**CIS 163 Project 3**

**A “Reserve Camping” program**

**Due Date**

* At the beginning of the lab; see the schedule, last page of the syllabus

**Before Starting the Project**

* Review Chapters 8 - 9 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 Abstract Models
* Save and restore objects using the Serialization API
* Save and restore objects to/from a text file
* Using simple Date and GregorianCalendar classes

**Program description:** Your assignment is to create a camp reservation program. Your program will be able to occupy (this is referred to as checking in) a site for tents and RVs. That is, you can Check-in (occupy) a Tent or a RV site using your program for a specific time period (check-in date, check-out date). For grading purposes, the campground has only 5 sites that can be occupied (it is a very small camping area! Very exclusive! ☺). Your program can also allow a user to checkout (no longer occupy the site, that is, leave a site) from the campground. At checkout, your program will calculate cost for occupying that site based upon the number of days the site was occupied.

**A completed program must have the following functionality:**

* Save and load the reservation database with serialized files using JFileChooser
* Save and load the reservation database with Text files using JFileChooser
* Check-in a RV with a date, site #, estimated days staying, power need and a Occupier’s name
* Check-in a Tent with a date, site #, estimated days staying, Occupier’s name and number of tenters
* Checkout (camper is leaving) a RV and generate a cost (see below for details)
* Checkout (camper is leaving) a Tent and generate a cost (see below for details)
* **This program will be demonstrated in class to show the full functionality of the program, so attending class is very important**.

*Before you turn in your work: use the* [*Java Style Guide*](http://www.cis.gvsu.edu/studentsupport/javaguide) *to document your project. (10 pts)*

*Steps 0 – 7 should be completed first (the ordering is a suggestion). Step 8 – 9 should be completed second (the ordering is a suggestion.) Finally, Steps 10, and 11 (the ordering is a suggestion)*

**Step 0: Create an initial UML model of your proposal solution of your project. Use a tool like dia from the EOS lab to create your class diagram. Due on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. A final UML model that represents your final solution of this project is due upon presentation.**

**Step 1: Create an Eclipse project named “CampingPrj”**

* Create a package.
* Create a class named: Site implements Serializable
* Create a class named: Tent that extends Site
* Create a class named: RV that extends Site
* Create a class named: GUICampingReg extends JFrame implements ActionListener
* Create a classes named: DialogCheckInRv extends JDialog
* Create a classes named DialogCheckInTent extends JDialog
* Create a class named: SiteModel extends AbstractTableModel

Note: The Site class is the base class and the Tent, RV classes extend the Site class.

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

*public class Site implements Serializable {*

*private static final long serialVersionUID = 1L;*

*/\*\* The name of the person who is occupying the Site \*/*

*protected String nameReserving;*

*/\*\* The date the Site was checked-in (occupied) \*/*

*protected GregorianCalendar checkIn;*

*/\*\* The estimated number of days the person is reserving \*/*

*/\*\* This is just an estimate when the camper is \*/*

*/\*\* is checking in \*/*

*protected int daysStaying;*

*/\*\* The date the Site was checked out \*/*

*/\*\* This is the exact day they checked-out \*/*

*protected GregorianCalendar checkOutOn;*

*/\*\* The Site number \*/*

*protected int siteNumber;*

// add constructors

// add getter, setter methods

**Step 3: Tent is a derived class by extending Site and using the following:**

public class Tent extends Site {

/\*\* Represents the number of tenters on this site \*/

private int numOfTenters;

// add constructor

// add getter, setter methods

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

*public class Tent extends Site {*

*/\*\* Represents the power supplied to the site \*/*

*private int power; // 30, 40, 50 amps of service.*

// add constructor

// add getter, setter methods

**Step 4: Implement the class GUICampingReg using the following:**

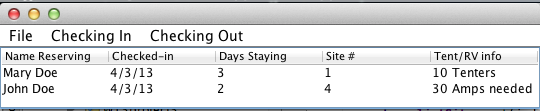
*public class GUICampingReg extends JFrame implements ActionListener{*

