Strings & StringBuffers

Introduction

This unit covers the String and StringBuffer objects.

Strings

Up to this point we have used Strings similar to how we use primitives, but we will learn more about what makes them different.

The Immutable String

First off, Strings variables are **immutable,** meaning they cannot be changed. You may think that this is not true and you have changed the value of a String.

Code That Makes Us Think Strings can change:

```
String s = “Bob”;
System.out.println(s);
s = “Joe”;
System.out.println(s);
```

Output:

```
Bob
Joe
```

In the above code variable s stores “Bob” and then is changed to hold “Joe”. Remember from unit 3 that setting a String variable to text is a shortcut for what is actually happing.

Code without the Shortcut:

```
String s = new String(“Bob”);
System.out.println(s);
s = new String(“Joe”);
System.out.println(s);
```

Output:

```
Bob
Joe
```

s is not a String, it is a container that stores the memory address of a String object. Initially s stores an address of String object that has the text “Bob”. Next s is changed to store the address of a new String object that stores the text “Joe”. “Bob” was never changed; the memory address that variable s held was changed to the address of a different String object.

String Methods

Strings are objects meaning they have methods that can be called on them to preform operations. The below chart contains a list of the String methods will we be using:

Method	Description	Example
charAt(int x)	Returns the char at the given index.	String s = "Bob likes cheese"; System.out.println(s.charAt(2)); **Output:** b
compareTo(String s)	Returns an int value that describes how far apart the first letter that is different between the calling and received string is. • Negative – The received string is first alphabetically • Zero – They are the same. • Negative – The calling string is first alphabetically. Note: 'a' is different from 'A' 'A' has an int value of 65 'a' has an int value of 97	String s = "apples"; String t = "Apples"; String u = "apples"; String v = "bear" System.out.println(s.compareTo(t)); System.out.println(t.compareTo(s)); System.out.println(s.compareTo(u)); System.out.println(s.compareTo(v)); **Output:** 32 -32 0 -1
contains(String s)	Returns true when the received String exists in the calling String and false otherwise.	String a = "jojo"; String b = "joe"; String c = "joj"; System.out.println(a.contains(b)); System.out.println(a.contains(c)); **Output:** false true
endsWith(String s)	Returns true when the received String is at the end of the calling String and false otherwise.	String a = "jojo"; String b = "jo"; String c = "joj"; System.out.println(a.endsWith(b)); System.out.println(a.endsWith(c)); **Output:** true false
equals(String s)	Returns true when the received String and the calling String are the same and false otherwise.	String s = "apples"; String t = "Apples"; String u = "apples"; System.out.println(s.compareTo(t)); System.out.println(s.compareTo(u)); **Output:** false true

equalsIgnoreCase(String s)	Returns true when the received String and the calling String are the same and false otherwise. This method treated uppercase and lowercase as the same.	String s = “apples”; String t = “Apples”; String u = “apples”; System.out.println(s.equalsIgnoreCase (t)); System.out.println(s.equalsIgnoreCase (u)); **Output:** true true
indexOf(char c)	Returns the 1st index of the received char. -1 if not found.	String s = “apples”; System.out.println(s.indexOf(‘x’)); System.out.println(s.indexOf(‘p’)); **Output:** -1 1
indexOf(char c, int fromIndex)	Returns the 1st index of the received char, starting from the given index. -1 if not found.	String s = “bob”; System.out.println(s.indexOf(‘b’)); System.out.println(s.indexOf(‘b’,1)); **Output:** 0 2
indexOf(String s)	Returns the 1st index of the received String. -1 if not found.	String s = “apples”; System.out.println(s.indexOf(“pl”)); System.out.println(s.indexOf(“esa”)); **Output:** 2 -1
indexOf(String s, int fromIndex)	Returns the 1st index of the received String, starting from the given index. -1 if not found.	String s = “apples”; System.out.println(s.indexOf(“pl”)); System.out.println(s.indexOf(“pl”,3)); **Output:** 2 -1
length()	Returns the number of characters in the String	String s = “apples”; System.out.println(s.length()); **Output:** 6
split(String s)	Returns a String array of the calling String created by using the received String as a spacer	String s = “aa*cc*bb”; String[] broken = s.split(“*”); System.out.println(broken[0]); System.out.println(broken[1]); System.out.println(broken[2]); **Output:** aa bb cc

