CIS 163 Project 3 – A Camping Reservation System
THIS IS A GROUP (2 members) PROJECT.

Due Date

- See schedule at the end of the syllabus

Why are we doing this project.
The instructor will fill in the details; here are some bullet points

- Whether campSites are being rented is irrelevant, most companies have an inventory of items to track.
- In most projects, partial source code will have been developed, so working with old code is important.
- In most projects today that are small (mid) size, client involvement is a must. In some projects, involvement could be weekly, while other projects could be less frequent. For us in a classroom setting, I expect questions from groups in office hours and on Piazza (or in class when appropriate), so I (as the client) know work is being done from the start of the project. (Being completely open here, if questions are not being asked, I will assume work is not getting done, and I would be very concerned if I was paying for this project)
- There are 4 major requirements that are difficult (save/load text, using streams, anonymous classes, and figuring out how this current program functions), I am happy to help any group that comes to the office or ask questions in class.
- **This project is challenging**, and is typical for what you will be doing in a company; not an inventory system as in our case, but something that starts with existing code, where your boss (or an actual client) has a set of goals for you to achieve.

Before Starting the Project

- Review Chapters 8 - 10 and Chapters 12, 13, 15, 18 of the CIS163 book
- Read this entire project description before starting, if you have any question please ask the instructor

Learning Objectives

After completing this project you should be able to:

- Use inheritance and polymorphism
- Use advanced Swing components like `JTable` and `AbstractTableModel`
- Save and restore objects using the **Serialization/text** files
- Use Collections.sort
- Using simple Date and GregorianCalendar classes
- Using streams and Lambda functions
- Using anonymous classes

Program description: Your assignment is to create a program that helps camping owners manage their inventory. You should be able to rent tentOnly and RV sites from your program. A full description follows.

A completed program must have the following functionality:

- Save, load the rental database with serialized files using JFileChooser
- Save, load the rental database with text files using JFileChooser
- Reserve a TentOnly or RV site with a checkin date, checkout date, name of camper
- Complete error checking

- **And much more! DETAILS BELOW**

Step 0: I have provided a functioning program that will get you started on this project. The intent of the code is to help you understand the different techniques that will be used in your final project. This sample program has many issues, such as, no error checking, incomplete results and most importantly does not implement most of the functionality that is required in your project. **In other words, there is much that must be changed on the sample code; it should get your team thinking about how best to proceed.**

Step 1: Create an Eclipse/Intellij project named "RentalPrj"

- Create a package named: project3 and install ALL of the provided code.

Step 2: Implement the CampSite (base class) and using the following:

(This has been done for you, i.e., in the staring code)

```
public class CampSite implements Serializable {
      // What is the purpose of this variable (search google)
      private static final long serialVersionUID = 1L;

      /** The date the CampSite was checkIn on */
      protected GregorianCalendar checkIn;

      /** The Name of person that is reserving the CampSite */
      protected String guestName;

      /** and so on */
```

Step 3a: TentOnly is a derived class by extending CampSite and using the following:

(This has been done for you, i.e., in the staring code)

```
public class TentOnly extends CampSite {

    private int numberOfTenters;
```

Step 3b: RV is a derived class by extending CampSite and using the following:

(This has been done for you, i.e., in the staring code)

```
public class RV extends CampSite {
       private int power;
```

Step 4: Implement the class GUICampReservationSystem using the following:

(This has been done for you, i.e., in the staring code)

```
public class GUICampReservationSystem extends JFrame implements ActionListener{
         // declare GUI components (menu items, buttons, etc.) needed
         // constructor method that prepares the GUI
         // event handlers and other methods needed to build the GUI
```

- The GUICampReservationSystem class is the class that displays the GUI to the user and allows the user to reserve an RV or tentOnly site, and allows the user to check out too. In addition, the GUI allows the users to save and load the database using serialized **and text files**. The

GUITentOnlyDealer must handle the following operations shown below. The first screen shot shows the main GUI screen:

Step 5: Implement the class ListModel using the following (THIS IS THE MOST IMPORTANT CLASS TO UNDERSTAND):

This class is used for managing the campSite units (TentOnly and RV) into an ArrayList<CampSite>. (Note: CampSite is the base class and review chapter 9 of your book). The functionality of this class is similar in concept to code presented in chapter 9, specifically, the staffList array. The main difference is that this class must handle all the operations from the GUI class. That is, reserving a tentOnly or RV sites, checking out of tentOnly or RV sites, save and load, etc.

