L28

Friday, January 28, 2022 1:18 AM

Text processing Methods - length(), charAt(), toLowerCase(), toUpperCase(), startsWith(), endsWith(), equals()

String methods : length(), charAt(), toLowerCase(), toUpperCase()

Q1. The length of a string can be obtained using the method public int length() of the String class.

For Example:
"Pentagon".length() returns 8.
The characters in a string are indexed from 0, just like in an array. The method public char charAt(int index) returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the string is at index 0, the next at index 1, and so on, as for array indexing.

If we want to convert a string to lowercase, we should use public String toLowerCase() method. It returns a new string that has all the characters of the original string in lower case. Similarly, we have a public String toUpperCase() method for converting to upper case.

Retype the code below which demonstrates the usage of the above methods.

```
package q11151;
public class StringMethods {
    public static void main(String[] args) {
        String str = "FiveTimesFive";
        System.out.println("Length : " + str.length());
        System.out.println("5th character : " + str.charAt(4));
        System.out.println("Upper case : " + str.toUpperCase());
        System.out.println("Lower case : " + str.toLowerCase());
    }
}
```

Q2. Create a class TestStringMethods with a **main** method. The method receives one command line argument. Calculate the length of the argument and print the output.

For Example:
Cmd Args : Independance
12

```
package q11152;
public class TestStringMethods{
    public static void main(String[] args){

        System.out.println(args[0].length());
    }
}
```

Q3. Create a class TestStringMethods with a **main** method. The method receives one command line argument. Print the **5th** character in the argument.

For Example:
Cmd Args : Strange
n

```
package q11153;

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

        System.out.println(args[0].charAt(4));
    }
}
```

Q4. Create a class TestStringMethods with a **main** method. The method receives one command line argument. Convert the argument to **lowercase** and print the output.

For Example:
Cmd Args : KRISHNA
krishna

```
package q11162;

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

        System.out.println(args[0].toLowerCase());
    }
}
```

Q5. Create a class TestStringMethods with a **main** method. The method receives one command line argument. Convert the argument to **uppercase** and print the result.

For Example:
Cmd Args : India
INDIA

```
package q11163;

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

        System.out.println(args[0].toUpperCase());
    }
}
```

Q6. To check if two strings have the same content or not, we can use public boolean equals(Object anObject) method. The method returns true if and only if the argument is not null and is a String object that represents the same sequence of characters as the string.

To compare two strings ignoring the case, there is a public boolean equalsIgnoreCase(String aStr) method. The method returns true if and only if the argument is not null and is a String object that represents the same sequence of characters as the string, ignoring the case.

To check if a string starts with a certain prefix, the method public boolean startsWith(String prefix) can be used.

Similarly, we can use public boolean endsWith(String suffix) to check if a string ends with the specified suffix.

Retype the code below that shows the usage of the above methods.

```
package q11164;
public class StringMethods {
    public static void main(String[] args) {
        String str1 = "Jurassic";
        String str2 = "juraSSic";
        System.out.println("equals : " + str1.equals(str2));
        System.out.println("equalsIgnoreCase: " + str1.equalsIgnoreCase(str2));
        System.out.println("startsWith " + str1.startsWith("Jur"));
        System.out.println("endsWith " + str2.endsWith("Sic"));
    }
}
```

Q7. Write a class StringCompare with a **main** method. The method receives two command line arguments and prints true if both are equal.

For Example:
Cmd Args : India India
True

```
package q11165;

public class StringCompare{
    public static void main(String[] args){
        System.out.println(args[0].equals(args[1]));
    }
}
```

Q8. Write a class StringCompare with a **main** method. The method receives two command line arguments and prints true if both are equal.

Assumptions:
1. Ignoring the case of the two arguments

For Example:
Cmd Args : Congrats conGrats
true

```
package q11166;

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

        System.out.println(args[0].equalsIgnoreCase(args[1]));
    }
}
```

Q9. Write a class StringCompare witha a **main** method. The method receives one command line argument. Print **true** if the argument **starts with** the prefix **pre**, otherwise print **false**.

For Example:
Cmd Args : preparation
true

```
package q11167;


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

        System.out.println(args[0].startsWith("pre"));
    }
}
```

Q10. Write a class StringCompare with a **main** method. The method receives one command line argument. Print **true** if the argument **ends with** the suffix **ed**, else print **false**.

For Example:
Cmd Args : prefixed
true

```
package q11168;

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

        System.out.println(args[0].endsWith("ed"));
    }
}
```

Q11. Write a class StringCompare with a **main** the method. The method receives two command line arguments compare the first and second arguments using **equals()** method and also by using **==** operator and print the result.

For example :
Cmd Args : Ganga Ganga
The result with equals : true
The result with == : false

```
package q11169;
public class StringCompare {
    public static void main(String[] args) {
        System.out.println("The result with equals : " + args[0].equals(args[1]));
        System.out.println("The result with == : " + (args[0]==args[1]));
    }
}
```

Q12. The String class provides a method public int compareTo(String anotherString) to perform a lexicographical comparison of itself with another string passed to it.

It returns 0 (zero) if the strings are **equal**.
It returns a **positive number** (> 0), when the string appears before the string argument passed.
It returns a **negative number** (< 0), when the string appears after the string argument passed.

For example:
"Moon".compareTo("Moon") returns 0
"Earth".compareTo("Moon") returns -8
"Moon".compareTo("Earth") returns 8
Write a class StringCompareTo with a **main** method.

Assume the main method receives **two** string arguments. Write code to print the below output by comparing first argument and the second argument.
arg1 is lesser than arg2 *//if first argument lexicographically appears before the second argument* arg1 is greater than arg2 *//if first argument lexicographically appears after the second argument* arg1 and arg2 are equal*//if first argument is equal to the second argument*

```java
package q11170;

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

        int arg = args[0].compareTo(args[1]);

        if(arg > 0){
            System.out.println("arg1 is lesser than arg2");
        } else if(arg == 0){
            System.out.println("arg1 and arg2 are equal");
        } else if(arg < 0){
            System.out.println("arg1 is greater than arg2");
        }

    }
}
```