Lab 10: Strings and Text Files

Objectives:

1. To do some string processing.
2. To learn how to read text files. You will find this useful for Assignment 4.
3. To learn how to parse a line of text. You will find this useful for Assignment 4 also.

Preparation:

1. Go over the Lecture Notes Topic 12
2. Textbook reading (optional): Chapter 12 Section 12.2

Exercise 1: Using some String methods

a) Type the following sequence of Java statements in the Interactions Pane:

```
String firstName = "harry";
System.out.println(firstName.toUpperCase());
System.out.println(firstName);
```

Does the `toUpperCase` method change the string on which it is invoked?
Strings are called ***immutable*** because once they are created, they are not changed; a new string is created and returned when a string manipulation method is invoked on a String object.

b) Enter the following sequence of Java statements in the Interactions Pane:

```
String test = "hello";
System.out.println(test.charAt(0));
System.out.println(test.charAt(3));
System.out.println(test.charAt(4));
```

What are the results? What would you type to print out the letter `e` from the test string?

c) The method `length()` of the String class returns the number of characters in a string. (Note that this is different from the `length` that we used with arrays.). Complete the simple program below that prints all the characters of a test string, using the `charAt()` method and the `length()` method. (Hint: see Topic 12). Compile and run it.

```
public class PlayWithStrings
{
     public static void main (String[] args)
    {
        String test = "This is a test.";
        /* insert your code here */

     }
}
```

d) Add code to the program of part c) so that it will also print all the characters of the test string *backwards*, again using the `charAt()` method and the `length()`method. (Hint: start your `for` loop at the last character in the string)

Exercise 2: String equality

The String method `equals` determines whether two strings are the same, and returns the boolean value `true` if they are the same and `false` if they are not.

a) Type the following variable declarations in the Interactions pane:

```
String str1 ="Hello";
String str2 ="Goodbye";
String str3 ="Goodbye";
System.out.println(str1.equals(str2));
```

b) Then complete and type in the following if-else statement such that the correct message is printed when comparing str1 to str2:

```
if  ( // fill in the condition here)
  System.out.println("The strings are the same")
else
  System.out.println("The strings are different");
```

c) Now complete and type in the following if-else statement such that the correct message is printed when comparing str2 to str3:

```
if  ( // fill in the condition here)
  System.out.println("The strings are different")
else
  System.out.println("The strings are the same");
```

Exercise 3: Reading text from a file

In this exercise, you will learn how to read lines of text (i.e. strings) from a file into memory. You will use a class `SimpleReader` which we have developed, that provides methods for reading text from a file. (You will find this useful for Assignment 4.)

a) Download the file *SimpleReader.java* from the Lab 10 section of the Labs folder on the course website. Also download the file *test.txt*, which is the file you will be reading. It is a text file, so you can look at its contents using Notepad.

b) To read from a file, you first need to create a `SimpleReader` object. The constructor for the `SimpleReader` class takes a parameter that is the name of the file to be read. In the Interactions pane, type the following statement:

```
SimpleReader reader = new SimpleReader("test.txt");
```

c) The `SimpleReader` class has two methods that we will use: `readFile()` and `getFileLength()`. The `readFile()` method is used to read all the lines in the file and store them in memory. It creates a String array, where each element of the array contains a line of text from the file. It **returns a reference to that String array.** The method `getFileLength()` returns the number of lines in the file. In the Interactions pane enter the statements below:

```
String []lineArray = reader.readFile();
```

The variable `lineArray` now references the String array that contains the lines of text read from the file.

```
System.out.println("The number of lines in the file is " +
reader.getFileLength());
System.out.println(lineArray[0]);
```

d) Type the Java statement that prints the third line of text.

e) Type the following Java statement. What happens?

```
reader = new SimpleReader("blahblah");
```

Exercise 4: A program to read text from a file

In this exercise, you will complete a Java program that uses the methods provided by the `SimpleReader` class to print all the lines of the file *test.txt*. Type this program into the Definitions pane, filling in the blank spaces, and then compile and run your program.

```
import java.io.*;

public class ReadandPrintFromFile
{
   public static void main(String[] args)
   {
     String fileName = "test.txt";

     SimpleReader reader = new _____________________________;
```

```
     String [] lineArray = reader. _____________________;

     for (int i = 0; i < reader._____________________; i++)

         System.out.println(lineArray[i] +  "\n");
   }
}
```

Exercise 5: How to parse a line of text

In this exercise, you will learn how to ***parse*** a line of text, i.e. to break it up into individual components (words). (You will find this useful for Assignment 4.)

Our file *test.txt* contains the following lines of text:

```
Homer Simpson hsimpson
Lisa Simpson  lsimpson
Bart Simpson  bsimpson
Marge Simpson msimpson
```

Using the class `SimpleReader` as in Exercises 3 and 4 results in the String array `lineArray` with these values:

```
lineArray[0] = "Homer Simpson hsimpson"
lineArray[1] = "Lisa Simpson  lsimpson"
lineArray[2] = "Bart Simpson  bsimpson"
lineArray[3] = "Marge Simpson msimpson"
```

Suppose we need to access the components in each line individually, i.e. first name, last name, username. Java provides a class called `StringTokenizer` that allows us to separate a string into ***tokens*** (i.e. individual components or words). For example, the string `"Homer Simpson hsimpson"` has three tokens: `"Homer", "Simpson"` and `"hsimpson".` The tokens are separated by one or more blank spaces in a line.

a) We will start by reading the lines from the file *test.txt* into lineArray, as in Exercise 3, and then creating a `StringTokenizer` object.The constructor for the `StringTokenizer` class takes a parameter that is the string to be tokenized. Reset the Interactions Pane and then type the following statements:

```
    import java.util.StringTokenizer;
    SimpleReader reader = new SimpleReader("test.txt");
    String []lineArray = reader.readFile();
    StringTokenizer tokenizer = new
StringTokenizer(lineArray[0]);
```

b) The StringTokenizer class has a method `nextToken()` that returns the next token in the line. Note that a token is a *substring* of the line. The first call to `nextToken()` will return the first token: the substring up to the first space in the

line. Type the following statements:

```
String firstName = tokenizer.nextToken();
System.out.println(firstName);
```

c) Type the following statements, filling in the blanks, to return and print the second token in the line (the person's last name):

```
String lastName = _____________________________;
System.out.println(lastName);
```

d) The next call to `nextToken()` will return the third token (the person's username). Type the statements to return and print the person's username.

e) Now type statements to print the person's name in the form **`Simpson, Homer`** followed by his username.

f) Type the statements needed to the print the information for Lisa Simpson as you did in part e) for Homer Simpson. What statements from parts a) to e) will you need to repeat?

Optional Exercise 6: A program to read a file and tokenize the lines

If you have time, write a complete Java program that reads the file *test.txt* and prints the information from each line of the file in the form you used in Exercise 5 part e).
Hint: use a for loop.