Note: The following code is just a start; to fully understand how to create the ListModel see the class notes. Examples of a ListModel class will be presented in class.

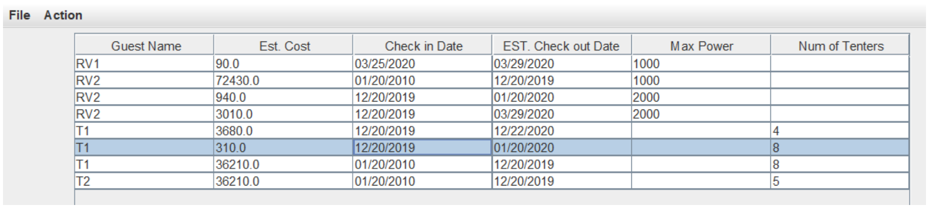
```
public class ListModel extends AbstractTableModel {

       private ArrayList<CampSite> listCampSites;

       private String[] columnNamesCurrentPark = {"Guest Name", "Est. Cost",
        "Check in Date", "EST. Check out Date ", "Max Power", "Num of Tenters"};

       @Override
       public int getColumnCount() {

       @Override
       public int getRowCount() {

       @Override
       public Object getValueAt(int row, int col) {

       @Override
       public String getColumnName(int col) {

              // add methods to add, delete, and update.
              // add methods to load/save accounts from/to a binary/text file
              // add other methods as needed
```

The above description is just a starting point, please run the code provided and fully understand how it functions.

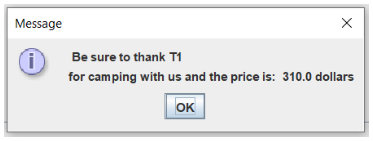
This section starts the new functionality your group must implement.

The real step 1: Change the cost function.

Tents are 10 dollars per day, with an extra 10 dollars a day for more than 10 tenters. For example:
RVs start at 10 dollars, with 20 days per day, with an extra 10 dollars per day for power greater than 1000. To see all cost functions, see Est Cost be column below.

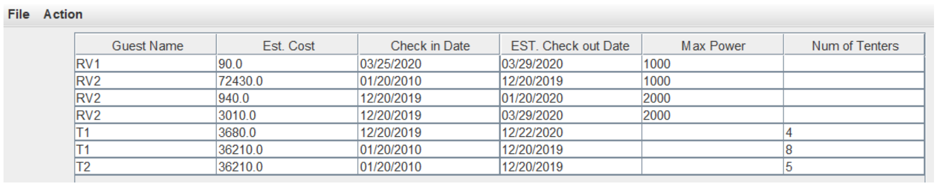


Guest Name	Est. Cost	Check in Date	EST. Check out Date	Max Power	Num of Tenters
RV1	90.0	03/25/2020	03/29/2020	1000	
RV2	72430.0	01/20/2010	12/20/2019	1000	
RV2	940.0	12/20/2019	01/20/2020	2000	
RV2	3010.0	12/20/2019	03/29/2020	2000	
T1	3680.0	12/20/2019	12/22/2020		4
T1	310.0	12/20/2019	01/20/2020		8
T1	36210.0	01/20/2010	12/20/2019		8
T2	36210.0	01/20/2010	12/20/2019		5



