Programming in Java

String Handling

CODERINDEED

Introduction

- Every string we create is actually an object of type String.
- String constants are actually String objects.
- Example:

System.out.println("This is a String, too");

String Constant

- Objects of type String are immutable i.e. once a String object is created, its contents cannot be altered.
- Because String objects are immutable, whenever we want to modify a String, it will construct a new copy of the string with modifications.

Introduction

- In java, four predefined classes are provided that either represent strings or provide functionality to manipulate them. Those classes are:
  - String
  - StringBuffer
  - StringBuilder
  - *StringTokenizer*

- String, StringBuffer, and StringBuilder classes are defined in java.lang package and all are final.

- All of them implement the CharSequence interface.

Declaring and creating string

To represent a string of characters, use the data type called String. For example, the following code declares message to be a string with the value "Welcome to Java".

```
String message = "Welcome to Java";
```

String is a predefined class in the Java library, just like the classes System and Scanner. The String type is not a primitive type. It is known as a reference type. Any Java class can be used as a reference type for a variable. The variable declared by a reference type is known as a reference variable that references an object. Here, message is a reference variable that references a string object with contents Welcome to Java.

Different ways of creating strings:

There are two ways to create string in Java:

String literal

String s = “Hello”;

Using new keyword

String s = new String (“Hello”);

Simple Methods for String object

Method	*Description*
`length()`	Returns the number of characters in this string.
`charAt(index)`	Returns the character at the specified index from this string.
`concat(s1)`	Returns a new string that concatenates this string with string s1.
`toUpperCase()`	Returns a new string with all letters in uppercase.
`toLowerCase()`	Returns a new string with all letters in lowercase
`trim()`	Returns a new string with whitespace characters trimmed on both sides.

Examples

```
class Example
{
        public static void main(String[] args)
        {
        String s="Hello World";
        System.out.println("Length of the string s is "+ s.length());
        System.out.println("Character at position 4 is "+ s.charAt(4));
        String s1=" Welcome to java";
        System.out.println("String after joining of s and s1"+ s.concat(s1));
        System.out.println("String in upper case letters"+ s.toUpperCase());
        System.out.println("String in lower case letters"+ s.toLowerCase());
        String s2=" Hello ";
        System.out.println("String s2 after trimming white spaces from both ends "+s2.trim());

    }
}
```

Output:

```
Length of the string s is 11
Character at position 4 is o
String after joining of s and s1: Hello World Welcome to java
String in upper case letters: HELLO WORLD
String in lower case letters: hello world
String s2 after trimming white spaces from both ends Hello
```

Reading a String

- Two methods can be used.
- next()
- nextLine()

- next() method is used to take input of string that ends with a whitespace character.
- nextLine() You can use the nextLine() method to read an entire line of text. The nextLine() method reads a string that ends with the Enter key pressed. For example, the following statements read a line of text.

Example:

```
//next() method
import java.util.Scanner;
public class Main
{
  public static void main (String[]args)
  {
    Scanner input = new Scanner (System.in);
    System.out.print ("Enter three words separated by spaces: ");
    String s1 = input.next ();
    String s2 = input.next ();
    String s3 = input.next ();
    System.out.println ("s1 is " + s1);
    System.out.println ("s2 is " + s2);
    System.out.println ("s3 is " + s3);
  }
}
```

Output:

```
Enter three words separated by spaces: Hi Hello Bye    //user input
s1 is Hi
s2 is Hello
s3 is Bye
```

Example:

nextLine():

