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

|  |  |
| --- | --- |
| Student Name | Zach Hopman |
| 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** |  |

**Additional Comments: (How many hours did you spend developing this project?)**

Code

package package1;

import java.io.Serializable;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.GregorianCalendar;

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;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Constructor that sets all varibles equal to paramaters

\* @param nameReserving the name reserving campground site

\* @param checkin the date of checkin

\* @param daysStaying the number of days staying

\* @param checkOutOn the date checking out

\* @param siteNumber the number of the site

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public Site(String nameReserving, GregorianCalendar checkIn,

int daysStaying, GregorianCalendar checkOutOn,

int siteNumber) {

super();

this.nameReserving = nameReserving;

this.checkIn = checkIn;

this.daysStaying = daysStaying;

this.checkOutOn = checkOutOn;

this.siteNumber = siteNumber;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Constructor that sets all varibles equal to paramaters

\* @param nameReserving the name reserving campground site

\* @param checkin the date of checkin

\* @param daysStaying the number of days staying

\* @param siteNumber the number of the site

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// constructor with no checkOut

public Site(String nameReserving, GregorianCalendar checkIn,

int daysStaying, int siteNumber) {

super();

this.nameReserving = nameReserving;

this.checkIn = checkIn;

this.daysStaying = daysStaying;

this.siteNumber = siteNumber;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Constructor that creates a default site

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public Site() {

super();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*used to recieve the name of person reserving site

\*@return nameReserving

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public String getNameReserving() {

return nameReserving;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*used to set the Name of the person Reserving the site

\*@param nameReserving the name of person reserving

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void setNameReserving(String nameReserving) {

this.nameReserving = nameReserving;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the date checked in

\*@return checkIn

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public GregorianCalendar getCheckIn() {

return checkIn;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sets the CheckIn value

\*@param checkIn the date checked in

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void setCheckIn(GregorianCalendar checkIn) {

this.checkIn = checkIn;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the number of days staying

\*@return daysStaying

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public int getDaysStaying() {

return daysStaying;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sets the number of days staying

\*@param daysStaying

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void setDaysStaying(int daysStaying) {

this.daysStaying = daysStaying;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the check out date

\*@return checkOutOn

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public GregorianCalendar getCheckOutOn() {

return checkOutOn;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sets the check out date

\*@param checkOutOn

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void setCheckOutOn(GregorianCalendar checkOutOn) {

this.checkOutOn = checkOutOn;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the site number

\*@reutnr siteNumber

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public int getSiteNumber() {

return siteNumber;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sets the site number

\*@param siteNumber

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void setSiteNumber(int siteNumber) {

this.siteNumber = siteNumber;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the amount of money the camper is expected to owe

\*@return daysStaying \* 30;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public double CheckIn() {

return daysStaying \* 30;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*calculates the total amount of owed after the camper checks out

\*@param checkout the date checking out

\*@return the number of days times 30

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public double CheckOut(String checkout) throws ParseException {

SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");

String checkin = sdf.format(checkIn.getTime());

final long DAY\_IN\_MILLIS = 1000 \* 60 \* 60 \* 24;

int diffInDays = (int) ((sdf.parse(checkout).getTime()

- sdf.parse(

checkin).getTime()) / DAY\_IN\_MILLIS);

return diffInDays \* 30;

}

}

package package1;

import java.text.ParseException;

import java.util.GregorianCalendar;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

one of two options to be checked into the campsite, tent has a unique

variable of number of tenters.

@author Zach Hopman

@version March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class Tent extends Site {

private static final long serialVersionUID = 1L;

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

private int numOfTenters;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Constructor that intilizes all variable to the values given in

\*paramaters

\*@param nameReserving name of the person reserving

\*@param checkIn the date checked in

\*@param daysStaying the number of days staying

\*@param siteNumber the site number

\*@param numOfTenters the number of tenters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public Tent(String nameReserving, GregorianCalendar checkIn,

int daysStaying, int siteNumber, int numOfTenters) {

super(nameReserving, checkIn, daysStaying, siteNumber);

this.numOfTenters = numOfTenters;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Constructor that intilizes all variable to the values given in