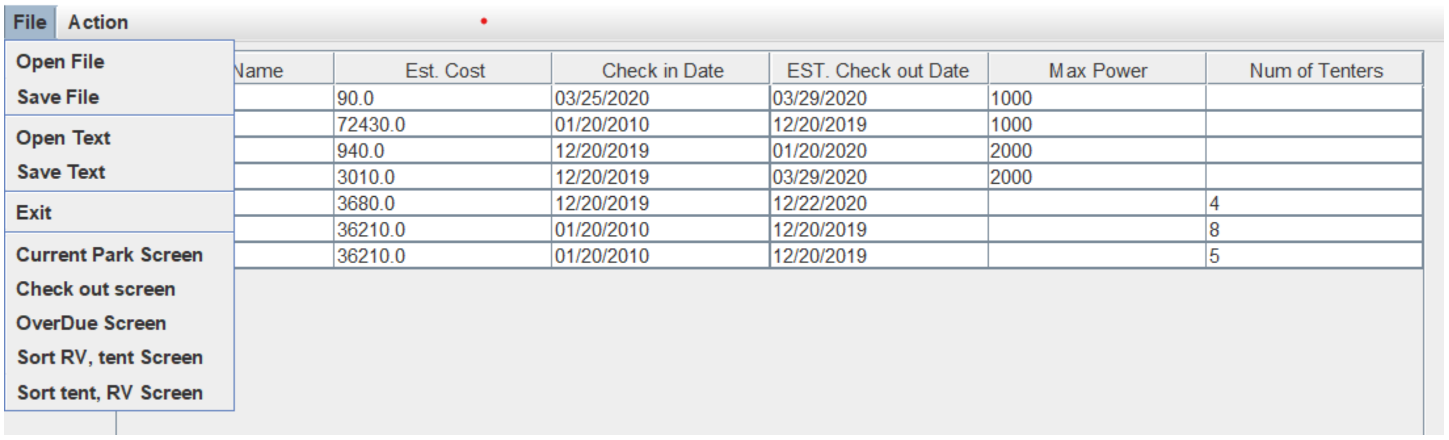
Be sure to thank T1
for camping with us and the price is: 310.0 dollars

(Note: that campsite has been removed)



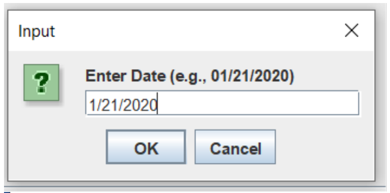
Guest Name	Est. Cost	Check in Date	EST. Check out Date	Max Power	Num of Tenters
RV1	90.0	03/25/2020	03/29/2020	1000	
RV2	72430.0	01/20/2010	12/20/2019	1000	
RV2	940.0	12/20/2019	01/20/2020	2000	
RV2	3010.0	12/20/2019	03/29/2020	2000	
T1	3680.0	12/20/2019	12/22/2020		4
T1	36210.0	01/20/2010	12/20/2019		8
T2	36210.0	01/20/2010	12/20/2019		5

Step 2: Add on the saving/loading the database using a text file.

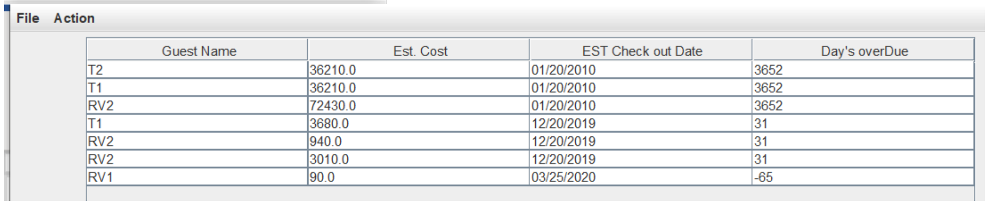


Name	Est. Cost	Check in Date	EST. Check out Date	Max Power	Num of Tenters
	90.0	03/25/2020	03/29/2020	1000	
	72430.0	01/20/2010	12/20/2019	1000	
	940.0	12/20/2019	01/20/2020	2000	
	3010.0	12/20/2019	03/29/2020	2000	
	3680.0	12/20/2019	12/22/2020		4
	36210.0	01/20/2010	12/20/2019		8
	36210.0	01/20/2010	12/20/2019		5

Step 2: Add on the OverDue screen that sorts all the campsites based upon number of days overdue. First enter a date to be used as the reference point.