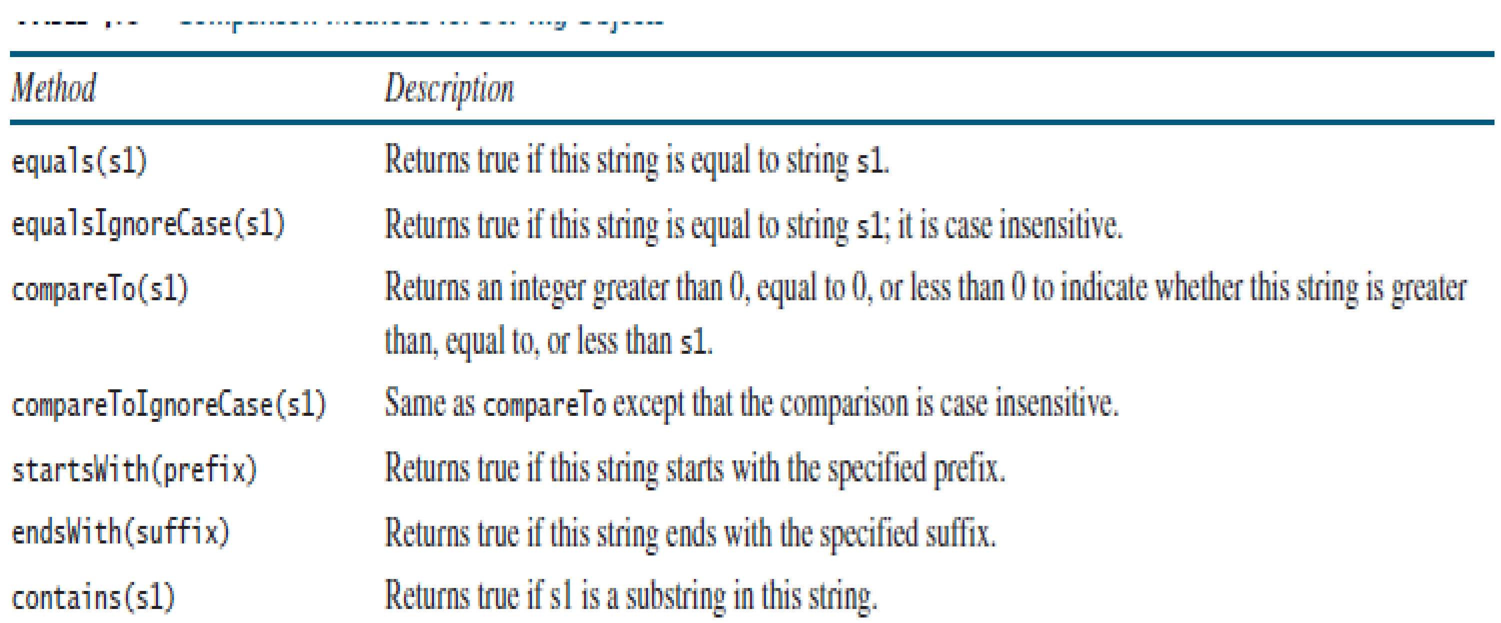
```
import java.util.Scanner;
public class Main
{
  public static void main (String[]args)
  {

    Scanner input = new Scanner (System.in);
    System.out.println ("Enter a line: ");
    String s = input.nextLine ();
    System.out.println ("The line entered is " + s);
  }
}
```

Output:

```
Enter a line:
Hello this is one string   //user input
The line entered is Hello this is one string
```

Comparing Strings



[illegible]