\*paramaters

\*@param nameReserving name of the person reserving

\*@param checkIn the date checked in

\*@param checkOutOn the date checking out

\*@param daysStaying the number of days staying

\*@param siteNumber the site number

\*@param numOfTenters the number of tenters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public Tent(String nameReserving, GregorianCalendar checkIn,

int daysStaying, GregorianCalendar checkOutOn,

int siteNumber,

int numOfTenters) {

super(nameReserving, checkIn, daysStaying, checkOutOn,

siteNumber);

this.numOfTenters = numOfTenters;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Default constructor that creates a tent with default values

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public Tent() {

super();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns number of tenters

\*@return the number of tenters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public int getNumOfTenters() {

return numOfTenters;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sets the number of tenters

\*@param numOfTenters the number of tenters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void setNumOfTenters(int numOfTenters) {

this.numOfTenters = numOfTenters;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the amount expected to owe

\*@return the days staying times the number of tenters times 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public double CheckIn() {

return (super.CheckIn() / 10) \* numOfTenters;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*calculates the amount owed after a tenter checks out

\*@return the site checkout value / 10 time number of tenters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public double CheckOut(String checkout) throws ParseException{

return (super.CheckOut(checkout)/10) \* numOfTenters;

}

}

**package** package1;

**import** java.util.GregorianCalendar;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

one of two options to check into a campsite, RV has a unique varible

of the power or amps needed.

**@author** Zach Hopman

**@version** March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **class** RV **extends** Site {

**private** **static** **final** **long** *serialVersionUID* = 1L;

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

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Constructor that intilizes all variable to the values given in

\*paramaters

\***@param** nameReserving name of the person reserving

\***@param** checkIn the date checked in

\***@param** daysStaying the number of days staying

\***@param** siteNumber the site number

\***@param** power the amount of power needed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** RV(String nameReserving, GregorianCalendar checkIn,

**int** daysStaying,

**int** siteNumber, **int** power) {

**super**(nameReserving, checkIn, daysStaying, siteNumber);

**this**.power = power;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Constructor that intilizes all variable to the values given in

\*paramaters

\***@param** nameReserving name of the person reserving

\***@param** checkIn the date checked in

\***@param** daysStaying the number of days staying

\***@param** checkOutOn the date checking out

\***@param** siteNumber the site number

\***@param** numOfTenters the number of tenters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** RV(String nameReserving, GregorianCalendar checkIn,

**int** daysStaying,

GregorianCalendar checkOutOn, **int** siteNumber, **int** power) {

**super**(nameReserving, checkIn, daysStaying, checkOutOn,

siteNumber);

**this**.power = power;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the default constructor for a RV

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** RV() {

**super**();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the amount of power

\***@return** power

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **int** getPower() {

**return** power;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sets the amount of power

\***@param** power the amount of power

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** setPower(**int** power) {

**this**.power = power;

}

}

package package1;

import java.awt.HeadlessException;

import java.awt.event.\*;

import java.text.ParseException;

import java.util.\*;

import javax.swing.\*;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The graphical representation of the current and future residents of the

campground.

@author Zach Hopman

@version March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class GUICampingReg extends JFrame implements ActionListener {

private static final long serialVersionUID = 1L;

/\*\*the menu bar for the GUI \*/

private JMenuBar menus;

/\*\*file drop down menu \*/

private JMenu fileMenu;

/\*\*check in drop down menu \*/

private JMenu CheckInMenu;

/\*\*check out drop down menu \*/

private JMenu CheckOutMenu;

// fileMenu

/\*\*opens serizable save \*/

private JMenuItem openSerItem;

/\*\*exits the program \*/

private JMenuItem exitItem;

/\*\*saves current log as serizable file \*/

private JMenuItem saveSerItem;

/\*\*opens text file save \*/

private JMenuItem openTextItem;

/\*\*saves current log as text file \*/

private JMenuItem saveTextItem;

/\*\*opens new window displaying info on current residents \*/

private JMenuItem statusItem;

/\*\*sorts the campers by name\*/

private JMenuItem sort;

// CheckInMenu

/\*\*opens window to check in a tent \*/

private JMenuItem CheckInTent;

/\*\*opens a window to check in an rv \*/

private JMenuItem CheckInRV;

// CheckOutMenu

/\*\*checks out the currenttly selected camper \*/

private JMenuItem CheckOut;

/\*\*the table to hold all information on campers \*/

private JTable JTableArea;

/\*\*the generic model for a camper \*/

private SiteModel model;

/\*\*the scroll pane for the JTable \*/

private JScrollPane ScrollPane;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the Constructor used to create the GUI window. sets up the menubar