// declare GUI components (menu items, buttons, etc.) needed

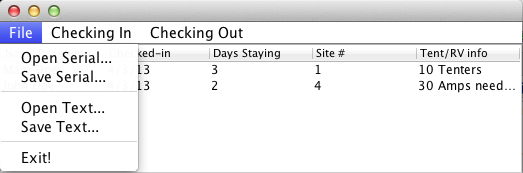
// constructor method that prepares the GUI

// event listeners and other methods needed to build the GUI

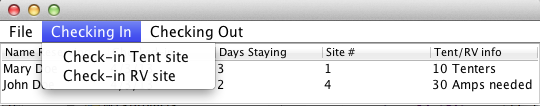
* The GUICampingReg class is the class that displays the GUI to the user and allows the user to occupy a Site (RV, or Tent). In addition, the GUI allows users to save and load the reservation database (sites that are occupied) using serialized files. The GUICampingReg must handle the following operations shown below (and underlined). The following are suggestions on where to place these operations on a GUI JMenu. The first screen shot shows the main “suggested” GUI screen:



* The (File Menu) has read and write operations to save or load the database to Serializable files and use a JFileChooser to select the file (see class notes and google). For example:

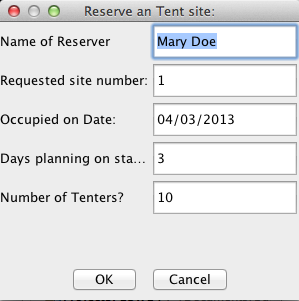


* The (Action Menu) has RV and Tent check-in operations available under this menu. For example:



**Step 5: Implement the class JDialog classes (DialogCheckInRV, DialogCheckInTent):**

These dialog boxes are invoked with the user selects Check-in. The only difference between the two dialog boxes is: for Reserving a Tent site: ask the number of tenters; for RVs, as the power needed. Here is a sample screen shot:



To create a dialog box the following code is suggested. Note: The following code is just a start; to fully understand how to create dialog boxes, more details are needed. Examples of other dialog boxes will be presented in class.

*public class DialogCheckInRv extends JDialog implements ActionListener{*

*private JTextField nameTxt;*

*private JTextField OccupyedOnTxt;*

*private JTextField stayingTxt;*

*private JTextField siteNumberTxt;*

*private JTextField powerTxt;*

*private JButton okButton;*

*private JButton cancelButton;*

*private boolean closeStatus;*

*private Site unit;*

*public DialogCheckInRv(JFrame paOccupy, Site d) {*

*unit = d;*

*...*

*}*

*public void actionPerformed(ActionEvent e) {*

*...*

*}*

Finally, to invoke this dialog box from the GUICampingReg class, the following code may help.

*if (resvRVItem == comp) {*

*RV t = new RV();*

*DialogCheckInRv x = new DialogCheckInRv(this, t);*

**Step 6: Implement the class SiteModel using the following:**

This class is used for maintaining the database of sites (RV and Tent) and there status into an ArrayList<Site>. (Note: Site is the base class and it is suggested to review the pages on inheritance and polymorphism in your book. The functionality of this class is similar in concept of polymorphism presented in your book, specifically, the staffList array.) The main difference is that this class must handle all the operations from the GUI class. That is, check-in a RV, check-in a Tent, store the db, load the db, etc. Note: The following code is just a start; to fully understand how to create the SiteModel, more details are needed. Examples of SiteModel classes will be presented in class.

*public class SiteModel extends AbstractTableModel {*

*private ArrayList<Site> listSites;*

*private String[] columnNames = { "Name Reserving", "Checked in", "Days Staying", "Site #", "Tent/RV info"};*

// constructor method that initializes the arraylist

// override these two methods from AbstractTableModel class

*public Object getValuesAt(int row, int col) {*

*...*

*}*

*public String getColumnName(int col) {*

*return columnNames[col];*

*}*

*public int getSize() {*

*...*

*}*

*public int getColumnCount() {*

*return columnNames.length;*

*}*

*public int getRowCount() {*

*return listSites.size();*

*}*

// add methods to add, delete, and update.

