Java String

Lecture - 6

Tokey Ahmmed

Lecturer, Dept. of CSE

Varendra University

tokey@vu.edu.bd

Strings

- Java string is a sequence of characters. They are objects of type String.
- Once a String object is created it cannot be changed. **Stings are Immutable**.
- The default constructor creates an empty string.

```
String s = new String();
```

Creating Strings

- `String str = "abc";` equivalent to:

```
char[] data = {'a', 'b', 'c’};
String str = new String(data);
```

- If data array in the above example is modified after the string object **`str`** is created, then **`str`** remains unchanged.
- Construct a string object by passing another string object.

```
String str2 = new String(str);
```

Example:

```
public class StringDemo{

public static void main(String args[]){

        char[] helloArray ={'h','e','l','l','o','.’};

        String helloString =new String(helloArray);

        System.out.println(helloString);

    }
}
```

String Operations

+ operator

- The **+ operator** is used to concatenate two or more strings.

  Eg:

```
String myname = “Tokey”
String str = “My name is” + myname+ “.”;
```

- For string concatenation the Java compiler converts an operand to a String whenever the other operand of the + is a String object.

length()

- The **length()** method returns the length of the string.

  Eg:

```
System.out.println("Hello".length());
```

// prints 5

charAt()

- Characters in a string can be extracted in a number of ways.
- **public char charAt(int index)**
  - Returns the character at the specified index. An index ranges from 0 to length() - 1. The first character of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

```
char ch;
ch = “abc”.charAt(1);
```

// ch = “b”

Example:

```
public class Test{
    public static void main(String args[]){
        String s ="Strings are immutable";
        char result = s.charAt(8);
        System.out.println(result);
    }
}
```

compareTo()

- **compareTo()** - Compares two strings lexicographically.
  - Each character of both the strings is converted into a Unicode value for comparison.
  - If both the strings are equal then this method returns 0 else it returns positive or negative value.
  - The result is positive if the first string is lexicographically greater than the second string else the result would be negative.

Example:

```
public static void main(String args[]) {
        String str1 ="Strings are immutable";
        String str2 ="Strings are immutable";
        String str3 ="Integers are not immutable";
        int result = str1.compareTo(str2);
        System.out.println(result);
        result = str2.compareTo(str3);
        System.out.println(result);
}
```

Output:
0
10

concat()

- This **concat()** method appends one String to the end of another

Syntax:

```
public String concat(String s)
```

Example:

```
String s ="Strings are immutable";
s = s.concat(" all the time");
System.out.println(s);
```

Output:

Strings are immutable all the time

copyValueOf(char[] data)

- Syntax:

```
public staticString copyValueOf(char[] data)
```

or

```
public staticString copyValueOf(char[] data, int offset, int count)
```

Parameters:

- data -- the character array. offset -- initial offset of the subarray. count -- length of the subarray.

Example:

```
public class Test{
public static void main(String args[]){
        char[ ] Str1={'h','e','l','l','o',' ','w','o','r','l','d'};
        String Str2="";
        Str2=Str2.copyValueOf(Str1);
        System.out.println("Returned String: "+Str2);
        Str2=Str2.copyValueOf(Str1,2,6);
        System.out.println("Returned String: "+Str2);
    }
}
```

Output:

Returned String: hello world
Returned String: llo wo

startsWith(), endsWith()

- **startsWith()** – Tests if this string starts with the specified prefix.

```
public boolean startsWith(String prefix)
```

```
“Figure”.startsWith(“Fig”);
```
// true

- **endsWith()** - Tests if this string ends with the specified suffix.

```
public boolean endsWith(String suffix)
```

```
“Figure”.endsWith(“re”);
```
// true

Example:

```
String Str = new String("This is really not immutable!!");
boolean retVal;
retVal =Str.endsWith("immutable!!");
System.out.println("Returned Value = “ + retVal );
retVal =Str.startsWith("immutable!!");
System.out.println("Returned Value = "+ retVal );
```

Output:
Returned Value = true
Returned Value = false

equals()

- **equals()** - Compares the invoking string to the specified object. The result is true if and only if the argument is not null and is a String object that represents the same sequence of characters as the invoking object.

public boolean **equals**(Object anObject)

Example:

```
String Str1=new String("This is really not immutable!!");
String Str2=Str1;
String Str3=new String("This is really immutable!!");
boolean retVal;
retVal =Str1.equals(Str2);
System.out.println("Returned Value = "+ retVal );
retVal =Str1.equals(Str3);
System.out.println("Returned Value = "+ retVal );
```

Output:
Returned Value = true
Returned Value = false

replace(), replaceAll()

- **replace()-** Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

  public String **replace**(char oldChar, char newChar)

  Example:

```
String str = "How was your day today?";
System.out.println(str.replace('o','T'));
```

Output:
HTw was yTur day tTday?

- **replaceAll() -**

  String replaceAll(String regex, String replacement)

Example:

```
String str = "How was your day today?";
System.out.println(str.replaceAll(str,“Varendra University"));
```

Output:
Varendra University

indexOf()

- **indexOf()**– Searches for the **first occurrence** of a character or substring. Returns -1 if the character does not occur.

  - `public int indexOf(char ch)` - Returns the index within this string of the first occurrence of the specified character.
  - `public int indexOf(String str)` - Returns the index within this string of the first occurrence of the specified substring.

Example:

```
String str = ("How was your day today?");
System.out.println(str.indexOf('a'));
System.out.println(str.indexOf("was"));
```

Output:
5
4

lastIndexOf()

- **lastIndexOf()** –Searches for the last occurrence of a character or substring. The methods are similar to indexOf().

```
String str = ("How was your day today?");
System.out.println(str.lastIndexOf('a'));
```

Output:
20

substring()

- **substring()** - Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

  - `public String substring(int beginIndex)`

    Eg: `"unhappy".substring(2)` returns "happy"

  - `public String substring(int beginIndex, int endIndex)`

    Eg: `"smiles".substring(1, 5)` returns "mile“

trim()

- **trim()** - Returns a copy of the string, with leading and trailing whitespace omitted.
  public String trim()

```
String s = “  Hi Mom!  ”.trim();
```

S = “HiMom!”

Other String Operations

- **toLowerCase()** : Converts all of the characters in a String to lower case.
- **toUpperCase()** : Converts all of the characters in this String to upper case.
- **toString()** : This method returns itself a string

Example:

```
“HELLO THERE”.toLowerCase();
“hello there”.toUpperCase();
“hello there”. toString();
```