\*and the JTable.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private GUICampingReg() {

// file menu

fileMenu = new JMenu("File");

openSerItem = new JMenuItem("Open Serial");

saveSerItem = new JMenuItem("Save Serial");

exitItem = new JMenuItem("Exit");

openTextItem = new JMenuItem("Open Text");

saveTextItem = new JMenuItem("Save Text");

statusItem = new JMenuItem("Status");

sort = new JMenuItem("Sort");

fileMenu.add(openSerItem);

fileMenu.add(saveSerItem);

fileMenu.add(exitItem);

fileMenu.add(openTextItem);

fileMenu.add(saveTextItem);

fileMenu.add(statusItem);

fileMenu.add(sort);

openSerItem.addActionListener(this);

exitItem.addActionListener(this);

saveSerItem.addActionListener(this);

openTextItem.addActionListener(this);

saveTextItem.addActionListener(this);

statusItem.addActionListener(this);

sort.addActionListener(this);

// check in menu

CheckInMenu = new JMenu("Check In");

CheckInTent = new JMenuItem("Check in Tent");

CheckInRV = new JMenuItem("Check in RV");

CheckInMenu.add(CheckInTent);

CheckInMenu.add(CheckInRV);

// check out menu

CheckOutMenu = new JMenu("Check Out");

CheckOut = new JMenuItem("Check out");

CheckOutMenu.add(CheckOut);

CheckInTent.addActionListener(this);

CheckInRV.addActionListener(this);

CheckOut.addActionListener(this);

menus = new JMenuBar();

menus.add(fileMenu);

menus.add(CheckInMenu);

menus.add(CheckOutMenu);

setJMenuBar(menus);

// table

model = new SiteModel();

JTableArea = new JTable(model);

ScrollPane = new JScrollPane(JTableArea);

add(ScrollPane);

setVisible(true);

setSize(600, 400);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the main method to be ran

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void main(String[] args) {

new GUICampingReg();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the method for all actions. ie buttons and menuItems.

\*@param e the action

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void actionPerformed(ActionEvent e) {

if (CheckOut == e.getSource()) {

int index = JTableArea.getSelectedRow();

if (index == -1)

JOptionPane.showMessageDialog(this, "Empty");

else{

Site s = model.getSites(JTableArea.getSelectedRow());

String CheckOutDate = JOptionPane.showInputDialog

(this," mm/dd/yyyy","Enter the Date",1);

try {

JOptionPane.showMessageDialog

(JTableArea,"You owe $"+ s.CheckOut(CheckOutDate));

} catch (HeadlessException e1) {

e1.printStackTrace();

} catch (ParseException e1) {

e1.printStackTrace();

}

model.removeSite(index);

}

}

if (openSerItem == e.getSource()) {

model.loadDatabase("testDB");

}

if (saveSerItem == e.getSource()) {

model.saveDatabase("testDB");

}

if (openTextItem == e.getSource()) {

model.loadFromText("testDBtext");

}

if (saveTextItem == e.getSource()) {

model.saveAsText("testDBtext");

}

if (exitItem == e.getSource()) {

System.exit(0);

}

if (CheckInTent == e.getSource()) {

Tent t = new Tent();

DialogCheckInTent x = new DialogCheckInTent(this, t);

if (x.getCloseStatus() == x.OK) {

if (model.isEmpty(t.getSiteNumber())) {

model.addSite(t);

JOptionPane.showMessageDialog(JTableArea,

"You will owe: $"

+ model.CheckIn());

}

else

JOptionPane.showMessageDialog(JTableArea,"Site "

+t.getSiteNumber()

+" is taken.");

}

}

if (CheckInRV == e.getSource()) {

RV r = new RV();

DialogCheckInRV x = new DialogCheckInRV(this, r);

if (x.getCloseStatus() == x.OK) {

if (model.isEmpty(r.getSiteNumber())) {

model.addSite(r);

JOptionPane.showMessageDialog(JTableArea,

"You will owe: $"

+ model.CheckIn());

}

else

JOptionPane.showMessageDialog(JTableArea,"Site "+

r.getSiteNumber()

+" is taken.");

}

}

//status

if (statusItem == e.getSource()){

String CheckOutDate = JOptionPane.showInputDialog

(this," mm/dd/yyyy","Enter the Date",1);

try {

DialogGetDateStatus x = new DialogGetDateStatus

(this, CheckOutDate, model);