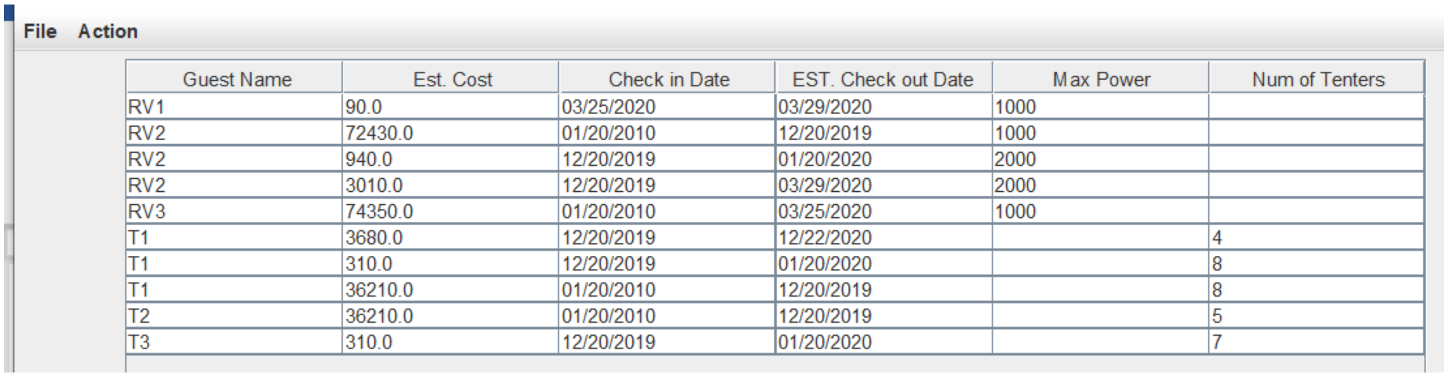
Enter Date (e.g., 01/21/2020)
1/21/2020



Guest Name	Est. Cost	EST Check out Date	Day's overDue
T2	36210.0	01/20/2010	3652
T1	36210.0	01/20/2010	3652
RV2	72430.0	01/20/2010	3652
T1	3680.0	12/20/2019	31
RV2	940.0	12/20/2019	31
RV2	3010.0	12/20/2019	31
RV1	90.0	03/25/2020	-65

Step 3: Add on Sort RV tent screen. This will sort RVs first by guest name, then Tents second by name. Must use a Lamda function and put a comment directly above it in your code

```
// LAMBDA FUNCTION USED HERE
```



Guest Name	Est. Cost	Check in Date	EST. Check out Date	Max Power	Num of Tenters
RV1	90.0	03/25/2020	03/29/2020	1000	
RV2	72430.0	01/20/2010	12/20/2019	1000	
RV2	940.0	12/20/2019	01/20/2020	2000	
RV2	3010.0	12/20/2019	03/29/2020	2000	
RV3	74350.0	01/20/2010	03/25/2020	1000	
T1	3680.0	12/20/2019	12/22/2020		4
T1	310.0	12/20/2019	01/20/2020		8
T1	36210.0	01/20/2010	12/20/2019		8
T2	36210.0	01/20/2010	12/20/2019		5
T3	310.0	12/20/2019	01/20/2020		7

Step 3: Add on Sort tent RV screen. This will sort tents first by guest name, then RV second by guest name. If the names are the same, sort by estCheckOut Date. Must use a anonymous class and put a comment directly above it in your code

```
// ANONYMOUS CLASS USED HERE
```

Step 4: TOTALLY error checked your program (the whole program). For example: CheckOut Date was before checkIn date; improper date such as: "abc/abc/abc"; etc.

Step 5: Use JUnits to test fully your ListModel class. Minimal 100% coverage.

Step 6: Comment all of your code, but, use the Java Style Guide on the ListModel. That class will be graded.

Important, I will have test data to be used for this project during demonstrations using as a starting point the data found at the bottom of the list engine.

--------------------------- YOU'RE DONE ☺ -------------------------------

CIS 163 – Computer Science II
Project 3: “Camping reservation” Program

Student Name	
Date Submitted, Days Late, Late Penalty	

Graded Item	Pts	Points Secured / Comments
Javadoc Comments and Coding Style/Technique (http://www.cis.gvsu.edu/studentsupport/javaguide) • Code Indentation (campSite format source code in IDE) • Naming Conventions (see Java style guide) • Proper access modifiers for fields and methods • Use of helper (private) methods • Using good variable names • Header/class comments • Every method uses @param and @return (1 sentence after) • Every method uses a /**************** separator • Overall layout, readability, **No text wrap** • Using /** … / for each Instance variable • Has many inner “inner” comments	10	
• Saving and loading text files • Cost function • Error on input checking	20 13 10	
• Sort by RV Tent • Sort by Tent RV • JUnits	10 15 12	
Cleaning up the existing code that you used in your project. In other words, have a good design, no wasted lines of code, No extra code. etc. **MISC stuff**	**10**	
Total	**100**	

Note: Your score may be lowered from the rubric above based on effective group work, your individual contributions to the project, and knowledge of all parts of the project. It is fine if you and your partner split up the work, but you must work effectively as a group, the work distribution must be close to even, and you must understand how all parts of the project work.

Comments: (extra credit)