

```

java - Session11/src/StringMethods.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X
StringMethods.java X
17 int idx = names.indexOf('o');
18 System.out.println("Index of o is: "+idx);
19
20 idx = names.lastIndexOf('o');
21 System.out.println("Last Index of o is: "+idx);
22
23 idx = names.indexOf("Jim");
24 System.out.println("idx of Jim is: "+idx);
25
26 // Strings are IMMUTABLE
27 // Strings cannot be modified
28 String upperCaseNames = names.toUpperCase();
29 System.out.println("names: "+names);
30 System.out.println("upperCaseNames: "+upperCaseNames);
31
32 String replaced = names.replace('J', 'K');
33 System.out.println("names: "+names);
34 System.out.println("replaced names: "+replaced);
35
36 /*String subString = names.substring(idx, 17);
37 String subString = names.substring(idx, 17);
38 System.out.println("subString is: "+subString);*/
39

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj/openjdk.hotspot.jre.full/linux/x86\_64\_17.0.1.v20211116-1657/re/bin/java [javac]
Character at length-1 is: e
Index of o is: 1
Last Index of o is: 26
idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jennie, Jim, Jack, Joe
replaced names: Kohn, Kennie, Kim, Kack, Koe
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin 14, end 3, length 28
at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)

4.6 Now run the same program and see the entire word called Jim being extracted from the string. It started with the 14, then the 15, and then the 16. 17 is not inclusive. It is less than 17

```

java - Session11/src/StringMethods.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X
StringMethods.java X
17 int idx = names.indexOf('o');
18 System.out.println("Index of o is: "+idx);
19
20 idx = names.lastIndexOf('o');
21 System.out.println("Last Index of o is: "+idx);
22
23 idx = names.indexOf("Jim");
24 System.out.println("idx of Jim is: "+idx);
25
26 // Strings are IMMUTABLE
27 // Strings cannot be modified
28 String upperCaseNames = names.toUpperCase();
29 System.out.println("names: "+names);
30 System.out.println("upperCaseNames: "+upperCaseNames);
31
32 String replaced = names.replace('J', 'K');
33 System.out.println("names: "+names);
34 System.out.println("replaced names: "+replaced);
35
36 /*String subString = names.substring(idx, 17);
37 String subString = names.substring(idx, 17);
38 System.out.println("subString is: "+subString);*/
39

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj/openjdk.hotspot.jre.full/linux/x86\_64\_17.0.1.v20211116-1657/re/bin/java [javac]
Character at length-1 is: e
Index of o is: 1
Last Index of o is: 26
idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jennie, Jim, Jack, Joe
replaced names: Kohn, Kennie, Kim, Kack, Koe
subString is: Jim

4.7 You are going to use a comma, which typically means that you will be splitting the string based on the comma. When you execute this split method, you will get an array of strings in return. Type **String[] results = name.split(",");**

```

    11     char ch = names.charAt(0);
    12     System.out.println("Character at index 0 is: "+ch);
    13
    14     ch = names.charAt(length-1);
    15     System.out.println("Character at length-1 is: "+ch);
    16
    17     int idx = names.indexOf('o');
    18     System.out.println("Index of o is: "+idx);
    19
    20     idx = names.lastIndexOf('o');
    21     System.out.println("Last Index of o is: "+idx);
    22
    23     idx = names.indexOf("Jim");
    24     System.out.println("Idx of Jim is: "+idx);
    25
    26     // Strings are IMMUTABLE
    27     // Strings cannot be modified
    28     String upperCaseNames = names.toUpperCase();
    29     System.out.println("names: "+names);
    30     System.out.println("upperCaseNames: "+upperCaseNames);
    31
    32     String replaced = names.replace('J', 'K');
    33     System.out.println("names: "+names);
    34     System.out.println("replaced names: "+replaced);
    35
    36     /*String subString = names.substring(idx);
    37     String subString = names.substring(idx, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
    38     System.out.println("subString is: "+subString);
    39
    40     String[]results = names.split(",");
    41     for(String name: results) {
    42         System.out.println(name);
    43     }
    44 */
  
```

4.8 Run the code and it gives the data as John, Jenny, Jim, Jack, and Joe. But you can observe that there is a space coming in the leading of the string. The reason is that whenever you are trying to come up and perform a split that was based on a comma, you do have a space attached to it

```

<terminated> StringMethods [Java Application] Ausr/eclipse/plugins/org.eclipse.justmyjava
names is 28 characters wide
Character at index 0 is: J
Character at length-1 is: e
Index of o is: 1
Last Index of o is: 26
idx of Jim is: 14
names: John, Jenny, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jenny, Jim, Jack, Joe
replaced names: Kohn, Kennie, Kim, Kack, Koe
subString is: Jim
John
  Jennie
  Jim
  Jack
  Joe
  
```

## Step 5: Use the trim function on strings

5.1 Now there are two remedies to this. You can use a function or method called trim which is used to eliminate the wide spaces from the leading and trailing you can see now there is no whitespace coming in front of the string

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session11/src/StringMethods.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Panel:** Shows the project structure under "StringMethods.java X".
- Code Editor:** Displays the Java code for `StringMethods.java`. The code includes various string manipulation methods like `charAt`, `indexOf`, `lastIndexOf`, `toUpperCase`, `replace`, and `split`.
- Console View:** Shows the output of the Java application. It prints:
  - names is 28 characters wide
  - Character at index 0 is: J
  - Character at length-1 is: e
  - Index of o is: 1
  - Last Index of o is: 26
  - idx of Jim is: 14
  - names: John, Jennie, Jim, Jack, Joe
  - upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
  - names: John, Jennie, Jim, Jack, Joe
  - replaced names: Kohn, Kennie, Kim, Kack, Koe
  - subString is: JimBelow the console, there is a small preview window showing the first few characters of the string "John Jennie Jim Jack Joe".