// add methods to load/save accounts from/to a binary file

// add other methods as needed

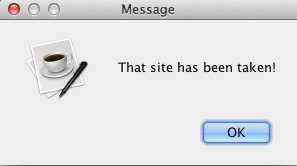
Notes regarding the SiteModel class:

1. To make updates to the Sites in the model immediately visible in the JTable on your GUI, it is important that the methods in the SiteModel class that modify (add, delete, and update) the Sites notify the JTable immediately after any changes. These notifications can be sent from SiteModel class using one of these methods: fireTableRowsDeleted(), fireTableRowsInserted(), and fireTableRowsUpdated(). The SiteModel class inherits these methods from the AbstractTableModel class.

2. To save and load from a serialized file is not shown in this document, however, a demonstration of this ability will be done in class.

**Step 7: Implementing the Check-In function:**

At check-in (RV or Tent) a calculation of the estimated cost is displayed using the following (this cost is based on the estimated number of days staying). For RV sites: the cost is 30 dollars per day. For Tents, 3 dollars per day times the number of Tenters. Use a JOptionPane.showMessageDialog to output the cost. Also, do not allow the same site to have two campers checked in (i.e., one Occupier per site). If a camper attends to check-in on an already checked-in site (i.e. same site), block the action and show warning dialog box. Here are some examples:

 **

*----------- Do not start step 8 until the above is completed ----------*

**Step 8: Add on the following functionality to the GUI and SiteModel class.**

* Create a new JMenuItem to allow the user to save and load the database using a text File (use the JFileChooser to select the filename). In project 1, you saved information using a text file and the technique used there is similar to what is used here. Following code snippet will help you with writing to a file.
* Implement Good error checking in your program. For example, entering in a check-out date in the JDialog box that is before the check-in date. Be sure to have good error checking on user input.

**Here is a code snippet that may help you when writing to a text file:**

public void sampleWriteData(){

try {

PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(filename)));

out.println(listSites.size());

for (int i = 0; i < listSites.size(); i++) {

// listSites is an ArrayList<Site>

// Site SiteUnit = listSites.get(i);

// Output the class name.

out.println(SiteUnit.getClass().getName());

// Output the OccupyOn date to a file in a readable format.

out.println(DateFormat.getDateInstance(DateFormat.SHORT).

format(SiteUnit.getOccupyedOn().getTime()));

// Output the Title of the Site

out.println(SiteUnit.getTitle());

// See if the curOccupy SiteUnit is a Tent, if so output the player.

if (SiteUnit instanceof Tent)

out.println(((Tent)SiteUnit).getPlayer());

}

out.close();

} catch (IOException ex) {

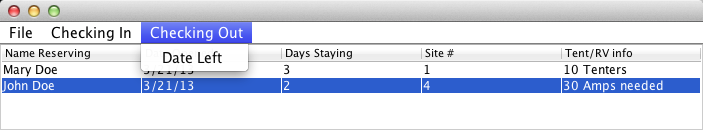
System.out.println ("IO Error!");

}

}

**Step 9: Add on the following functionality to the GUI, SiteModel and Dialog classes.**

* Add on a new JMenu item that allows the camper to checkout. Have a dialog box appear asking for the checkout date. For example:



