L27

Friday, January 28, 2022 1:18 AM

Text processing – String Class, String concatenation

String Class

Q1. String literals and String class

A string literal consists of zero or more characters (also called Unicode code points) enclosed in double quotes. Some examples of string literals are

"Ganga" , "123", "", " ", "Practice makes perfect", " 24@3yApp_"

String literals are instances of java.lang.String class.

An instance of String is immutable. Meaning, once a string object is created in memory, the contents of the string cannot be changed. For example, when a string is created as given below :

String str = "Ganga";

The contents of "Ganga" cannot be changed. However, str can be made to point to some other string object like "Yamuna".

Below program creates some String literals. Retype the program and submit.

```java
package q11144;
public class StringLiterals {
    public static void main(String[] args) {
        String aStr = "Himalayas";
        String bStr = "are beautiful";
        String spaceStr = " ";
        String emptyStr = "";//This is an empty string that does nothing.
        System.out.println(aStr + spaceStr + bStr + "." + emptyStr);
    }
}
```

Q2. String class

A string object can also be created using any of the constructors of the String class. For example:

```
String aStr = new String("Lemuria");
char charArr[] = {'A', 't', 'l', 'a', 'n', 't', 'i', 's'};
String anotherStr = new String(charArr);
```

Instances of String class are **immutable**, meaning their values cannot be changed after they are created. If any alteration has to be made to a string instance, a new string object with the required content must be created.

In the below example, you will find the correct way of comparing two strings:

```
String aString = "abc";
String bString = new String("abc");
System.out.println(aString == bString); //This prints false
System.out.println(aString.equals(bString)); //This prints true
```

In the above code, when we use == comparison operator, we are comparing the address stored in aString with the address stored in bString. Since bString is created using the constructor, it creates a new object which has similar content as that of aString. Since bString points to a new object, its address is different from the address stored in aString. Hence, when we compare addresses, it returns false.

There is a equals method in String class which is used to compare the contents stored in two string objects. This returns true if the contents are the same and false if they differ. In our example, since the contents are same, we get true as the output.

Hence, the correct way of comparing contents of two strings is to use equals method.

See and reproduce the below code.

```java
package q11145;
public class StringLiterals {
    public static void main(String[] args) {
        String str1 = "Encyclopedia";
        String str2 = new String("Encyclopedia");
        System.out.println("str1 == str2 : " + (str1 == str2));
        System.out.println("str1.equals(str2) : " + (str1.equals(str2)));
    }
}
```

Escape Characters

Q1. Escape sequences
Since a string literal is enclosed in double quotes, if we want to have a literal with a double quote as part of its content, we need to escape the double quote.

The way to escape a double quote is by providing the escape character \ before the double quote.

For example:

String text = "He said, \"Hi\"";
Escape character is also used to represent non-graphic characters like the new line(\n) and tab(\t) characters.

See and retype the code below:

```
package q11146;
public class EscapeSequence {
    public static void main(String[] args) {
        String textStr = "She said, \"It is wrongly called 'common sense' because it is rarely used\". \nWhat a wise saying!";
        System.out.println(textStr);
        char singleQuoteChar = '\'';
        System.out.println("Single Quote Char = " + singleQuoteChar);
        char backSlashChar = '\\';
        System.out.println("Back Slash Char = " + backSlashChar);
    }
}
```

Q2. Understanding Escape sequences

Fix the text in the **main** method using an escape character to produce the below output.

There is a single double-quote before "BLUE

```
package q11147;
public class EscapeSequence {
    public static void main(String[] args) {
        String text = "There is a single double-quote before \"BLUE";
        System.out.println(text);
    }
}
```

Q3. Fix the text in the **main** method using escape characters to produce the below output.

There are double-quotes around "Red"

```
package q11148;
public class EscapeSequence {
    public static void main(String[] args) {
        String text1 = "There are double-quotes around \"Red\"";
        System.out.println(text1);
    }
}
```

Q4. Fix the text in the **main** method using escape characters to produce the below output.

insert into Employee(id, name) values("10","Meka");
insert into Employee(id, name) values("11","Reka");

```
package q11149;
public class EscapeSequence {
    public static void main(String[] args) {
        String text1 = "insert into Employee(id, name) values(\"10\",\"Meka\");";
        String text2 = "insert into Employee(id, name) values(\"11\",\"Reka\");";
        System.out.println(text1);
        System.out.println(text2);
    }
}
```

String Concatenation

Q1. Strings can be concatenated using the + operator.

For example:
String helloStr = "Hello " + "World";
If one operand in the expression is of type String and the other operand is not a String, the latter is converted to a string by automatically invoking the toString() method on it.

The result of string concatenation is a reference to a new String object that is created by merging the contents of the two strings.

Some examples of string concatenation are give below:

"Hello " + "World"	Hello World
"The square root of 2 is " + Math.sqrt(2)	The square root of 2 is 1.4142135623730952
3 + 2 + " monkeys"	5 monkeys
"" + 3 + 2 + " monkeys"	32 monkeys
"monkeys " + 3 + 2	monkeys 32

The + operator is syntactically left-associative, i.e. it evaluates the left side of an expression first.

For example, it evaluates the expression a + b + c as (a + b) + c. As soon as it sees a string in one of the operands, from that point onwards, it converts all the remaining operands as strings.

That is why in the example 3 + 2 + " monkeys", first 3 + 2 is evaluated to 5 using integer addition and later the expression becomes 5 + " monkeys", where 5 is converted to String and concatenated to " monkeys".

Retype and submit the code below. Observe the output.

```
1  package q11150;
2  public class StringConcatenation {
3      public static void main(String[] args) {
4          System.out.println("Hello!" + " I am learning Java.");
5          System.out.println("Total = " + 3 + 2);
6          System.out.println("Total = " + (3 + 2));
7      }
8  }
```