Lecture 7: Strings Revisited: String and StringBuilder classes

Based on Introduction to Java Programming, Y. Daniel Liang, Brief Version, 10/E

Topics Covered

1. The String class
   1.1 Immutable Objects
   1.2 String comparison
   1.3 Accessing parts of a string object and combining strings
   1.4 Strings and Other Data Types

2. Case Study: Parsing Strings

3. Command Line Arguments: the secret behind main(String [] args)

4. StringBuilder Class
String - a sequence of characters in Java represented as an object, in other languages might be represented as an array or characters.

Several classes in Java libraries to represent strings:

- String
- StringBuilder

1 The String class

For documentation see: [http://docs.oracle.com/javase/7/docs/api/java/lang/String.html](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html).

There are many different ways of constructing a String object, i.e., there are many constructors. For example:

```java
String s1 = "Hello world!";
String s2 = new String ("Hello NY!");
String s3 = new String (s2);
char [] letters = {'H', 'e', 'l', 'l', 'o', ' ', 'N', 'Y', 'U', '!'};
String s4 = new String (letters);
```

1.1 Immutable Objects

What happens when we execute the following lines of code?

1) String s1 = "I like C++";
2) String s2 = s1;
3) System.out.println(s1 + "\n" + s2);
4) s1 = "I like Java";
5) System.out.println(s1 + "\n" + s2);

String objects are examples of immutable objects. Immutable objects cannot be modified once they are created.

See ImmutableStrings.java

1.2 String comparison

Comparison methods:

```java
equals( ... )
compareTo( ... )
equalsIgnoreCase ( ... )
compareToIgnoreCase ( ... )
```
1.3 Accessing parts of a string object and combining strings

Methods worth knowing:

- `length (... )`
- `charAt (... )`
- `concat (... )`
- `toLowerCase (... )`
- `toUpperCase (... )`
- `split (... )`
- `replace (... )`

1.4 Strings and Other Data Types

- `toCharArray ( )` returns character array that contains all letters of the String objects
- `valueOf ( TYPE v )` (used with class name `String`) returns a String object representing the variable v. TYPE can be: `char`, `char []`, `double`, `float`, `int`, `long`, `boolean`
- `format ( FORMAT_SPECIFIER_AS_IN_PRINTF)` (used with class name `String`) returns String object containing formatted string

2 Case Study: Parsing Strings

See `StockQuote.java` and `StockQuoteApp.java`
These program connect to Yahoo Finance to access the up to date stock prices.

3 Command Line Arguments:

- The secret behind `main(String [] args)`

The arguments from the command line are passed to the `main()` method via the `String` array `args`.
See `Echo.java`

4 StringBuilder Class

`StringBuilder` class allows for more flexibility than the `String` class. Unlike `String` objects, `StringBuilder` objects are mutable.
So why use `String` instead of `StringBuilder`? **SPACE!** `StringBuilder` allocates extra space to make it possible to add things to the string. When the allocated object runs out of space, new space needs to be allocated.

See [http://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html](http://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html).

**Constructors:**

`StringBuilder()` Constructs a string builder with no characters in it and an initial capacity of 16 characters.

`StringBuilder(int capacity)` Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.

`StringBuilder(String str)` Constructs a string builder initialized to the contents of the specified string.

**Selected methods:**

`StringBuilder append(TYPE v)` Appends the string representation of the variable/object `v` to the sequence.

`int capacity()` Returns the current capacity (NOTE: this is not the length of the stored string).

`char charAt(int index)` Returns the char value in this sequence at the specified index.

`StringBuilder insert(int offset, char c)` Inserts the string representation of the char argument into this sequence. (There are many other versions of the `insert` method.)

`int length()` Returns the length (character count).

`StringBuilder replace(int start, int end, String str)` Replaces the characters in a substring of this sequence with characters in the specified `String`.

`StringBuilder reverse()` Causes this character sequence to be replaced by the reverse of the sequence.

`String substring(int start, int end)` Returns a new `String` that contains a subsequence of characters currently contained in this sequence.

`String toString()` Returns a string representing the data in this sequence.

See `PalindromeIgnoreNonAlphanumeric.java` and `ImmutableString.java` for examples of use of the `StringBuilder` class.