Chapter 7 Strings

Prerequisites for Part II

---

The String Class

- Constructing a String:
  - String message = "Welcome to Java";
  - String message = new String("Welcome to Java");
  - String s = new String();
- Obtaining String length and Retrieving Individual Characters in a string String
- String Concatenation (concat)
- Substrings (substring(index), substring(start, end))
- Comparisons (equals, compareTo)
- String Conversions
- Finding a Character or a Substring in a String
- Conversions between Strings and Arrays
- Converting Characters and Numeric Values to Strings

---

Constructing Strings

String newString = new String(stringLiteral);

String message = new String("Welcome to Java");

Since strings are used frequently, Java provides a shorthand initializer for creating a string:

String message = "Welcome to Java";
Canonical Strings
Since strings are immutable, to improve efficiency and save memory, the JVM stores two String objects in the same object if they were created with the same string literal using the shorthand initializer. Such a string is referred to as a canonical string. You can also use a String object’s intern method to return a canonical string, which is the same string that is created using the shorthand initializer.

Examples
String s = "Welcome to Java";
String s1 = new String("Welcome to Java");
String s2 = s1.intern();
String s3 = "Welcome to Java";
System.out.println("s1 == s is " + (s1 == s));
System.out.println("s2 == s is " + (s2 == s));
System.out.println("s == s3 is " + (s == s3));
display
s1 == s is false
s2 == s is true
s == s3 is true

Finding String Length
Finding string length using the length() method:
message = "Welcome";
message.length() (returns 7)

Retrieving Individual Characters in a String
Do not use message[0]
Use message.charAt(index)
Index starts from 0

String Concatenation
String s3 = s1.concat(s2);
String s3 = s1 + s2;
s1 + s2 + s3 + s4 + s5 same as
((s1.concat(s2)).concat(s3)).concat(s4)).concat(s5);

Extracting Substrings
String is an immutable class; its values cannot be changed individually.
String s1 = "Welcome to Java";
String s2 = s1.substring(0, 11) + "HTML";

message.substring(0, 11)
message.substring(11)
String Comparisons

```java
String s1 = new String("Welcome");
String s2 = "welcome";
if (s1.equals(s2)){
    // s1 and s2 have the same contents
}
if (s1 == s2) {
    // s1 and s2 have the same reference
}
```

String Comparisons, cont.

```java
String s1 = new String("Welcome");
String s2 = "welcome";
if (s1.compareTo(s2) > 0) {
    // s1 is greater than s2
} else if (s1.compareTo(s2) == 0) {
    // s1 and s2 have the same contents
} else {
    // s1 is less than s2
}
```

String Conversions

The contents of a string cannot be changed once the string is created. But you can convert a string to a new string using the following methods:

- toLowerCase
- toUpperCase
- trim
- replace(oldChar, newChar)

Finding a Character or a Substring in a String

"Welcome to Java".indexOf('W') returns 0.
"Welcome to Java".indexOf('x') returns -1.
"Welcome to Java".indexOf('o', 5) returns 9.
"Welcome to Java".indexOf("come") returns 3.
"Welcome to Java".indexOf("Java", 5) returns 11.
"Welcome to Java".indexOf("java", 5) returns -1.
"Welcome to Java".lastIndexOf('a') returns 14.

Convert Character and Numbers to Strings

The String class provides several static valueOf methods for converting a character, an array of characters, and numeric values to strings. These methods have the same name valueOf with different argument types char, char[], double, long, int, and float. For example, to convert a double value to a string, use String.valueOf(5.44). The return value is string consists of characters ‘5’, ‘.’, ‘4’, and ‘4’.

Example 7.1
Finding Palindromes

Objective: Checking whether a string is a palindrome: a string that reads the same forward and backward.