5.2 The other way is to perform a split right based on a comma with a space. when you do a split as comma space together, it will split based on comma and space. This can even be a regular expression

The screenshot shows the Eclipse IDE interface with the following details:

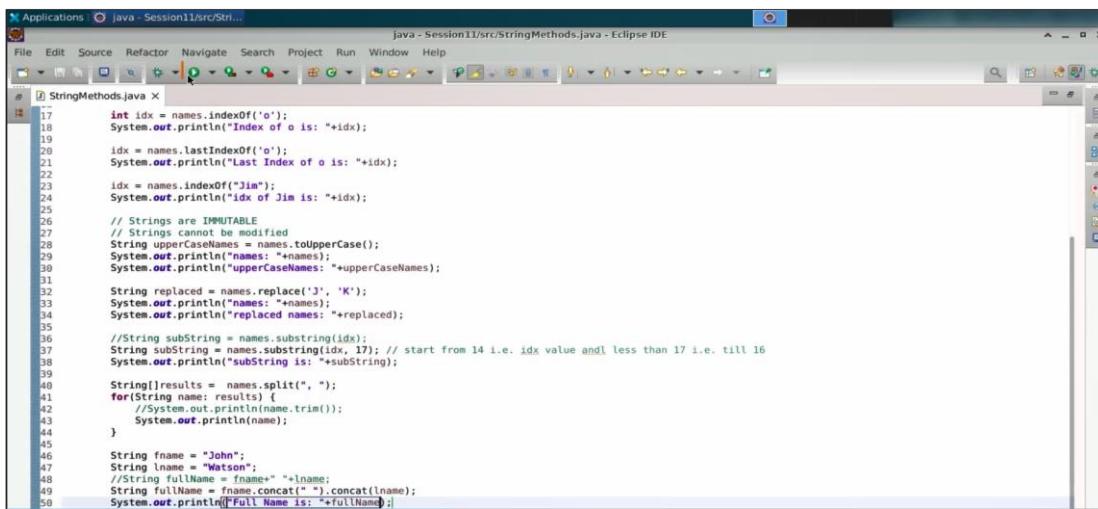
- Title Bar:** Applications > java - Session11/src/StringMethods.java
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Panel (Code View):** The code editor displays `StringMethods.java` containing Java code that manipulates strings, including operations like `charAt`, `length`, `indexOf`, `lastIndexOf`, `replace`, `split`, and `substring`.
- Right Panel (Output View):** The console output shows the results of the program's execution, including the original string, character extraction, index locations, modified strings, and split results.

```
StringMethods.java X
12   char ch = names.charAt(0);
13   System.out.println("Character at index 0 is: "+ch);
14   ch = names.charAt(length-1);
15   System.out.println("Character at length-1 is: "+ch);
16
17   int idx = names.indexOf('o');
18   System.out.println("Index of o is: "+idx);
19
20   idx = names.lastIndexOf('o');
21   System.out.println("Last Index of o is: "+idx);
22
23   idx = names.indexOf("Jim");
24   System.out.println("Idx of Jim is: "+idx);
25
26   // Strings are IMMUTABLE
27   // Strings cannot be modified
28   String upperCaseNames = names.toUpperCase();
29   System.out.println("names: "+names);
30   System.out.println("upperCaseNames: "+upperCaseNames);
31
32   String replaced = names.replace('J', 'K');
33   System.out.println("names: "+names);
34   System.out.println("replaced names: "+replaced);
35
36   //String subString = names.substring(idx);
37   String subString = names.substring(idx, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
38   System.out.println("subString is: "+subString);
39
40   String[]results = names.split(",");
41   for(String name: results) {
42       //System.out.println(name.trim());
43       System.out.println(name);
44   }
```

```
<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj/openide/names is 28 characters wide
Character at index 0 is: J
Character at length-1 is: e
Index of o is: 4
Last Index of o is: 26
Idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jennie, Jim, Jack, Joe
replaced names: Kohn, Kennie, Kim, Kack, Koe
substring results: Jim
John
Jennie
Jim
Jack
Joe I
```

## Step 6: Implement the concat method for concatenation of strings

6.1 There is a string called **firstName**, let's say "**John**", and there is a string called **lastName**, which will be "**Watson**". Now you are going to create a string called **fullName**. You will add **firstName**, a space, and then **lastName**. If you wish, you can also use the concat method. You can type **fullName = firstName.concat(" ").concat(lastName);**. Then simply print out **fullName**

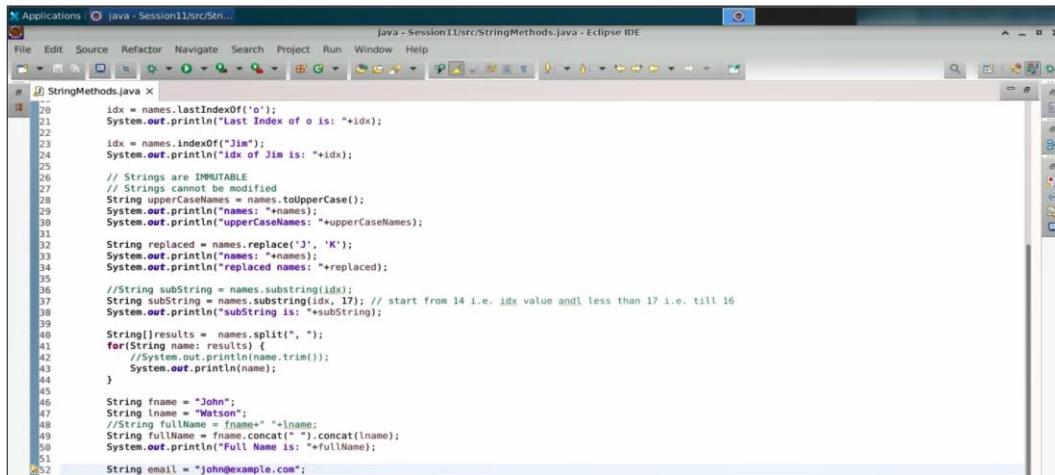


```

17     int idx = names.indexOf('o');
18     System.out.println("Index of 'o' is: "+idx);
19
20     idx = names.lastIndexOf('o');
21     System.out.println("Last Index of 'o' is: "+idx);
22
23     idx = names.indexOf("Jim");
24     System.out.println("idx of Jim is: "+idx);
25
26     // Strings are IMMUTABLE
27     // Strings cannot be modified
28     String upperCaseNames = names.toUpperCase();
29     System.out.println("names: "+names);
30     System.out.println("upperCaseNames: "+upperCaseNames);
31
32     String replaced = names.replace('J', 'K');
33     System.out.println("names: "+names);
34     System.out.println("replaced names: "+replaced);
35
36     //String subString = names.substring(idx);
37     String subString = names.substring(idx, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
38     System.out.println("subString is: "+subString);
39
40     String[]results = names.split(", ");
41     for(String name: results) {
42         //System.out.println(name.trim());
43         System.out.println(name);
44     }
45
46     String fname = "John";
47     String lname = "Watson";
48     //String fullName = fname+" "+lname;
49     String fullName = fname.concat(" ").concat(lname);
50     System.out.println("Full Name is: "+fullName);
51
52

