Lab 2

Important Guidelines

All labs will be submitted on Blackboard, unless otherwise noted in the lab description. Export the project and upload the zip file. **In Eclipse, click on the project, select Export, and choose Archive File.**

You need to submit a zipped file that contains the whole project. You need to submit a zipped version of your whole project. If you submit only the class files or src files, that is not acceptable.

We will then import your project and run from our end. If you don't submit the whole project, we won't be able to run your project.

The name of each project must be **abc123-labX**, where abc123 is your UTSA ID and X is the number of the lab. All letters are lowercase.
The name of your submission must be **abc123-labX.zip**. All letters are lowercase.
Any submission not following these requirements may not receive credit.

Reminder: laboratories must be completed individually. Collaboration with anyone violates the policies for academic integrity.
You may discuss algorithms and approaches with others. You may not code together or share a computer, under any circumstance.
If you have questions or concerns about this policy, contact your instructor immediately.

In the rare event that a change must be made to an individual lab description, the change will be announced in lecture, and a note will appear on the lab submission on Blackboard.

Objectives:

- Javadoc
- ArrayList
- File I/O
- UML diagrams

Task: Make your Lab1 Better!

The school principal is very pleased with your work. Now, she wants you to develop the version 2.0.

In version 2.0 of their software, they have provided some sample data files for your program to pull from. With this update, zones and students will automatically be loaded into their system from these files. Additionally, school teachers will be able to monitor more information. Also, based on the zone safety factor, the allocation of resources and teachers will be determined to take care of the students.

Getting Started

Begin by creating a new Java project in Eclipse, named according to the lab guidelines. For this lab, you may reuse your code from Lab 1, but you should correct any mistakes. If you copy the files over, ensure that you choose "copy" if prompted, rather than "link", as the latter will not move the file into this project directory.

Your project should now contain **School.java**, **Zone.java**, and **Student.java**. All classes in this lab will be in the default package of your project.

Your application will read in data from text files placed in a **data** directory. Sample files can be downloaded as a zip file (provided in lab2 folder). Please unzip it and place the folder with files at the top of your project in Eclipse. *Note: the top of your project is within the folder for the project, not within the src folder.*

To get you started, we've provided a test class, Lab2.java. Your final submission must include this class exactly as it appears here, and the data files given. The only permitted modification to this code is to add Javadoc comments as needed. Once your application is completed, running Lab2.java with the given data files will result in the exact output shown below.

Lab2.java

Output

```
Welcome to Monroe_May_Elemn_School!
-------------------------
```

B: Classroom (low risk)
*Lawrence -1st- (veg)
*Arman – 1st-(veg)
|
|
|
|

D:Hallway (medium risk)
*Dude – K – (non-veg)
*Dave – K – (non-veg)
|
|
|
|
|

****after relocation****

```
Welcome to Monroe_May_Elemn_School!
-------------------------
```

B: Classroom (low risk)
*Arman – 1st-(veg)
|
|
|
|

D:Hallway (medium risk)
*Dude – K – (non-veg)
*Dave – K – (non-veg)
|
|
|
|
*Lawrence -1st- (veg)

Note that this format is different from Lab 1 - this should affect your toString() methods!

Student.java

Students objects will be as previously defined, however they will additionally have:

- An assigned zone code (i.e. TY for the Field-Trip zone)

All class variables must have getters and setters.

Zone.java

Zone objects will be as previously defined, however they will additionally have:

- An **ArrayList** of Student objects (rather than the previous array)
- A zone code (i.e. TY for the Field-Trip zone) in addition to the existing zone name.
- A critical safety rating (i.e. low, medium, high)

This class should have a method **addStudent** which takes in a single Student and adds them to that Zone.
It should also have a method **removeStudent** which takes in a single Student and removes them from that Zone.

All class variables must have getters and setters.

School.java

The School class will be as previously defined, however it will contain an **ArrayList** of Zone objects, rather than an array.

