ITIS/ITCS 4180/5180 Mobile Application Development
In Class Assignment 1

Basic Instructions:

---

1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of the student.
2. This is an individual assignment. Each student is required to submit the assignment on Canvas.
3. Please download the support files provided with this assignment and use them when implementing your project.
4. **Submit Codes:**
   a. This is a basic Java assignment. You are free to use any development environment.
   b. Use any IDE or Text editor to write your code. Zip all the source files together and submit.
5. Submission details:
   a. The file name is very important and should follow the following format: **Student800#_InClass01.zip**
   b. You should submit the assignment through Canvas: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

In Class Assignment 1 (100 Points)

In this assignment you will practice using Data Structures and Object Oriented concepts in Java. ***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 one or more of the below data structures:

- ArrayList :
  - JavaDoc: http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html
  - Tutorial: http://docs.oracle.com/javase/tutorial/collections/interfaces/list.html
- HashSet :
  - JavaDoc: http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html
  - Tutorial: http://docs.oracle.com/javase/tutorial/collections/interfaces/set.html
- HashMap :
  - JavaDoc: http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html
  - Tutorial: http://docs.oracle.com/javase/tutorial/collections/interfaces/map.html

Question (100 Points)

You are given a file "userList1.txt" which contains a list of users. Each line of the file represents a single user. Each record consists of a user first name, middle name, last name, and street address. The values are comma separated, for example:

John,M,Snow,10,Edinburg,VA

You are asked to perform the following tasks:

1. InClassOne.java should include the implementation for this question
2. Write "User" class that represents the details provided in the file
3. 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.
4. 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. **Hint**: you need to use one of the data structures listed above.
5. Print the set of repeated user records in ascending order by last name. **Hint**: To sort use the Collections.sort(). http://docs.oracle.com/javase/6/docs/api/java/util/Collections.html

Some Sample Code

The following code reads in a file line by line. It is assumed that 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());
}
```