ITIS/ITCS 4180/5180 Mobile Application Development
Homework 1
Date Posted: 01/15/2013 at 23:00
Due Date: 01/21/2013 at 23:55

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 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 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 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 (100 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. ***Your implementation should target the most efficient algorithms and data structures. You will be graded based on the efficiency of your implementation. You will not be awarded any points if you use simple nested loops to implement the below tasks.*** You should use the Map, List, and/or Set interfaces, and you are encouraged to review the lecture slides and the Java documentation. This assignment consists of 4 parts:

Part 1 (25 Points):
You are given the file "*words.txt"*, which includes a list of words. Each line of the text file contains a single word. You are asked to perform the following tasks:

1. PartOne.java should include the implementation for this part. Feel free to create any additional classes that are needed.
2. Track the number of times each word appears in the provided file. That is, a word could be repeated multiple times in the file, and you need to track how many times each word appears.
3. Output in descending order only the ***top 10*** repeated words, that is, for each of the top 10 repeated words print to the console the word and the number of times it was detected in the file. Hint: To sort use the Collections.sort() method.

Part 2 (25 Points):
You are given the file "*userList1.txt"*, which includes user records. Each line of the file represents a single user record. Each record consists of a user's first name, middle initial, last name, age, city, and state. The values are comma separated, for example, Avis,E,Camacho,65,Millburn,NJ.

You are asked to perform the following tasks:
1. PartTwo.java should include the implementation for this part. You are also provided with a User class. Feel free to create any additional classes that are needed.
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 comma, see provided code below. Each user record should to be assigned to a User object.
3. Your goal is to locate the set of repeated user records in the provided file. Repeated user records are records that appear more than once in the provided file. You should use HashSets of User objects to perform this task.
4. Print the set of repeated user records in ascending order by last name.

Part 3 (25 Points):
In this part, in addition to the file "*userList1.txt*" you are also given another file "*userList2.txt*". Both files are formatted in a similar pattern.

You are asked to perform the following tasks:
1. PartThree.java should include the implementation for this part. You are also provided with a User class. Feel free to create any additional classes that are needed.
2. Your goal is to find the set of users that exist in both files. Users are equal if they have equal attributes. In other words, you should find the set of intersecting records between the two provided files provided.
3. Print the set of overlapping users, sorted in ascending order by age. First print the total number of users who are in the overlapping set, then print the set content in ascending order by age.

Part 4 (25 Points):
You are given the file "*countries-info.txt*" which includes a list of country information. Each line of the text file contains a country record, which includes continent, country name, ISO3, country number, country full name. You are asked to perform the following tasks:
1. PartFour.java should include the implementation for this part. Feel free to create any additional classes that are needed.
2. Read the file contents. Track for each continent the countries contained in it. Hint: Use both HashMap and ArrayList.
3. For each continent output the continent name and a comma separated list of the country names in that continent sorted in ascending order. The printed continent summaries should be formatted as follows:

```
North America:
      Anguilla,Antigua and Barbuda,Aruba,Bahamas,.....
South America:
      Argentina,Bolivia,Brazil,Chile,Colombia,.....
Antarctica:
      Antarctica,Bouvet Island (Bouvetoya),French Southern Territories,.....
```

Code Snippets

Read File:
The following code that reads in a file line by line. It is assumed the file is included in root folder of the Eclipse project. Use this code to help you read the provided files.

```
public void readFileAtPath(String filename) {
	// Lets make sure the file path is not empty or null
	if (filename == null || filename.isEmpty()) {
		System.out.println("Invalid File Path");
		return;
	}
	String filePath = System.getProperty("user.dir") + "/" + filename;
	BufferedReader inputStream = null;
	// We need a try catch block so we can handle any potential IO errors
	try {
		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) {
				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:
To split the contents of a single line read a file.

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