In order to work with the code provided in Lab2.java, this class will need two new methods: **loadZones(..)** and **loadStudents(..)**. Both methods will take in a file name and throw an IOException. These object methods will load the data in the given file into the appropriate objects (zones are added to the school, and students are added to zones).

Create an object method **relocate(..)** which takes as parameters the name of a student and the zone code to which they should be relocated. This method should remove the student from their currently assigned zone, and add them to the zone given by the zone code parameter.

Create an object method **save()** which takes no parameters and saves all current school information back into the data files provided. *This way, if any students have been relocated, their new zone information will appear with their name in the school files.* Be sure to *overwrite* the data in the file, rather than append to it. The data must be saved in the same format as provided.

All class variables must have getters and setters.

Creating Javadoc

1) First consult this oracle webpage to learn about how you can comment your code. Also, go over the slides. You need to include @author, @param, @return etc. Please follow the standard guidelines. After you are done commenting your code, then you need to create the Javadoc.
   https://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

2) In Eclipse, select Project > Generate Javadoc
3) Destination: workspace/(your_project)/doc
4) Next, Select all "referenced archives & projects"
5) Select, Finish

Creating UML Diagram

1) Please follow the instructions to install the violet UML editor

http://www.cs.utsa.edu/~cs3443/uml/violetUML.html

2) After that create a class diagram for your project using violet UML.
3) Export your diagram as an img file and place that img file in your project folder.
4) It's OK if you want to use other software to generate your UML diagram.

More info on UML:

1) Employee example code: http://cs.utsa.edu/~cs3443/notes/chapter08/ch08.html
2) UML relationships & associations: http://cs.utsa.edu/~cs3443/uml/uml.html
3) Advanced UML software & concepts: http://www.uml.org
4) In-depth overview of syntax: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

Rubric:

- (30pts) School.java
- (30pts) Student.java and Zone.java
- (20pts) Correctness - Your program will be tested using different data files, in the same format as given.
- (10pts) Javadoc comments on all classes, including getters and setters. Each class must have your name and abc123 at the top, in addition to the description of the class.
- (10pts) UML diagram

Submissions which do not compile will receive a maximum of 10 points in total.

There are two versions of Lab2 class is provided below. Use any one of them. Lab2 class is your Test class to Test your other classes.

```java
import java.io.IOException;
/**
 * Lab2 is a Java class containing a main method to run your program when completed.
 * This class will not compile until you have completed the requirements outlined in
 * the lab description.
 *
 * @author
 * UTSA CS 3443 - Lab 2
 * Fall 2020
 */

//version 1 of Lab2
public class Lab2 {

     public static void main( String[] args ) {
          School m = new School( "Monroe_May_Elemn_School" );

          try {
               // Load data for the school, and print its status
               m.loadZones( "data/zones.csv" );
               m.loadStudents( "data/students.csv" );

               System.out.println( m );

               // Relocate Blue to the Field-Trip zone, and print school status
               m.relocate( "Blue", "TY" );
               m.save();

               System.out.println( m );

          }catch( IOException e ) {
               System.out.println( "Error loading the file - please check its location." );
```

```
                e.printStackTrace();
            }
        }
}
```

```
import java.io.IOException;


/**
 * Lab2 is a Java class containing a main method to run your
program when completed.
 * This class will not compile until you have completed the
requirements outlined in
 * the lab description.
 *
 * @author
 * UTSA CS 3443 - Lab 2
 * Fall 2020
 */
//version 2 of Lab2
public class Lab2 {

    public static void main( String[] args ) throws
IOException {
        School m = new School( "Monroe_May_Elemn_School" );

        // Load data for the school, and print its status
        m.loadZones( "data/zones.csv" );
        m.loadStudents( "data/students.csv" );

        System.out.println( m );

        // Relocate Blue to the "TY" zone, and print school
status
        //m.relocate( "Blue", "TY" );
```

```
            //m.save();

            System.out.println( m );
        }
}
```

-