String Class, Output and Static Methods

CSCI 161 – Introduction to Programming I

Professor Thomas Rogers

Overview

- Related reading: Section 3.3
- Topics:
  - String class
  - Output
  - Identifiers and Keywords
  - Static Methods
  - Pitfalls
  - Procedural Decomposition

String Class

- Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.
- The most direct way to create a string is to write
  String greeting = "Hello world!";
  Note: Strings are literals within double quotes and cannot span multiple lines.
- Methods used to obtain information about an object are known as **accessor methods**. One accessor method that you can use with strings is the **length()** method, which returns the number of characters contained in the string object.

String Class (continued)

- You can concatenate strings with the + operator, or the **.concat()** method:

```
String string1 = "Hey";
String string2 = " now";
string1.concat(string2);
System.out.println(string1); // Maybe not what is expected
string1 = string1.concat(string2);
System.out.println(string1); // Interesting!
```

- Strings are immutable, meaning they don't actually change. The only way to change them is to re-assign them which causes a new object and new String in memory.

- Compare and contrast that to **StringBuffer** which is mutable and allows for in-place updating of a string:

```
StringBuffer b = new StringBuffer("Hello");
System.out.println(b); b.append(", world!");
System.out.println(b);
```

String Class (continued)

- One of the really cool methods of the String class is **format():**

  ```
  String fs; fs = String.format("The value of the
  float variable is " + "%f, while the value of
  the integer " + "variable is %d, and the string
  " + "is %s", floatVar, intVar, stringVar);
  System.out.println(fs);
  ```

- **toUpperCase()** - a method that uppercases the string.

- **indexOf()** - a method that determines if a specific character is contained withing the string.

- A great overview of the String class along with its various methods here: https://www.tutorialspoint.com/java/java_strings.htm

Output

- Output is done through the **System** class.
- The **System** class contains several useful class fields and methods. It cannot be instantiated. Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.
- **System.out** has **println** and **print** methods for output to standard output (aka "the console").

Output (continued)

- **println** - Used to output a single line of text.
- **print** - Used to output data to the current line without including a new line. Repeated calls will place output on the same line.
- Escape sequences:
  - \t - tab character
  - \n - new line character
  - \" - quotation mark (double quote)
  - \\ - backslash character

Identifiers and Keywords

- **identifier** - a name given to an entity in a program such as a class, methor or variable.
- Can start with a letter followed by any letter or number.
- Can also include an underscore _ or dollar sign $.
- Cannot include plus sign +, minu sign/hyphen -, space or start with a number.
- Conventions:
  - **Class names** - Start with a capital, then lowercase. e.g. Product
  - **Method names** - start with lower case then capitalize the first letter of each new word (aka camel case). e.g. addProduct
  - **Constant names** - All upper case with underscores between words. e.g. MAX_LENGTH
  - **Variable names** - Camel Case, with our without data type prefix. (more on that later). e.g. intTotalCost, or totalCost.

Identifiers and Keywords (continued)

- Java Keywords (aka Reserved Words):

  abstract default goto package this assert do
  if private throw boolean double implements
  protected throws break else import public
  transient byte enum instanceof return true
  case extends int short try catch false
  interface static void char final long strictfp
  volatile. class finally native super while const
  float new switch continue for null
  synchronized

Static Methods

- A static method of a class can be called without an instance of the class existing.
- For example:

```
System.out.println("Hello"); // println is a
static method
System.out.println("Length of the word car is: "
+ "car".length()); // length method of String
class not static (is an instance method)
```

- For our first few labs you will deal exclusively with static methods.
- *More on static methods vs instance methods in future lectures.*

Pitfalls

- Three general types of programming errors:
  - **Syntax Errors** - The programming equivalent of bad grammar. These are caught by the Java compiler.
  - **Runtime Errors** - An error in your program that occurs while it is running that is so severe the Java VM stops your program from executing. An example is dividing by zero. These errors may often be avoided with proper error handling.
  - **Logic Errors (aka "bugs" or "defects")** - Your code is syntactically correct and it runs without error but does not perform the tasks you intend it to. These errors are often much harder to find.

Pitfalls (continued)

- **Syntax Errors** - a list of common syntax errors are:
  - File name does not match class name.
  - Misspellings - a misspelled class, method or variable name, or even a mis-match in case.
  - Missing semicolon.
  - Forgetting a required keyword: (static on a method, method return type, etc).
  - Not closing a String Literal or Comment.

Procedural Decomposition

- **Procedural Decomposition** - A separation into discernible parts, each of which is simpler than the whole.
- **Cake** example:
  - Make the batter
  - Bake the cake
  - Make the frosting
  - Frost the cake
- However, making the batter may require sub-tasks:
  - Mix the dry ingredients
  - Cream the butter and sugar
  - Beat the eggs, beat eggs into creamed mixture
  - Stir dry ingredients into wet mixture

Procedural Decomposition (continued)

- In Java, the easiest way to separate a problem into parts is through the use of methods.
- **Iterative Enhancement** - The process of producing a program in stages, adding new functionality at each stage. A key feature is that you test as you go.
- * Some materials from www.tutorialspoint.com.