ITIS/ITCS 4180/5180 Mobile Application Development
Homework 1
Date Posted: 05/23/2012 at 01:45pm
Due Date: 05/26/2012 at 11:59pm

Basic Instructions:

1. In every file submitted you MUST place the following comments:
   - Assignment #
   - File Name.
   - Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. Please download the support files provided with this assignment and use them when implementing your project.
4. Submission details:
   a) When you submit the assignment, compress all `.java` and `.txt` files into a single zip file. The format of compressed file name is:
      i. HW#.zip
   b) You should submit the assignment through Moodle:
      i. Submit the zip file.
5. **Failure to do the above instruction will result in deduction points**

Assignment 1 (60 Points)

In this assignment you will get familiar with Java's `HashMap` and `LinkedList` collection classes. You will also practice some Object Oriented techniques by creating your own Java class to use within your code. This assignment consists of 2 parts:

Part 1 (20 Points):
You are given the file `countries.txt` which includes a list of country names. Each line of the text file contains a single country name. You are asked to perform the following tasks:

1- `AssignmentOnePartOne.java` should include the Part 1 of this assignment. PartOne.java should include the implementation for this part. In addition this class should implement the IPartOne.java Interface.
2- Read the file contents. You are given the code to read a file, see below.
3- Track the number of times each specific country name was listed in the file. That is, a country name could be repeated at multiple lines within the file, and you need to track how many times each country is repeated. Hint: you can use a `HashMap` to track the number of repetitions for each country.
4- Output the statistics your code has tracked, that is, each country name and the number of times it was detected in the file. Use a simple console output method for printing the collected statistics to include "country name" followed by the number of repetitions.

Part 2 (40 Points):
In this part, you are given the file `users.txt`, which includes information on a set of users. Each line in the file represents a single user record, where each record consists of a user's ID, NAME, AGE, ADDRESS, and EMAIL. The values are separated by a semicolon. Example record:

```
1 ; John Hodgeman ; 33 ; California, USA ; john@example.com
```

You are asked to perform the following tasks:

1- `AssignmentOnePartTwo.java` should include the implementation for this part. You are provided with the Person Class, which should be extended to implement the User Class.
2- Read the records in the users.txt file, Hint: extract each value from a user's record using Java's `String.split` method and set the delimiter to a semicolon, see provided code below.
3- Each user record needs to be assigned to a User object. To do so, you will create a User class, that extends from the person class, with instance variables: id, name, age, address, and email. These variables will hold the values of a user record read from the `users.txt` file. The User class should override the `toString()` method to print the user's `id`, `name`, `age`, `address` and `email`.
4- Keep track of all User objects by using a List.

5- Finally, you are asked to use the generated List to print out the information of all users who are younger than 20 years of age, sorted in ascending order by name. Hint: Use the `Collections.sort()` method.

Code Snippets

Read File:
For both parts, you can make use of the following code that reads in a file line by line. Use this code to help you read the files `countries.txt` and `users.txt`.

```
public void readFileAtPath(String filePath) {
	// Lets make sure the file path is not empty or null
	if (filePath == null || filePath.isEmpty()) {
		System.out.println("Invalid File Path");
		return;
	}

	BufferedReader inputStream = null;
	//We need a try catch block so we can handle any potential IO errors
	try {
		//Try block so we can use 'finally' and close BufferedReader
		try {
			inputStream = new BufferedReader(new FileReader(filePath));
			String lineContent = null;
			//Loop will iterate over each line within the file.
			//It will stop when no new lines are found.
			while ((lineContent = inputStream.readLine()) != null) {
				//Here we have the content of each line.
				//For now, I will print the content of the line.
				System.out.println("Found the line: " + lineContent);
			}
		}
		//Make sure we close the buffered reader.
		finally {
			if (inputStream != null)
				inputStream.close();
		}
	}
	catch (IOException e) {
		e.printStackTrace();
	}
}// end of method
```

String Tokenization:
For the purposes of Part 2 in this assignment, you would split the contents of a single line read from users.txt.

```
String[] resultingTokens = lineContent.split(";");
for (int i = 0; i < resultingTokens.length; i++)
	System.out.println(resultingTokens [i].trim());
```