startsWith(String s)	Returns true if the calling String starts with the received String.	String a = “jojo”; String b = “jo”; String c = “oj”; System.out.println(a.startsWith(b)); System.out.println(a.startsWith(c)); **Output:** true false
substring(int start)	Returns a String created from the calling String starting at the given index and going all the way to the end of the calling String.	String s = “apples”; System.out.println(s.substring(2)); **Output:** ples
substring(int start, int end)	Returns a String created from the calling String starting at the given index and at the index before end.	String s = “apples”; System.out.println(s.substring(1,4)); **Output:** ppl
toCharArray()	Returns a charAt that contains the same letters as the String	String s = “turtles”; char[] letters = s.toCharArray(); System.out.println(letters[2]); **Output:** r
toLowerCase()	Returns a new String that is an all lowercase version of the calling String.	String s = “tUrtLes”; System.out.println(s.toLowerCase()); **Output:** turtles
toUpperCase()	Returns a new String that is an all uppercase version of the calling String.	String s = “tUrtLes”; System.out.println(s.toUpperCase()); **Output:** TURTLES

StringBuffer

A StringBuffer is similar to a String in the fact that stores text data, but many of its methods change the data it stores.

Creating a StringBuffer

Format:

StringBuffer name = new StringBuffer(“text”);

Example 1:

StringBuffer company = new StringBuffer(“Microsoft”);

Example 2:

String app = “Apple”;
StringBuffer otherCompany = new StringBuffer(app);

String Methods

StringBuffers have some of the same methods as String and many new methods that change the data the StringBuffer stores.

Method	Description	Example
StringBuffer delete(int start, int end)	Deletes the text in the range [start,end). Returns the new StringBuffer in addition to changing it.	StringBuffer text = new StringBuffer("abcdefghi"); text.delete(1,3); System.out.println(text); **Output:** Adefghi

StringBuffer deleteCharAt(int index)	Deletes the char value at the given index.
	
	Returns the new StringBuffer in addition to changing it.
		StringBuffer text = new StringBuffer("abcdefghi");
		text.deleteCharAt(0);
		text.deleteCharAt(3);
		
		System.out.println(text);
		
		Output:
		bcdfghi

StringBuffer insert(int offset,String s)	Inserts the given String at the given index. Returns the new StringBuffer in addition to changing it. **Note: There are also different insert methods for each primitive type too. Example insert(int offset, boolean b)**	StringBuffer text = new StringBuffer("abcdefghi"); text.insert(2,"***"); System.out.println(text); **Output:** ab***cdefghi
StringBuffer replace(int start, int end, String s)	Replaces the text in the range [Start,end) with the given String. Returns the new StringBuffer in addition to changing it.	StringBuffer text = new StringBuffer("abcdefghi"); text.replace(2,6,"***"); System.out.println(text); **Output:** ab***ghi
StringBuffer reverse()	Reverses the text. Returns the new StringBuffer in addition to changing it.	StringBuffer text = new StringBuffer("abcdefghi"); text.reverse(); System.out.println(text); **Output:** ihgfedcba
void setCharAt(int index, char c)	Changes the char value at the given index to the given char value.	StringBuffer text = new StringBuffer("abcdefghi"); text.setCharAt(3,'Z'); System.out.println(text); **Output:** abcZefghi

Blue Pelican Section

Lesson 2
Lesson 3

Terms

Immutable	Cannot be changed.

Assignments

String Practice 1
String Practice 2
StringBuffer Practice