CheckPalindrome
### The Character Class

<table>
<thead>
<tr>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character(value: char)</td>
</tr>
<tr>
<td>charValue(): char</td>
</tr>
<tr>
<td>compareTo(anotherCharacter: Character): boolean</td>
</tr>
<tr>
<td>equals(anotherCharacter: Character): boolean</td>
</tr>
<tr>
<td>isUpperCase(ch: char): boolean</td>
</tr>
<tr>
<td>isLowerCase(ch: char): boolean</td>
</tr>
<tr>
<td>isLetterOrDigit(ch: char): boolean</td>
</tr>
<tr>
<td>isLetter(ch: char): boolean</td>
</tr>
<tr>
<td>isDigit(ch: char): boolean</td>
</tr>
<tr>
<td>equals(anotherCharacter: Character): boolean</td>
</tr>
<tr>
<td>compareTo(anotherCharacter: Character): int</td>
</tr>
<tr>
<td>charValue(): char</td>
</tr>
<tr>
<td>Character(value: char)</td>
</tr>
</tbody>
</table>

Constructs a character object with char value
Returns the char value from this object
Compares this character with another
Returns true if this character equals to another
Returns true if the specified character is a digit
Returns true if the specified character is a letter
Returns true if the character is a lowercase letter
Returns true if the character is an uppercase letter
Returns the lowercase of the specified character
Returns the uppercase of the specified character

### Examples

Character charObject = new Character('b');

charObject.compareTo(new Character('a')) returns 1
charObject.compareTo(new Character('b')) returns 0
charObject.compareTo(new Character('c')) returns -1
charObject.compareTo(new Character('d')) returns -2
charObject.equals(new Character('b')) returns true
charObject.equals(new Character('d')) returns false

### Example 7.2 Counting Each Letter in a String

This example gives a program that counts the number of occurrence of each letter in a string. Assume the letters are not case-sensitive.

#### CountEachLetter

StringBuffer

- StringBuffer() Constructs an empty string buffer with capacity 16
- StringBuffer(int length) Constructs a string buffer with the specified capacity
- append(aPrimitiveType): StringBuffer Appends a value converted to string into this buffer
- append(charArray): StringBuffer Appends a char array into this string buffer
- append(String): StringBuffer Appends a string to this string buffer
- deleteCharAt(int index) Deletes a character at the specified index
- delete(startIndex: int, endIndex: int) Deletes characters from startIndex to endIndex
- length(): int Returns the number of characters in this buffer
- replace(int startIndex, int endIndex, String str) Replaces the characters in this buffer from startIndex to endIndex with the specified string
- reverse(): StringBuffer Reverses the characters in this buffer
- setCharAt(index: int, ch: char): void Sets a new character at the specified index
- setLength(newLength: int): void Sets a new length in this buffer
- substring(startIndex: int, endIndex: int): String Returns a substring from startIndex to endIndex
- substring(startIndex: int): String Returns a substring starting at startIndex
- String Buffer(charArray: char[], offset: int, len: int): StringBuffer Append a subarray in data into this string buffer
- String Buffer(charArray: char[], offset: int, len: int): StringBuffer Append a subarray of the data in the array to the buffer at the specified index
- String Buffer(charArray: char[], offset: int, len: int): StringBuffer Append a subarray of the data in the array to the buffer at the specified index
- String Buffer(str: String): StringBuffer Constructs a string buffer with the specified string
- String Buffer(StringBuffer): StringBuffer Returns a substring from startIndex to endIndex
- String Buffer(charArray: char[], offset: int, len: int): StringBuffer Append a subarray of the data in the array to the buffer at the specified index
- String Buffer(str: String): StringBuffer Constructs a string buffer with the specified string
- String Buffer(): StringBuffer Constructs a string buffer with the specified capacity
- String Buffer(): StringBuffer Constructs a string buffer with the specified capacity
- String Buffer(): StringBuffer Constructs an empty string buffer with capacity 16
- String Buffer(int length): StringBuffer Constructs a string buffer with the specified capacity
- String Buffer(int length): StringBuffer Constructs a string buffer with the specified capacity
Appending New Contents into a StringBuffer

```java
StringBuffer strBuf = new StringBuffer();
strBuf.append("Welcome");
strBuf.append(' ');
strBuf.append("to");
strBuf.append(' ');
strBuf.append("Java");
```

Example 7.3
Checking Palindromes Ignoring Non-alphanumeric Characters

This example gives a program that counts the number of occurrence of each letter in a string. Assume the letters are not case-sensitive.

```java
PalindromeIgnoreNonAlphanumeric
```

