ITIS/ITCS 4180/5180 Mobile Application Development
Homework 1
Date Posted: 09/04/2012 at 12:06am
Due Date: 09/10/2012 at 05:00pm

Basic Instructions:

1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. 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. Export your Android project as follows:
   a. From eclipse, choose *"Export…"* from the File menu.
   b. From the Export window, choose *General* then *File System*. Click *Next*.
   c. Make sure that your Android project for this assignment is selected. Make sure that all of its subfolders are also selected.
   d. Choose the location you want to save the exported project directory to. For example, your *Desktop* or *Documents* folder.
   e. When exporting make sure you select *Create directory structure for files*.
   f. Click Finish, and then go to the directory you exported the project to. Make sure the exported directory contains all necessary files, such as the .java and resource files.
5. Submission details:
   a. When you submit the assignment, compress your exported Android project into a single zip file. The format of compressed file name is HW#.zip
   b. You should submit the assignment through Moodle: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

Homework 1 (80 Points)

In this assignment you will get familiar with Java's Map, List and Set Interfaces. You will also practice some Object Oriented techniques by creating your own Java class to use within your code. This assignment consists of 3 parts:

Part 1 (25 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 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 for this purpose.
4. Output only the ***top 10*** repeated countries tracked by your code, that is, for each of the top 10 repeated countries print the 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 (25 Points):
In this part, you are given the file *userList1.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 first name, middle initial, last name, age, city, and state. The values are separated by a semicolon. Example record: Avis,E,Camacho,65,Millburn,NJ

You are asked to perform the following tasks:

1. *AssignmentOnePartTwo.java* should include the implementation for this part. You are provided with the User Class.
2. Read the records in the *userList1.txt* file. You should implement the parseUser() method in the User class. 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. The User class should override the *toString()* method to print the user’s first name, middle initial, last name, age, city and state.
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 24 years of age, sorted in ascending order by age. First print the total number of users who are younger than 24 years old, then print the list content in ascending order by age. Hint: Use the Collections.sort() method.

Part 3 (30 Points):
In this part, in addition to the file *userList1.txt* you are also given another file *userList2.txt*, which is formatted in a similar pattern. Each line in the file represents a single user record, where each record consists of a user's first name, middle initial, last name, age, city, and state. The values are separated by a semicolon.

You are asked to perform the following tasks:
1. *AssignmentOnePartThree.java* should include the implementation for this part.
2. Your goal is to find the list of users that exist in both files. Users are equal if they have exactly the same set of attributes. In other words, you should find the list of intersecting records between the two provided files provided and store the overlapping records in a list.
3. Finally, you are asked to use the generated List of overlapping users to print out their attributes, sorted in ascending order by age. First print the total number of users who are in the overlapping list, then print the list content in ascending order by age.

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());
```