Unit 6
Handling Strings

Java Strings

- In Java, a string is a sequence of characters
- For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o’
- We use double quotes to represent a string in Java

  String language =“Java Programming”;
- Strings in Java are not primitive types (like int, char, etc.)
- Instead, all strings are objects of a predefined class named String.
- And, all string variables are instances of the String class

  String language = new String(“Java Programming”);

Java String Operations

❖ Java String provides various methods to perform different operations on strings.

1. length() method
   - To find the length of a String, we can use length() method of String.
2. concat() method
   - We can join/concatenate two strings in Java using the concat() method.
3. equals() method
   - We can make comparison between two strings in Java using equals() method.
4. equalsIgnoreCase()
   - It converts both the string to same case (i.e. either both on upper case or both on lower case) and compares
5. charAt()
   - We can obtain the character at a specified position of the string

Java String Operations

6. indexOf()
   - This method returns the index a character present in a string
7. contains()
   - This method is used to check if a string value contains specified character or not.
8. endsWith()
   - This method returns true if the specified string ends with specified character sequence.
9. replace()
   - This method is used to replace one string value with another
10. toLowerCase()
    - Converts a String to lowercase
11. toUpperCase()
    - Converts a String to uppercase

Java String Operations-Example

```java
public class StringOperations {
    public static void main(String[] args) {
        String firstString = "Hello World";
        String secondString= "How are you?";
        String thirdString ="hello world";
        System.out.println("Length of firstString is "+ firstString.length());
        String concatinatedString = firstString.concat(secondString);
        System.out.println("Concatinated String is "+ concatinatedString);
        System.out.println("First String equals to second? "+firstString.equals(secondString));
        System.out.println("First String equals to third? "+firstString.equals(thirdString));
        System.out.println("First String equals to third if ignored ignore case? "+firstString.equalsIgnoreCase(thirdString));
        System.out.println("Character of firstString at position 7 is "+firstString.charAt(7));
        System.out.println("Index of char W in firstString is "+ firstString.indexOf('W'));
        System.out.println("Does firstString contains 'Hello'? "+ firstString.contains("Hello"));
        System.out.println("Does firstString ends with 'World'? "+firstString.endsWith("World"));
        String replacedString = firstString.replace("World", "Hello");
        System.out.println("Replace World of first string with 'Hello' "+ replacedString);
        System.out.println("Upper case of firstString is "+ firstString.toUpperCase());
        System.out.println("Lower case of firstString is "+ firstString.toLowerCase());
    }
}
```

Java String Operations-Example-Output

Output - StringOperations (run)

```
run:
Length of firstString is 11
Concatinated String is Hello WorldHow are you?
First String equals to second? false
First String equals to third? false
First String equals to third if ignored ignore case? true
Character of firstString at position 7 is o
Index of char W in firstString is 6
Does firstString contains 'Hello'? true
Does firstString ends with 'World'? true
Replace World of first string with 'Hello' Hello Hello
Upper case of firstString is HELLO WORLD
Lower case of firstString is hello world
BUILD SUCCESSFUL (total time: 0 seconds)
```

More on Java String Operations

1. substring()
   - A part of String is called substring. In other words, substring is a subset of another String.
   - Syntax: stringVariable.substring(startIndex, endIndex);
   - Note: In case of Java String, *startIndex is inclusive but endIndex is exclusive*

Example:

```
public class StringOperations {
    public static void main(String[] args) {
        String firstString = "Application";
        System.out.println("Substring of firstString is "+ firstString.substring(3, 7));
    }
}
```

```
Output - StringOperations (run)
run:
Substring of firstString is lica
BUILD SUCCESSFUL (total time: 0 seconds)
```

More on Java String Operations

2. trim() method

- The trim() method removes whitespace from both directions of string

```
public class StringOperations {
    public static void main(String[] args) {
        String firstString = " Application ";
        System.out.println("Trimmec String is"+ firstString.trim());
    }
}
```

```
Output - StringOperations (run)
run:
Trimmec String isApplication
BUILD SUCCESSFUL (total time: 0 seconds)
```

More on Java String Operations

3. toCharArray() method

❖ This method converts a string value to character array.

```java
public class StringOperations {
    public static void main(String[] args) {
        String firstString = " Application ";
        char[] characterArray = firstString.toCharArray();
        for(int i=0; i<characterArray.length;i++){
            System.out.println(characterArray[i]);
        }
    }
}
```

Output - StringOperations (run)

```
run:

A
p
p
l
i
c
a
t
i
o
n

BUILD SUCCESSFUL (total time: 0 seconds)
```

More on Java String Operations

4. Splitting a string

- The Java String split() method divides the string at the specified regex and returns an array of substrings.
- The string split() method can take two parameters :
  - regex - the string is divided at this regex (can be strings)
  - limit (optional) - controls the number of resulting substrings

```java
public class StringOperations {
    public static void main(String[] args) {
        String thirdSemSubject ="Java,WebTechnology,DSA";
        String[] subjectArray = thirdSemSubject.split(",");
        for(int i=0; i<subjectArray.length;i++){
            System.out.println(subjectArray[i]);
        }

    }
```

```
Output - StringOperations (run)
run:
Java
WebTechnology
DSA
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Strings are Immutable

- Java String is an example of immutable type.
- Once we create a string, we cannot change that string i.e. String object always represents the same thing.
- **String is immutable but we have StringBuilder class in java whose objects are mutable.**

Java String Buffer

- **StringBuffer** is a peer class of **String** that provides much of the functionality of strings.
- The string represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.
- **StringBuffer** may have characters and substrings inserted in the middle or appended to the end.
- **Important Constructors of StringBuffer class**
  1. StringBuffer(): creates an empty string buffer with the initial capacity of 16.
  2. StringBuffer(String str): creates a string buffer with the specified string.
  3. StringBuffer(int capacity): creates an empty string buffer with the specified capacity as length.

Java String Buffer- commonly used methods

1. append() method
   - The append() method concatenates the given argument with this string.
2. insert() method
   - The insert() method inserts the given string with this string at the given position.
3. replace() method
   - The replace() method replaces the given string from the specified beginIndex and endIndex-1.
4. delete() method
   - The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex-1.
5. reverse() method
   - The reverse() method of StringBuilder class reverses the current string.
6. capacity() method
   - The capacity() method of StringBuffer class returns the current capacity of the buffer.

Java String Buffer- commonly used methods

```java
package stringbuffer;
public class StringBufferExample {
    public static void main(String[] args) {
      StringBuffer sb = new StringBuffer("Hello");
       sb.append("World");
       System.out.println("Value after append is "+sb);
       sb.insert(5, "Java");
       System.out.println("Value after insert is "+sb);
       sb.replace(5,8,"BCA");
       System.out.println("Value after replace is "+ sb);
       sb.delete(5, 9);
       System.out.println("Value after delete is "+ sb);
       sb.reverse();
        System.out.println("Value after reverse is "+sb);
    }
}
```

Output - StringBufferExample (run)

```
run:
Value after append is HelloWorld
Value after insert is HelloJavaWorld
Value after replace is HelloBCAaWorld
Value after delete is HelloWorld
Value after reverse is dlroWolleH
BUILD SUCCESSFUL (total time: 0 seconds)
```