```

6.2 Now let us say that there is an email. And this email is **john@example.com**. You need to validate that whether this email is correctly formed or not



```

20     idx = names.lastIndexOf('o');
21     System.out.println("Last Index of 'o' is: "+idx);
22
23     idx = names.indexOf("Jim");
24     System.out.println("idx of Jim is: "+idx);
25
26     // Strings are IMMUTABLE
27     // Strings cannot be modified
28     String upperCaseNames = names.toUpperCase();
29     System.out.println("names: "+names);
30     System.out.println("upperCaseNames: "+upperCaseNames);
31
32     String replaced = names.replace('J', 'K');
33     System.out.println("names: "+names);
34     System.out.println("replaced names: "+replaced);
35
36     //String subString = names.substring(idx);
37     String subString = names.substring(idx, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
38     System.out.println("subString is: "+subString);
39
40     String[]results = names.split(", ");
41     for(String name: results) {
42         //System.out.println(name.trim());
43         System.out.println(name);
44     }
45
46     String fname = "John";
47     String lname = "Watson";
48     //String fullName = fname+" "+lname;
49     String fullName = fname.concat(" ").concat(lname);
50     System.out.println("Full Name is: "+fullName);
51
52     String email = "john@example.com";
53

```

6.3 You are going to put a condition: **if email.isEmpty()**, print "Email cannot be empty".

Otherwise, handle the else case

```

25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced names: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(17); // start from 14 i.e. idx value andl less than 17 i.e. till 16
37 System.out.println("subString is: "+subString);
38
39 String[]results = names.split(", ");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fname = "John";
46 String lname = "Watson";
47 //String fullName = fname+" "+lname;
48 String fullName = fname.concat(" ").concat(lname);
49 System.out.println("Full Name is: "+fullName);
50
51 String email = "john@example.com";
52 if(email.isEmpty()) {
53     System.out.println("Email Cannot be Empty");
54 }else {
55 }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
780
781
782
783
784
785
786
787
788
788
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
22
```

6.5 Run this program, now, it says that the email is valid, and proceed with registration.

```

25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced names: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(17); // start from 14 i.e. idx value andl less than 17 i.e. till 16
37 System.out.println("subString is: "+subString);
38
39 String[]results = names.split(",");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fName = "John";
46 String lName = "Watson";
47 //String fullName = fName+" "+lName;
48 String fullName = fName.concat(" ").concat(lName);
49 System.out.println("Full Name is: "+fullName);
50
51 String email = "john@example.com";
52 if(email.isEmpty()) {
53     System.out.println("Email Cannot be Empty");
54 }else {
55     if(email.contains("@") && email.endsWith(".com")) {
56         System.out.println("Email is Valid. Proceed with Registration");
57     }
58 }

```

6.6 Let us try to say that the email goes something like a blank

```

25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced names: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(17); // start from 14 i.e. idx value andl less than 17 i.e. till 16
37 System.out.println("subString is: "+subString);
38
39 String[]results = names.split(",");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fName = "John";
46 String lName = "Watson";
47 //String fullName = fName+" "+lName;
48 String fullName = fName.concat(" ").concat(lName);
49 System.out.println("Full Name is: "+fullName);
50
51 String email = "";
52 if(email.isEmpty()) {
53     System.out.println("Email Cannot be Empty");
54 }else {
55     if(email.contains("@") && email.endsWith(".com")) {
56         System.out.println("Email is Valid. Proceed with Registration");
57     }
58 }

```

## 6.7 Run this program, now, it says Email Cannot Be Empty

```

StringMethods.java
25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(14, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
37 System.out.println("subString is: "+subString);
38
39 String[] results = names.split(",");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fname = "John";
46 String lname = "Watson";
47 //String fullName = fname+" "+lname;
48 String fullName = fname.concat(" ").concat(lname);
49 System.out.println("Full Name is: "+fullName);
50
51 String email = "";
52 if(email.isEmpty()) {
53     System.out.println("Email Cannot be Empty");
54 } else {
55     if(email.contains("@") && email.endsWith(".com")) {
56         System.out.println("Email is Valid. Proceed with Registration");
57     }
58 }

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk/bin/charAtStringMethods.java:14: error: character at index 0 is: J  
Character at index 0 is: J  
Character at length-1 is: e  
Index of o is: 1  
Last Index of o is: 26  
Index of J is: 14  
names: John, Jennie, Jim, Jack, Joe  
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE  
names: John, Jennie, Jim, Jack, Joe  
replaced names: Kohn, Kennie, Kim, Kack, Koe  
subString is: Jim  
John  
Jennie  
Jim  
Jack  
Joe  
Full Name is: John Watson  
Email Iannot be Empty

## 6.8 Let us simply remove this @. So, contains check anything as a part of your string

```

StringMethods.java
25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(14, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
37 System.out.println("subString is: "+subString);
38
39 String[] results = names.split(",");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fname = "John";
46 String lname = "Watson";
47 //String fullName = fname+" "+lname;
48 String fullName = fname.concat(" ").concat(lname);
49 System.out.println("Full Name is: "+fullName);
50
51 String email = "john@example.com";
52 if(email.isEmpty()) {
53     System.out.println("Email Cannot be Empty");
54 } else {
55     if(email.contains("@") && email.endsWith(".com")) {
56         System.out.println("Email is Valid. Proceed with Registration");
57     }
58 }

```

## 6.9 Run this program. Now, you can see else part is not written

```

25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced names: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(idx, 17); // start from 14 i.e. idx value andl less than 17 i.e. till 16
37 System.out.println("substring is: "+subString);
38
39 String[]results = names.split(", ");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fname = "John";
46 String lname = "Watson";
47 //String fullName = fname+" "+lname;
48 String fullName = fname.concat(" ").concat(lname);
49 System.out.println("Full Name is: "+fullName);
50
51
52 String email = "john@example.com";
53 if(email.isEmpty()) {
54     System.out.println("Email Cannot be Empty");
55 }else {
56     if(email.contains("@") && email.endsWith(".com")) {
57         System.out.println("Email is Valid. Proceed with Registration");
58     }
59 }
60

```

## 6.10 You can once again come up and say that this is an invalid email. You can even print the variable

```

25 // Strings are IMMUTABLE
26 // Strings cannot be modified
27 String upperCaseNames = names.toUpperCase();
28 System.out.println("names: "+names);
29 System.out.println("upperCaseNames: "+upperCaseNames);
30
31 String replaced = names.replace('J', 'K');
32 System.out.println("names: "+names);
33 System.out.println("replaced names: "+replaced);
34
35 //String subString = names.substring(idx);
36 String subString = names.substring(idx, 17); // start from 14 i.e. idx value andl less than 17 i.e. till 16
37 System.out.println("substring is: "+subString);
38
39 String[]results = names.split(", ");
40 for(String name: results) {
41     //System.out.println(name.trim());
42     System.out.println(name);
43 }
44
45 String fname = "John";
46 String lname = "Watson";
47 //String fullName = fname+" "+lname;
48 String fullName = fname.concat(" ").concat(lname);
49 System.out.println("Full Name is: "+fullName);
50
51
52 String email = "john@example.com";
53 if(email.isEmpty()) {
54     System.out.println("Email Cannot be Empty");
55 }else {
56     if(email.contains("@") && email.endsWith(".com")) {
57         System.out.println("Email is Valid. Proceed with Registration");
58     }else {
59         System.out.println("Invalid Email: "+email);
60     }
61 }

```

6.11 Run this program. Now, it says invalid email **Johnexample.com**

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - Java - Session11/src/StringMethods.java
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Panel:** Shows the file `StringMethods.java` open in the editor.
- Right Panel:** Shows the `Console` view with the following output:

