CSCE 155 - Java

Lab 9.0 - Strings

Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.
2. Review Oracle's documentation about Strings in Java: http://docs.oracle.com/javase/tutorial/java/data/strings.html

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- Understand the way Java implements and handles Strings
- Use methods of String and understand the difference between classes `String` and `StringBuilder`

2 Background

A string in Java is a collection of characters regarded and handled as a single unit. The `String` class is used to represent strings, so they are objects not primitive types. Like other objects, strings have their own set of constructors and methods. When used with methods, strings are passed by reference not by value. However, unlike most other objects Strings can be instantiated without using the `new` operator. For example, the following lines of code are all valid string declarations:

```
1 String myString = "This is the string lab for CSE 155";
2 String myOtherString = new String( "This is the string lab for CSE 155" );
3 String anotherString = null;
4 String emptyString = "";
```

The reference variable `myString` refers to a string literal and `anotherString` is a reference that points to `null`. However, it can still be used to eventually reference a valid `String` object. The reference variable `emptyString` does *not* point to `null`, it is the empty string, and simply points to a `String` object that has no characters. In Java strings are immutable. That is, once created their contents cannot be changed. `StringBuilder` is a `String`-like class that provides methods to change character contents.

See the Java API for a full description of the `String` and `StringBuilder` classes:

- http://docs.oracle.com/javase/6/docs/api/java/lang/String.html.
- http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/StringBuilder.html

3 Activities

This lab will familiarize you with some of these concepts. In particular, you will complete a program that implements a common children?s game, horse (also known as hangman).

In this game, an English word is chosen at random and its characters hidden. The player takes turns by guessing a letter; each instance (if any) of the guessed letter is revealed. If the user is able to guess the word before a certain number of guessed letters then they win. If they run out of guesses then they lose.

Most of the game mechanics have been implemented for you. However, you will need to complete the game by implementing several methods used by the game to manipulate and compare Strings. Clone the project for this lab from GitHub using the following URL: https://github.com/cbourke/CSCE155-Java-Lab09.

3.1 Implementing String Manipulation Methods

1. Examine the content of `ObtainInput.java` and `PlayHorse.java` but do not change them.
2. Open `HorseGame.java` in Eclipse. There are several methods already fully implemented in this file. Your task for this lab is to implement the following four methods:
   - `initializeBlankString()` - This method takes one variable as input: a `String` variable containing the secret word. It should return nothing. The method should alter the `StringBuilder` instance so that it is of equal length to the secret word and set all of its values to an underscore. The `StringBuilder` instance will slowly reveal correctly guessed letters
   - `printWithSpaces()` - This method will print the contents of the `StringBuilder` instance with spaces between each character. The method requires no input and should return nothing.
   - `revealGuessedLetter()` - This method will take a `String` (the secret word) and a character (the user's guess) as input. The method should alter the `StringBuilder` instance in the following way: for every position in the secret word that contains the character passed in as the second argument, change the same position in the `StringBuilder` instance to that character. For example, if the secret word is: `"dinosaur"` and the `StringBuilder` variable is: `"________"` and the character passed is: `a` the method should alter the `StringBuilder` so that it becomes `"_____a__"`. The method should return `true` if any letters were changed and `false` otherwise.
   - `checkGuess()` - This method checks if the `StringBuilder` instance is the same as the secret word string. The method should return `true` if they are, and `false` otherwise.
3. Complete the implementation of the methods, and run the program in Eclipse.

4. Answer the questions on your worksheet.

4 Handin/Grader Instructions

1. Hand in your completed files:
   - `HorseGame.java`
   - `ObtainInput.java`
   - `PlayHorse.java`
   - `worksheet.md`

   through the webhandin (https://cse-apps.unl.edu/handin) using your cse login and password.
2. Even if you worked with a partner, you *both* should turn in all files.
3. Verify your program by grading yourself through the webgrader (https://cse.unl.edu/~cse155h/grade/) using the same credentials.
4. Recall that both expected output and your program's output will be displayed. The formatting may differ slightly which is fine. As long as your program successfully compiles, runs and outputs the *same values*, it is considered correct.

5 Advanced Activity (Optional)

If strings are immutable, and their contents cannot be changed, why does the following code execute as expected?

```
1 String myString = "This week we?re studying Strings!";
2 System.out.println("myString contains: " + myString);
3 myString = "Next week we?re studying file I/O!";
4 System.out.println("myString now contains: " + myString);
```

Did the contents of `myString` change? Suppose we add another line of code as follows:

```
System.out.println( "myString now contains: "+ myString.toUpperCase() );
```

What did the above line of code do? Did it also change the contents of `myString`?