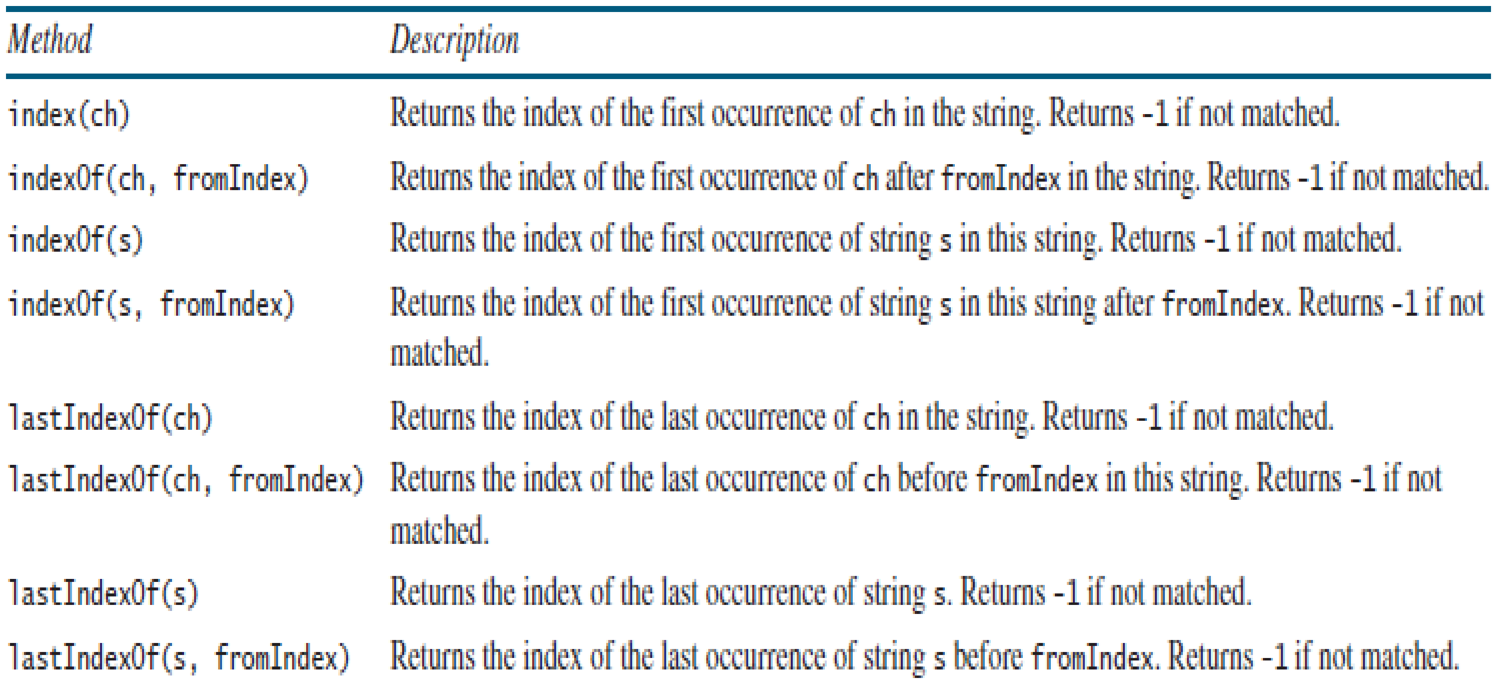
Method	*Description*
`equals(s1)`	Returns true if this string is equal to string `s1`.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string `s1`; it is case insensitive.
`compareTo(s1)`	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than `s1`.
`compareToIgnoreCase(s1)`	Same as `compareTo` except that the comparison is case insensitive.
`startsWith(prefix)`	Returns true if this string starts with the specified prefix.
`endsWith(suffix)`	Returns true if this string ends with the specified suffix.
`contains(s1)`	Returns true if s1 is a substring in this string.

Examples

```
class Example
{
		public static void main(String[] args)
		{
				String s1="Hello World";
				String s2="Hello World";
				String s3="Welcome to java";
				System.out.println(s1.equals(s2));// true
				System.out.println(s1.equals(s3));// false
				System.out.println(s1.compareTo(s3));// value less than 0
				System.out.println(s1.startsWith("H"));// true
				System.out.println(s3.startsWith("H"));// false
				System.out.println(s1.endsWith("d"));// true
				System.out.println(s3.contains("to"));// true
				System.out.println(s1.contains("to"));// false

		}
}
```

Methods for finding substrings/or characters in a given string



Method	*Description*
`index(ch)`	Returns the index of the first occurrence of `ch` in the string. Returns `-1` if not matched.
`indexOf(ch, fromIndex)`	Returns the index of the first occurrence of `ch` after `fromIndex` in the string. Returns `-1` if not matched.
`indexOf(s)`	Returns the index of the first occurrence of string `s` in this string. Returns `-1` if not matched.
`indexOf(s, fromIndex)`	Returns the index of the first occurrence of string `s` in this string after `fromIndex`. Returns `-1` if not matched.
`lastIndexOf(ch)`	Returns the index of the last occurrence of `ch` in the string. Returns `-1` if not matched.
`lastIndexOf(ch, fromIndex)`	Returns the index of the last occurrence of `ch` before `fromIndex` in this string. Returns `-1` if not matched.
`lastIndexOf(s)`	Returns the index of the last occurrence of string `s`. Returns `-1` if not matched.
`lastIndexOf(s, fromIndex)`	Returns the index of the last occurrence of string `s` before `fromIndex`. Returns `-1` if not matched.