```
<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justopened
Character at index 0 is: J
Character at length-1 is: e
End of s is: l
Last Index of o is: 26
idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
names: John, Jennie, Jim, Jack, Joe
replaced names: Kohn, Kennie, Kim, Kack, Koe
subString is: Jim
John
Jennie
Jim
Jack
Joe
Full Name is: John Watson
Invalid Email: :john@example.com
```

6.12 Let us have '@' at the same location, and you can eliminate the '.com' condition, or you can simply say this is not valid. However, as per the condition, it should end with '.com', so it will say "**Invalid email**". This clarifies what is meant by the `endsWith` method

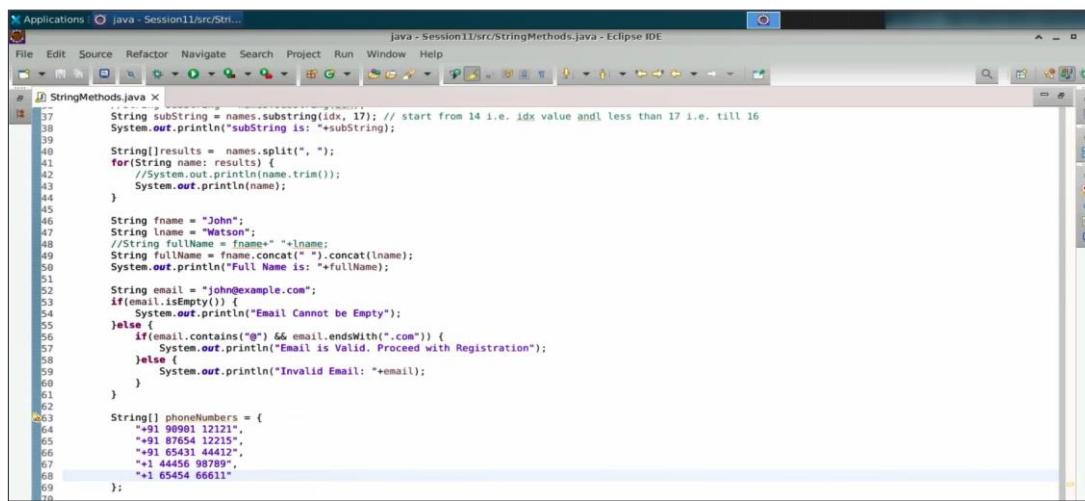
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session11/src/StringMethods.java
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Side:** Package Explorer and Navigator views.
- Code Editor:** The file `StringMethods.java` is open, containing Java code that manipulates strings using various methods like `toUpperCase`, `substring`, and `split`.
- Console View:** Shows the output of the program:

```
<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justmycode Character at index 0 is: J
Character at length-1 is: e
Index of o is: 5
Last character is: e is: 26
idx of Jim is: 14
names: John, Jennie, Jim, Jack, Joe
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
replaced names: Kohn, Kennie, Kim, Kack, Koe
subString is: Jim
John
Jennie
Jim
Jack
Joe
Full Name is: John Watson
Invalid Email: john@example.in
```
- Right Side:** Problems, Javadoc, Declaration, and Console tabs.

## Step 7: Validate the strings using the methods ends, contains ends with, and starts with

7.1 Let us try to explore something more on this string method called contains ends with and starts with. You are going to create a list of contacts or the phone numbers available in your address book. You have this array of phone numbers and you want to perform a search operation



The screenshot shows the Eclipse IDE interface with a Java file named 'StringMethods.java' open. The code demonstrates various string manipulation techniques:

```
String subString = names.substring(idx, 17); // start from 14 i.e. idx value and less than 17 i.e. till 16
System.out.println("subString is: "+subString);

String[] results = names.split(",");
for(String name: results) {
    //System.out.println(name.trim());
    System.out.println(name);
}

String fname = "John";
String lname = "Watson";
String fullName = fname.concat(" ").concat(lname);
System.out.println("Full Name is: "+fullName);

String email = "john@example.com";
if(email.isEmpty()) {
    System.out.println("Email Cannot be Empty");
} else {
    if(email.contains("@") && email.endsWith(".com")) {
        System.out.println("Email is Valid. Proceed with Registration");
    } else {
        System.out.println("Invalid Email: "+email);
    }
}

String[] phoneNumbers = {
    "+91 98901 12121",
    "+91 98901 12122",
    "+91 65431 44412",
    "+1 44456 98789",
    "+1 65454 66611"
};
```

7.2 Let us say there is a search keyword, and it is +91. You want to filter the phone numbers.

Put a loop that starts with the index value 0 and continues while the index is less than `myPhoneNumbers.length`. Use `idx++` to increment the index. Now, if the phone number at the current index starts with your search keyword, you can display this phone number. Add the phone numbers at the index



The screenshot shows the Eclipse IDE interface with the title bar "java - Session11/src/StringMethods.java - Eclipse IDE". The code editor displays Java code for a class named StringMethods. The code includes a for loop to print names, concatenation of first and last names, validation of an email address, and a search operation on an array of phone numbers.

```
StringMethods.java
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
```

7.3 Run this program, and it gives you all the phone numbers that start with +91

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session11/src/StringMethods...
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, and others.
- Code Editor:** Displays the `StringMethods.java` file containing Java code for validating names and emails, and searching phone numbers.
- Console View:** Shows the terminal output of the application's execution. It includes:
  - <terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.just.jopenid
  - Last Index of 0 is: 26
  - idx of 1 is: 15
  - names: John, Jennie, Jim, Jack, Joe
  - upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE
  - names: John, Jennie, Jim, Jack, Joe
  - replaced names: Kohn, Kennie, Kim, Kack, Koe
  - subString is: Jim
  - John
  - Jennie
  - Jim
  - Jack
  - Joe
  - Full Name is: John Watson
  - Email is Valid. Proceed with Registration
  - +91 99901 12121
  - +91 87654 12345
  - +91 65431 44422
  - +1 44456 98789
  - +1 65454 66611

7.4 If you just remove your search keyword with plus one, it is going to filter all your phone numbers with the +1