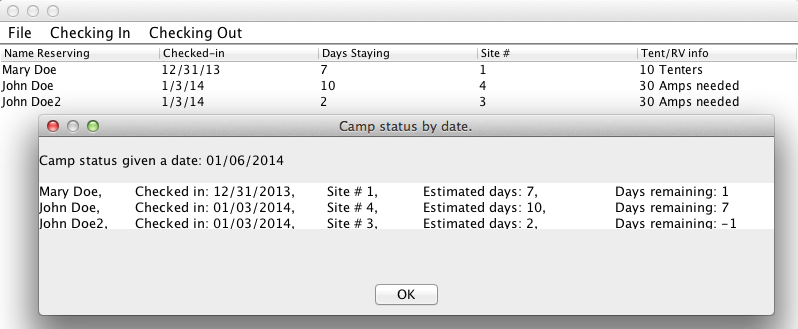
* When checking out of a site, use the following to calculate actual cost. Remember, the previous cost (step 7) was just an estimate. For RV sites: the cost is 30 dollars per actual days stayed. For Tents, 3 dollars times number of Tenters times number of actual days stayed. Again, use a JOptionPane.showMessageDialog to output the cost.
* TOTAL error checking of user input. Show Warning Dialog boxes showing the error and block the input from your program. You program should handle ALL situations!

**Step 10: Add on the following functionality to the GUI, SiteModel and Dialog classes.**

Create a warning dialog box that indicates all sites have been occupied.

**Step 11: Add on the following functionality to the GUI, SiteModel and Dialog classes.**

Create a new JMenuItem (named: Status) that when selected opens a JDialog (named: CampFullStatus) that accepts a date and returns all sites with the number of days left on the expected occupancy. For example, if a camper checked in (occupied) on 10/10/2013 with the estimated number of days to stay was 5, and the input date was 10/12/2013, then the site number, name, check-in date would be displayed in the list with a 3 (days left on site) next to that site. A negative number would indicate the camper has overstayed their estimated number of days.



**Step 12: Add on something you think has value to a camping system. Make the functionality complex enough to be worth a step 12. You must get instructor approval for your idea for completing step 12.**

--------------------------- YOU’RE DONE ☺ -------------------------------

**Some additional grading criteria**

There is a 70% penalty on programming projects if your solution does not compile.

* Stapled cover page with your name and signed pledge. (-5 pts if missing)

**Late Policy**

Projects are due at the START of the class period and the first 24 hours of being late is -15 pts.

* Each subsequent weekday is an additional -10 pts

**Turn In**

A professional document is stapled with an attractive cover page.

* Cover page - Your project must have a cover page that includes your name, a title, an interesting graphic or photograph related to the project topic and the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor). " You are responsible for understanding and adhering to the [School of CIS Guidelines for Academic Honesty](http://www.cis.gvsu.edu/Academics/Honesty/).

**Before you turn in your work: use the** [**Java Style Guide**](http://www.cis.gvsu.edu/studentsupport/javaguide) **to document your project. (10 pts)**

**CIS 163 – Computer Science II**

**Project 3: A “Reserve Camping” program**

|  |  |
| --- | --- |
| Student Name |  |
| Date Submitted, Days Late, Late Penalty |  |

|  |  |  |
| --- | --- | --- |
| **Graded Item** | **Points Assigned** | **Points Secured** |
| Javadoc Comments and Coding Style/Technique  (<http://www.cis.gvsu.edu/studentsupport/javaguide>)   * Code Indentation (auto 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 * Every method uses a /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* separator * Overall layout, readability, No text wrap * Using /\*\* … / for each Instance variable * Has many inner “inner” comments | 10 |  |
| **Steps 1 – 7: Basic Functionality**   * RV and Tent classes * SiteModel class * GUICampingReg class * RV and Tent classes * Occupy RV (input via DialogCheckInRv) * Occupy Tent (input via DialogCheckInTent) * Cost using estimated number of days * Menu item to save items in store as a serialized file * Menu item to load items from a serialized file * Final UML class diagram | 55 |  |
| **Step 8: Save/Load As Text**   * Menu item to save items in store as a text file * Menu item to load items from a text file | 10 |  |
| **Step 9** Checking Out Menu and actual cost | 7 |  |
| **Step 10 – 11: Search Functionality**   * All sites are occupied. * Returns all sites with the number of days left | 5  5 |  |
| **Step 12: Your new feature** | 8 |  |
| **Total** | **100** |  |

**Comments:**