The first method is indexOf(ch)----->Misprinted as index(ch)

Example

```
public class Main
{
  public static void main (String[]args)
  {
      String s = "Welcome to Java";
      System.out.println (s.indexOf ('W'));                // returns 0.
      System.out.println (s.indexOf ('o'));  // returns 4.
      System.out.println (s.indexOf ('o', 5));             // returns 9.
      System.out.println (s.indexOf ("come"));          // returns 3.
      System.out.println (s.indexOf ("Java", 5));       // returns 11.
      System.out.println (s.indexOf ("java", 5));       // returns -1.
      System.out.println (s.lastIndexOf ('W'));         // returns 0.
      System.out.println (s.lastIndexOf ('o'));         // returns 9.
      System.out.println (s.lastIndexOf ('o', 5));      // returns 4.
      System.out.println (s.lastIndexOf ("come"));// returns 3.
      System.out.println (s.lastIndexOf ("Java", 5));// returns -1.
      System.out.println (s.lastIndexOf ("Java"));      // returns 11.
  }
}
```

Extracting a substring from a given string

- substring(): used to extract a part of a string.

public String substring (int start_index)

public String substring (int start_index, int end_index)

Example: String s = “ABCDEFG”;

String t = s.substring(2); System.out.println (t);//CDEFG

String u = s.substring (1, 4); System.out.println (u);//BCD

Note: Substring from start_index to end_index-1 will be returned.

replace( ): The replace( ) method has two forms.

- The first replaces all occurrences of one character in the invoking string with another character. It has the following general form:

  String replace(char original, char replacement)

- Here, original specifies the character to be replaced by the character specified by replacement.

Example: String s = "Hello".replace('l', 'w');//All occurances of l will be replaced with w and s will take reference of object with value:Hewwo

- The second form of replace( ) replaces one character sequence with another. It has this general form:

  String replace(CharSequence original, CharSequence replacement)

Example:
String s = "This is java class".replace("java", "Python"); System.out.println(s);
Output: This is Python class

Q1(Output)??

```
import java.util.Scanner;
public class Main
{
  public static void main (String[]args)
  {
    String s=" Test ";
    System.out.print(s.length()+”,”);
    String s1=s.trim();
    System.out.print(s1.length());
  }
}
```

A. 6 6

B. 6 4

C. 4 4

D. 6 5

Q2(Output??)

```
import java.util.Scanner;
public class Main
{
  public static void main (String[]args)
  {
    String s1="Polling";
    String s2="Question";
    String s3=s1.concat(s2);
    System.out.println(s3.charAt(8));
  }
}
```

A. Q
B. u
C. Runtime error
D. g

Q3(Output??)

```
import java.util.Scanner;
public class Main
{
  public static void main (String[]args)
  {
    String s1="Hello";
    String s2="Halogen";
    System.out.println(s1.compareTo(s2));
  }
}
```

A. 5

B. 4

C. -4

D. 0

Q4(Output??)

```
public class Main
{
		public static void main(String[] args) {
				String s1="TESTING";
				String s2="testing";
System.out.println(s1.compareToIgnoreCase(s2))
;
		}
}
```

A. 0
B. -1
C. 1
D. false

Q5(Output??)

```
public class Main
{
        public static void main(String[] args) {
                String s1="This is the test phase";

        System.out.println(s1.lastIndexOf('t',11));
        }
}
```

A. 8
B. 12
C. -1
D. 7

Q6(Output??)

```
public class Main
{
        public static void main(String[] args) {
                String s1="Best among the
Best";

        System.out.println(s1.indexOf("Best"
));
        }
}
```

A. 0
B. 15
C. -1
D. Error

Q7(Output??)

```
public class Main
{
            public static void main(String[] args)
{
                        String s1="Programming
Skills";
            System.out.println(s1.substring(3,7))
;
            }
}
```

A. grammin
B. gram
C. gramm
D. ogram