```

    Applications java - Session11/src/StringMethods.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StringMethods.java X
1  package com.simplilearn;
2
3  public class StringMethods {
4      public static void main(String[] args) {
5          String name = "John";
6          String lname = "Watson";
7          //String fullName = fname+" "+lname;
8          String fullName = fname.concat(" ").concat(lname);
9          System.out.println("Full Name is: "+fullName);
10         System.out.println();
11
12         String email = "john@example.com";
13         if(email.isEmpty()) {
14             System.out.println("Email Cannot be Empty");
15         }else {
16             if(email.contains("@") && email.endsWith(".com")) {
17                 System.out.println("Email is Valid. Proceed with Registration");
18             }else {
19                 System.out.println("Invalid Email: "+email);
20             }
21         }
22
23         String[] phoneNumbers = {
24             "+91 99991 12121",
25             "+91 87654 12215",
26             "+91 65431 44412",
27             "+1 44456 98789",
28             "+1 65454 66611"
29         };
30
31         String searchKeyword = "+1";
32
33         for(idx=0;idx<phoneNumbers.length;idx++) {
34             if(phoneNumbers[idx].startsWith(searchKeyword)) {
35                 System.out.println(phoneNumbers[idx]);
36             }
37         }
38     }
39 }

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.opened  
Index of o is: 1  
Last Index of o is: 26  
idx of Jim is: 14  
names: John, Jennie, Jim, Jack, Joe  
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE  
replaced names: John, Jennie, Jim, Jack, Joe  
subString is: Jim  
John  
Jennie  
Jim  
Jack  
Joe  
Full Name is: John Watson  
Email is Valid. Proceed with Registration  
+I 44456 98789  
+I 65454 66611

7.5 Let us see a magical thing where the search keyword can be "**12**". If you implement this with **startsWith**, it will not work. This is where you can use a very useful concept called **contains**. You will check if "**12**" is contained within the string, and you can see the list of phone numbers with "**12**" appearing anywhere in them

```

    Applications java - Session11/src/StringMethods.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StringMethods.java X
1  package com.simplilearn;
2
3  public class StringMethods {
4      public static void main(String[] args) {
5          String name = "John";
6          String lname = "Watson";
7          //String fullName = fname+" "+lname;
8          String fullName = fname.concat(" ").concat(lname);
9          System.out.println("Full Name is: "+fullName);
10         System.out.println();
11
12         String email = "john@example.com";
13         if(email.isEmpty()) {
14             System.out.println("Email Cannot be Empty");
15         }else {
16             if(email.contains("@") && email.endsWith(".com")) {
17                 System.out.println("Email is Valid. Proceed with Registration");
18             }else {
19                 System.out.println("Invalid Email: "+email);
20             }
21         }
22
23         String[] phoneNumbers = {
24             "+91 99991 12121",
25             "+91 87654 12215",
26             "+91 65431 44412",
27             "+1 44456 98789",
28             "+1 65454 66611"
29         };
30
31         //String searchKeyword = "+1";
32         String searchKeyword = "12";
33
34         for(idx=0;idx<phoneNumbers.length;idx++) {
35             /*if(phoneNumbers[idx].startsWith(searchKeyword)) {
36                 System.out.println(phoneNumbers[idx]);
37             }*/
38             if(phoneNumbers[idx].contains(searchKeyword)) {
39                 System.out.println(phoneNumbers[idx]);
40             }
41         }
42     }
43 }

```

<terminated> StringMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.opened  
Index of o is: 1  
Last Index of o is: 26  
idx of Jim is: 14  
names: John, Jennie, Jim, Jack, Joe  
upperCaseNames: JOHN, JENNIE, JIM, JACK, JOE  
replaced names: John, Jennie, Jim, Jack, Joe  
subString is: Jim  
John  
Jennie  
Jim  
Jack  
Joe  
Full Name is: John Watson  
Email is Valid. Proceed with Registration  
+91 99991 12121  
+91 87654 12215  
+91 65431 44412

## Step 8: Implement an algorithm to know the number of times a character appears

8.1 Let us also see one more method. Type **names.toCharArray**, and you are going to come up and convert all your characters into the names as an array, you are going to type **CharArray**, the Char names are your **names.toCharArray**. And for, CharArray it is in the char names and you are going to print out the character after the character prints a space

```

StringMethods.java
54     System.out.println("Email Cannot be Empty");
55     else {
56         if(email.contains("@") && email.endsWith(".com")) {
57             System.out.println("Email is Valid. Proceed with Registration");
58         } else {
59             System.out.println("Invalid Email: "+email);
60         }
61     }
62
63     String[] phoneNumbers = {
64         "+91 989901 12121",
65         "+91 87654 12215",
66         "+91 65431 44412",
67         "+91 44456 98789",
68         "+91 65454 66611"
69     };
70
71     //String searchKeyword = "+1";
72     String searchKeyword = "54";
73
74     for(idx=0;idx<phoneNumbers.length;idx++) {
75         /*if(phoneNumbers[idx].startsWith(searchKeyword)) {
76             System.out.println(phoneNumbers[idx]);
77         }*/
78
79         if(phoneNumbers[idx].contains(searchKeyword)) {
80             System.out.println(phoneNumbers[idx]);
81         }
82     }
83
84     char[] charNames = names.toCharArray();
85     for(char chr: charNames) {
86         System.out.print(chr+" ");
87     }
88
89 }
90
91
92
93
94

```

8.2 Now, maintain a count variable initialized to 0. If your character equals 'o', then increment the count. This is a simple algorithm that counts how many times a character appears. Add an empty print line here so that the output appears on the next line. Run the code, and you will see that 'o' appears two times