The StringTokenizer Class

```java
java.util.StringTokenizer
+StringTokenizers(String)
+StringTokenizers(String, delimiters: String)
+StringTokenizers(String, delimiters: String, returnDelimiters: boolean)
+countTokens(): int
+hasMoreTokens(): boolean
+nextToken(): String
+nextToken(delimiters: String): String

Constructs a string tokenizer for the string.
Constructs a string tokenizer for the string with the specified delimiters.
Constructs a string tokenizer for the string with the delimiters and returnDelims.
Returns the number of remaining tokens.
Returns true if there are more tokens left.
Returns the next token.
Returns the next token using new delimiters.
```

Examples 1

```java
String s = "Java is cool.";
StringTokenizer tokenizer = new StringTokenizer(s);
System.out.println("The total number of tokens is " + tokenizer.countTokens());
while (tokenizer.hasMoreTokens())
    System.out.println(tokenizer.nextToken());
The code displays
```

```
The total number of tokens is 3
Java
is
cool.
```

Examples 2

```java
String s = "Java is cool.";
StringTokenizer tokenizer = new StringTokenizer(s, "ae");
System.out.println("The total number of tokens is " + tokenizer.countTokens());
while (tokenizer.hasMoreTokens())
    System.out.println(tokenizer.nextToken());
The code displays
```

```
The total number of tokens is 7
J
a
v
is
cool.
```

Examples 3

```java
String s = "Java is cool.";
StringTokenizer tokenizer = new StringTokenizer(s, "ae", true);
System.out.println("The total number of tokens is " + tokenizer.countTokens());
while (tokenizer.hasMoreTokens())
    System.out.println(tokenizer.nextToken());
The code displays
```

```
The total number of tokens is 7
J
a
v
a
is
cool.
```
No no-arg Constructor in StringTokenizer

The StringTokenizer class does not have a no-arg constructor. Normally it is a good programming practice to provide a no-arg constructor for each class. On rare occasions, however, a no-arg constructor does not make sense. StringTokenizer is such an example. A StringTokenizer object must be created for a string, which should be passed as an argument from a constructor.

The Scanner Class

The delimiters are single characters in StringTokenizer. You can use the new JDK 1.5 java.util.Scanner class to specify a word as a delimiter.

String s = "Welcome to Java! Java is fun! Java is cool!";
Scanner scanner = new Scanner(s);
scanner.useDelimiter("Java");
while (scanner.hasNext())
    System.out.println(scanner.next());

Welcome to Java!
is fun!
is cool!

Output

Scanning Primitive Type Values

If a token is a primitive data type value, you can use the methods nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble(), or nextBoolean() to obtain it. For example, the following code adds all numbers in the string. Note that the delimiter is space by default.

String s = "1 2 3 4";
Scanner scanner = new Scanner(s);
int sum = 0;
while (scanner.hasNext())
    sum += scanner.nextInt();
System.out.println("Sum is "+sum);

Console Input Using Scanner

Another important application of the Scanner class is to read input from the console. For example, the following code reads an int value from the keyboard:

System.out.print("Please enter an int value: ");
Scanner scanner = new Scanner(System.in);
int i = scanner.nextInt();

NOTE: StringTokenizer can specify several single characters as delimiters. Scanner can use a single character or a word as the delimiter. So, if you need to scan a string with multiple single characters as delimiters, use StringTokenizer. If you need to use a word as the delimiter, use Scanner.

Command-Line Parameters

class TestMain {
    public static void main(String[] args) {
        ...
    }
}

java TestMain arg0 arg1 arg2 ... argn

Processing Command-Line Parameters

In the main method, get the arguments from args[0], args[1], ..., args[n], which corresponds to arg0, arg1, ..., argn in the command line.
Example 7.4
Using Command-Line Parameters

Objective: Write a program that will perform binary operations on integers. The program receives three parameters: an operator and two integers.

<table>
<thead>
<tr>
<th>Calculator</th>
<th>java Calculator 2 + 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>java Calculator 2 - 3</td>
</tr>
<tr>
<td></td>
<td>java Calculator 2 / 3</td>
</tr>
<tr>
<td></td>
<td>java Calculator 2 **= 3</td>
</tr>
</tbody>
</table>