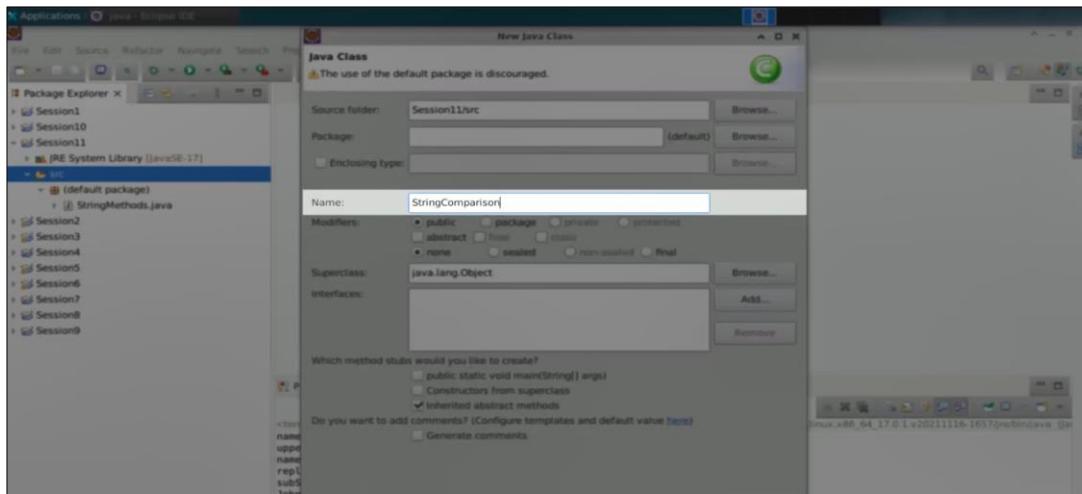
```

StringMethods.java
61
62
63     String[] phoneNumbers = {
64         "+91 989901 12121",
65         "+91 87654 12215",
66         "+91 65431 44412",
67         "+91 44456 98789",
68         "+91 65454 66611"
69     };
70
71     //String searchKeyword = "+1";
72     String searchKeyword = "54";
73
74     for(idx=0;idx<phoneNumbers.length;idx++) {
75         /*if(phoneNumbers[idx].startsWith(searchKeyword)) {
76             System.out.println(phoneNumbers[idx]);
77         }*/
78
79         if(phoneNumbers[idx].contains(searchKeyword)) {
80             System.out.println(phoneNumbers[idx]);
81         }
82     }
83
84     int count = 0;
85     char[] charNames = names.toCharArray();
86     for(char chr: charNames) {
87         System.out.print(chr+" ");
88         if(chr == 'o') {
89             count++;
90         }
91     }
92     System.out.println();
93
94     System.out.println("o appears "+count+" times");

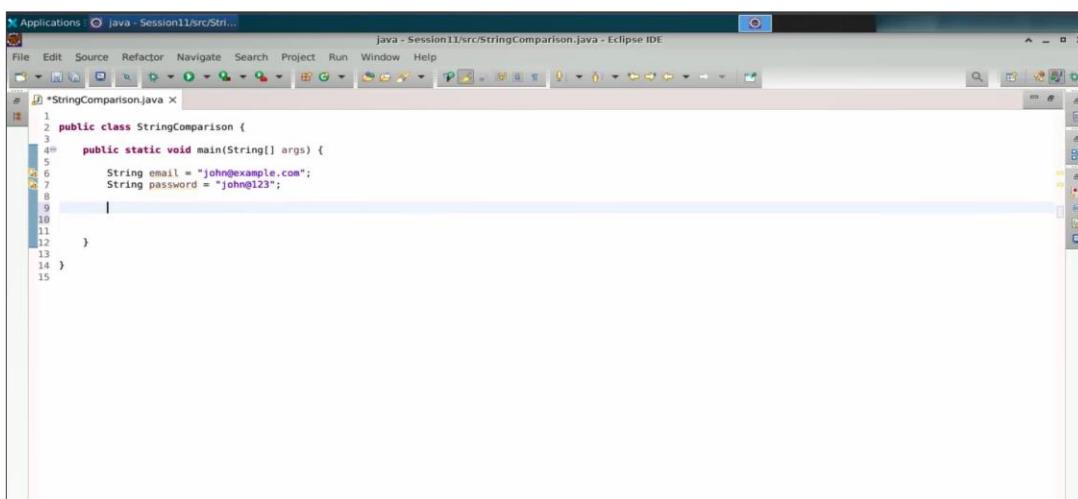
```

## Step 9: Implement String comparison and intern strings

9.1 Now the next thing is string comparison. It is an important concept, and you will create altogether one separate class on it. Let us name it as **StringComparison**



9.2 Let us say there is a string called email which is **john@example.com** and there is a string called password which is **John@123**. Now, these are the credentials, and you need to validate these credentials



9.3 You are going to create a Scanner class. Type **Scanner scanner = new Scanner(System.in);**

Then prompt the user to enter the email and password. Store the user email with **userEmail = scanner.nextLine();** and the user password with **userPassword = scanner.nextLine();**. Finally, close the scanner

```

Applications  java - Session11/src/StringComparison.java
File Edit Source Refactor Navigate Search Project Run Window Help
StringComparison.java X
1 import java.util.Scanner;
2
3 public class StringComparison {
4
5     public static void main(String[] args) {
6
7         String email = "john@example.com";
8         String password = "john@123";
9
10        Scanner scanner = new Scanner(System.in);
11        System.out.println("Enter Email: ");
12        String userEmail = scanner.nextLine();
13
14        System.out.println("Enter Password: ");
15        String userPassword = scanner.nextLine();
16
17        scanner.close();
18
19    }
20
21 }
22
23

```

9.4 Now, let us do a string comparison. If the email is equal to the user email and the password is equal to the user password, you are going to add a **System.out.printIn**. If the email and password match, Please Enter Home and if it does not, type Please Try Again

```

File Edit Source Refactor Navigate Search Project Run Window Help
StringComparison.java X
1 import java.util.Scanner;
2
3 public class StringComparison {
4
5     public static void main(String[] args) {
6
7         String email = "john@example.com";
8         String password = "john@123";
9
10        Scanner scanner = new Scanner(System.in);
11        System.out.println("Enter Email: ");
12        String userEmail = scanner.nextLine();
13
14        System.out.println("Enter Password: ");
15        String userPassword = scanner.nextLine();
16
17        scanner.close();
18
19        if(email == userEmail && password == userPassword) {
20            System.out.println("Email and Password Matched. Please Enter Home");
21        }else {
22            System.out.println("Email and Password Mismatched. Please Try Again");
23        }
24
25
26    }
27
28 }
29

```

9.5 Run the program and enter the email and password. It indicates a password mismatch even though you have entered the correct credentials. This issue arises because there are two different ways to create strings in Java. The first method involves creating an interned string. The other method creates a string object in memory, which can affect string comparison

```

 1 import java.util.Scanner;
 2
 3 public class StringComparison {
 4
 5     public static void main(String[] args) {
 6
 7         String email = "john@example.com";
 8         String password = "john@123";
 9
10         Scanner scanner = new Scanner(System.in);
11         System.out.println("Enter Email: ");
12         String userEmail = scanner.nextLine();
13
14         System.out.println("Enter Password: ");
15         String userPassword = scanner.nextLine();
16
17         scanner.close();
18
19         if(email == userEmail && password == userPassword) {
20             System.out.println("Email and Password Matched. Please Enter Home");
21         }else {
22             System.out.println("Email and Password MisMatched. Please Try Again");
23         }
24
25
26     }
27
28 }

```

9.6 Let us say my email is defined as a new string with the value **john@example.com**. Now, if you try to compare this email with my email, you will not get the correct result. Let us create another string called your email, which also has the value **john@example.com**. If you check whether 'email' is equal to my email, it will return false. If they match, you will print the Email matched. Otherwise, you will print Email mismatched

```

 1 import java.util.Scanner;
 2
 3 public class StringComparison {
 4
 5     public static void main(String[] args) {
 6
 7         String email = "john@example.com"; // Interned Strings
 8         String password = "john@123";
 9
10         String myEmail = new String("john@example.com"); // String Object
11         String yourEmail = "john@example.com";
12
13
14         if(email == myEmail) { // comparison on reference variables and not the data
15             System.out.println("MyEmail Matched");
16         }else {
17             System.out.println("MyEmail MisMatched");
18         }
19
20         if(email == yourEmail) { // comparing 2 interned Strings
21             System.out.println("YourEmail Matched");
22         }else {
23             System.out.println("YourEmail MisMatched");
24         }
25
26         Scanner scanner = new Scanner(System.in);
27         System.out.println("Enter Email: ");
28         String userEmail = scanner.nextLine();
29
30         System.out.println("Enter Password: ");
31         String userPassword = scanner.nextLine();
32
33         scanner.close();
34

```

9.7 Run the program. You get to see my email mismatched, but your email matched, so comparing the strings that are interned strings can be done through equal to operator. But not for string objects

```

StringComparison.java
1 import java.util.Scanner;
2
3 public class StringComparison {
4
5     public static void main(String[] args) {
6
7         String email = "john@example.com"; // Interned Strings
8         String password = "John@123";
9
10        String myEmail = new String("john@example.com"); // String Object
11        String yourEmail = "john@example.com";
12
13
14        if(email == myEmail) { // comparison on reference variables and not the data
15            System.out.println("MyEmail Matched");
16        } else {
17            System.out.println("MyEmail MisMatched");
18        }
19
20        if(email == yourEmail) { // comparing 2 interned Strings
21            System.out.println("YourEmail Matched");
22        } else {
23            System.out.println("YourEmail MisMatched");
24        }
25
26        Scanner scanner = new Scanner(System.in);
27        System.out.println("Enter Email: ");
28        String userEmail = scanner.nextLine();
29
30        System.out.println("Enter Password: ");
31        String userPassword = scanner.nextLine();
32
33        scanner.close();
34
35    }
36
37}

```

Console Output:

```

StringComparison [Java Application] /usr/eclipse/plugins/org.eclipse.jst.jdt.core/hotspot/jre/lib/client
MyEmail: MisMatched
YourEmail: Matched
Enter Email:

```

## Step 10: Use the method.equals method

10.1 If you want to compare the content, the equals operator will not work. You can use a method from the String class called equals. This method equals, will compare the content. After replacing the equals operator with the equals method, the equals method will return true or false

```

StringComparison.java
14     if(email == myEmail) { // comparison on reference variables and not the data
15         System.out.println("MyEmail Matched");
16     } else {
17         System.out.println("MyEmail MisMatched");
18     }
19
20     if(email == yourEmail) { // comparing 2 interned Strings
21         System.out.println("YourEmail Matched");
22     } else {
23         System.out.println("YourEmail MisMatched");
24     }
25
26     Scanner scanner = new Scanner(System.in);
27     System.out.println("Enter Email: ");
28     String userEmail = scanner.nextLine();
29
30     System.out.println("Enter Password: ");
31     String userPassword = scanner.nextLine();
32
33     scanner.close();
34
35     /*if(email == userEmail && password == userPassword) {
36         System.out.println("Email and Password Matched. Please Enter Home");
37     } else {
38         System.out.println("Email and Password MisMatched. Please Try Again");
39     }*/
40
41     if(email.equals(userEmail) && password.equals(userPassword)) {
42         System.out.println("Email and Password Matched. Please Enter Home");
43     } else {
44         System.out.println("Email and Password MisMatched. Please Try Again");
45     }
46
47}

```

## 10.2 Run the code and type **john@example.com** as the email and **John@123** as the password.

It shows that the email and password matched. Comparing strings is a risky operation. If you have string constants, such as interned strings where the references directly point to the literals, you can use the equals operator. However, when making string comparisons while reading from the console or some UI, you should not use the equals operator. Instead, you should work with the equals method

The screenshot shows the Eclipse IDE interface with the code editor displaying `StringComparison.java`. The code compares strings using the `==` operator. The output window shows the program's execution:

```

<terminated> StringComparison [Java Application] /usr/eclipse/plugins/org.eclipse.jst.jdt.core/
MyEmail Mismatched
YourEmail Matched
Enter Email:
john@example.com
Enter Password:
john@123
Email and Password Matched. Please Enter Home
  
```

## 10.3 Add equals ignore case, this will even ignore the case sensitivity

The screenshot shows the Eclipse IDE interface with the code editor displaying `StringComparison.java`. The code has been modified to use `equalsIgnoreCase` instead of `==` for string comparison. The output window shows the program's execution:

```

<terminated> StringComparison [Java Application] /usr/eclipse/plugins/org.eclipse.jst.jdt.core/
MyEmail Mismatched
YourEmail Matched
Enter Email:
john@example.com
Enter Password:
john@123
Email and Password Matched. Please Enter Home
  
```

10.4 Run the code and enter an email which is like **JOHN@example.com**, and the password is **JHON@123**, you can see it says email and password match

```

1 Applications : java - Session11/src/Str...
2 Java - Session11/src/StringComparison.java - Eclipse IDE
3
4 File Edit Source Refactor Navigate Search Project Run Window Help
5
6 StringComparison.java X
7
8 14     if(email == myEmail) { // comparison on reference variables and not the data
9 15         System.out.println("MyEmail Matched");
10 16     }else {
11 17         System.out.println("MyEmail MisMatched");
12 18     }
13
14     if(email == yourEmail) { // comparing 2 interned Strings
15         System.out.println("YourEmail Matched");
16     }else {
17         System.out.println("YourEmail MisMatched");
18     }
19
20     Scanner scanner = new Scanner(System.in);
21     System.out.println("Enter Email: ");
22     String userEmail = scanner.nextLine();
23
24     System.out.println("Enter Password: ");
25     String userPassword = scanner.nextLine();
26
27     scanner.close();
28
29     /*if(email == userEmail && password == userPassword) {
30         System.out.println("Email and Password Matched. Please Enter Home");
31     }else {
32         System.out.println("Email and Password MisMatched. Please Try Again");
33     }*/
34
35     /*if(email.equals(userEmail) && password.equals(userPassword)) {
36         System.out.println("Email and Password Matched. Please Enter Home");
37     }else {
38         System.out.println("Email and Password MisMatched. Please Try Again");
39     }*/
40
41     /*if(email.equalsIgnoreCase(userEmail) && password.equalsIgnoreCase(userPassword)) {
42         System.out.println("Email and Password Matched. Please Enter Home");
43     }else {
44         System.out.println("Email and Password MisMatched. Please Try Again");
45     }*/
46
47     if(email.equalsIgnoreCase(userEmail) && password.equalsIgnoreCase(userPassword)) {
48         System.out.println("Email and Password Matched. Please Enter Home");
49     }else {
50         System.out.println("Email and Password MisMatched. Please Try Again");
51     }
52
53     if(email.compareToIgnoreCase(userEmail) == 0 && password.compareTo(userPassword) == 0) {
54         System.out.println("Email and Password Matched. Please Enter Home");
55     }else {
56         System.out.println("Email and Password MisMatched. Please Try Again");
57     }

```

The console output shows:

```

<terminated> StringComparison [Java Application] /usr/eclipse/plugins/org.eclipse.justj.op
MyEmail MisMatched
YourEmail Matched
Enter Email:
JOHN@example.com
Enter Password:
JHON@123
Email and Password Matched. Please Enter Home

```

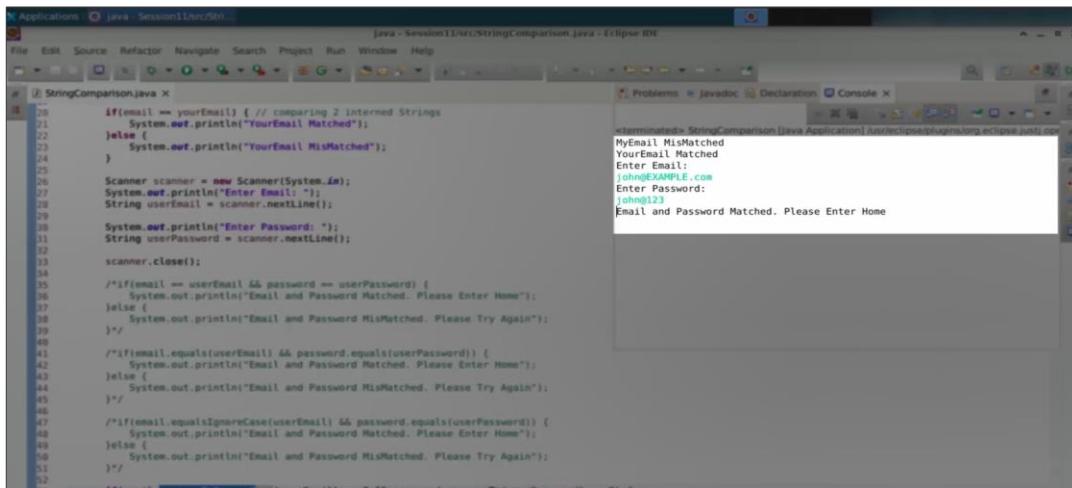
10.5 You have one more method to help with string comparison. This is **email.compareToIgnoreCase**, and it should return a value of zero. **compareTo** will compare with case sensitivity, whereas **compareToIgnoreCase** will ignore case sensitivity

```

1 Applications : java - Session11/src/Str...
2 Java - Session11/src/StringComparison.java - Eclipse IDE
3
4 File Edit Source Refactor Navigate Search Project Run Window Help
5
6 StringComparison.java X
7
8 14     if(email == yourEmail) { // comparing 2 interned Strings
9 15         System.out.println("YourEmail Matched");
10 16     }else {
11 17         System.out.println("YourEmail MisMatched");
12 18     }
13
14     Scanner scanner = new Scanner(System.in);
15     System.out.println("Enter Email: ");
16     String userEmail = scanner.nextLine();
17
18     System.out.println("Enter Password: ");
19     String userPassword = scanner.nextLine();
20
21     scanner.close();
22
23     /*if(email == userEmail && password == userPassword) {
24         System.out.println("Email and Password Matched. Please Enter Home");
25     }else {
26         System.out.println("Email and Password MisMatched. Please Try Again");
27     }*/
28
29     /*if(email.equals(userEmail) && password.equals(userPassword)) {
30         System.out.println("Email and Password Matched. Please Enter Home");
31     }else {
32         System.out.println("Email and Password MisMatched. Please Try Again");
33     }*/
34
35     /*if(email.equalsIgnoreCase(userEmail) && password.equalsIgnoreCase(userPassword)) {
36         System.out.println("Email and Password Matched. Please Enter Home");
37     }else {
38         System.out.println("Email and Password MisMatched. Please Try Again");
39     }*/
40
41     /*if(email.compareToIgnoreCase(userEmail) == 0 && password.compareTo(userPassword) == 0) {
42         System.out.println("Email and Password Matched. Please Enter Home");
43     }else {
44         System.out.println("Email and Password MisMatched. Please Try Again");
45     }*/
46
47     if(email.compareToIgnoreCase(userEmail) == 0 && password.compareTo(userPassword) == 0) {
48         System.out.println("Email and Password Matched. Please Enter Home");
49     }else {
50         System.out.println("Email and Password MisMatched. Please Try Again");
51     }
52
53     if(email.compareToIgnoreCase(userEmail) == 0 && password.compareTo(userPassword) == 0) {
54         System.out.println("Email and Password Matched. Please Enter Home");
55     }else {
56         System.out.println("Email and Password MisMatched. Please Try Again");
57     }

```

10.6 Run the code, here you are with the email shown as **John@EXAMPLE.com**, where the password is **John@1 2 3**. Password is case sensitive, and email can be the case insensitive



The screenshot shows the Eclipse IDE interface with the code editor and the Java perspective. The code editor displays `StringComparison.java` with the following content:

```
String email = "John@EXAMPLE.com";
String password = "John@1 2 3";
if(email == yourEmail) { // comparing 2 interned Strings
    System.out.println("YourEmail Matched");
} else {
    System.out.println("YourEmail MisMatched");
}
Scanner scanner = new Scanner(System.in);
System.out.print("Enter Email: ");
String userEmail = scanner.nextLine();
System.out.print("Enter Password: ");
String userPassword = scanner.nextLine();
scanner.close();
if(email == userEmail && password == userPassword) {
    System.out.println("Email and Password Matched. Please Enter Home");
} else {
    System.out.println("Email and Password MisMatched. Please Try Again");
}
/*if(email.equals(userEmail) && password.equals(userPassword)) {
    System.out.println("Email and Password Matched. Please Enter Home");
} else {
    System.out.println("Email and Password MisMatched. Please Try Again");
}
/*if(email.equalsIgnoreCase(userEmail) && password.equals(userPassword)) {
    System.out.println("Email and Password Matched. Please Enter Home");
} else {
    System.out.println("Email and Password MisMatched. Please Try Again");
}
```

The Java perspective shows the output of the program in the Console view:

```
MyEmail MisMatched
YourEmail Matched
Enter Email:
John@EXAMPLE.com
Enter Password:
John@1 2 3
Email and Password Matched. Please Enter Home
```

By following the above steps, you have successfully explored the various methods available with string data type in Java