} catch (ParseException e1) {

e1.printStackTrace();

}

}

if (sort == e.getSource()){

model.Sort();

repaint();

}

}

}

package package1;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.text.\*;

import java.util.\*;

import javax.swing.\*;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The window that opens when checking in a RV

@author Zach Hopman

@version March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class DialogCheckInRV extends JDialog implements ActionListener{

/\*\*textfield for the name of camper \*/

private JTextField name;

/\*\*textfield for the date \*/

private JTextField OccupyedOn;

/\*\*textfield for expected number of days staying \*/

private JTextField staying;

/\*\*textfield for the sitenumber \*/

private JTextField siteNumber;

/\*\*textfield for the amount of power needed \*/

private JTextField Power;

/\*\*button to add the camper \*/

private JButton okButton;

/\*\*button to cancel the check in \*/

private JButton cancelButton;

/\*\*int that holds either 1 or 0 to determine if window is open \*/

private int closeStatus;

/\*\*the site being checked in \*/

private Site unit;

public static final int OK = 0;

public static final int CANCEL = 1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*The constructor to set up and display the window for checking in

\*a RV

\*@param paOccupy the Jframe

\*@param d the site being added

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public DialogCheckInRV(JFrame paOccupy, Site d) {

super(paOccupy, true);

setTitle("Check In RV");

closeStatus = CANCEL;

setSize(400, 200);

unit = d;

// prevent user from closing window

setDefaultCloseOperation(WindowConstants.DO\_NOTHING\_ON\_CLOSE);

// instantiate and display text fields

JPanel textPanel = new JPanel();

textPanel.setLayout(new GridLayout(6, 2));

textPanel.add(new JLabel("Name:"));

name = new JTextField("John Doe", 30);

textPanel.add(name);

textPanel.add(new JLabel("Amount of Power: "));

Power = new JTextField("30", 30);

textPanel.add(Power);

textPanel.add(new JLabel("Site Number: "));

siteNumber = new JTextField("2", 30);

textPanel.add(siteNumber);

textPanel.add(new JLabel("Ocupied on Date: "));

OccupyedOn = new JTextField("03/18/2015", 30);

textPanel.add(OccupyedOn);

textPanel.add(new JLabel("Days Planning to Stay: "));

staying = new JTextField("1", 30);

textPanel.add(staying);

getContentPane().add(textPanel, BorderLayout.CENTER);

// Instantiate and display two buttons

okButton = new JButton("OK");

cancelButton = new JButton("Cancel");

JPanel buttonPanel = new JPanel();

buttonPanel.add(okButton);

buttonPanel.add(cancelButton);

getContentPane().add(buttonPanel, BorderLayout.SOUTH);

okButton.addActionListener(this);

cancelButton.addActionListener(this);

setSize(300, 300);

setVisible(true);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the method for all actions. ie buttons and menuItems.

\*@param e the action

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void actionPerformed(ActionEvent e) {

JButton button = (JButton) e.getSource();

// if OK clicked the fill the object

if (button == okButton) {

closeStatus = OK;

SimpleDateFormat df = new SimpleDateFormat("MM/dd/yyyy");

Date d;

try {

d = df.parse(OccupyedOn.getText());

GregorianCalendar opened = new GregorianCalendar();

opened.setTime(d);

int DayStay = Integer.parseInt(staying.getText());

int pow = Integer.parseInt(Power.getText());

int sitenum = Integer.parseInt(siteNumber.getText());

unit.setNameReserving(name.getText());

unit.setCheckIn(opened);

unit.setDaysStaying(DayStay);

unit.setSiteNumber(sitenum);

((RV) unit).setPower(pow);

} catch (ParseException e1) {

e1.printStackTrace();

}

}

// make the dialog disappear

dispose();

}

public int getCloseStatus() {

return closeStatus;

}

}

package package1;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.text.\*;

import java.util.\*;

import javax.swing.\*;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The window that opens when checking in a tent.

@author Zach Hopman

@version March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class DialogCheckInTent extends JDialog implements ActionListener{

/\*\*textfield for the name of camper \*/

private JTextField name;

/\*\*textfield for the date \*/

private JTextField OccupyedOn;

/\*\*textfield for expected number of days staying \*/

private JTextField staying;

/\*\*textfield for the sitenumber \*/

private JTextField siteNumber;

/\*\*textfield for the number of tenters \*/

private JTextField Tenters;

/\*\*button to add the camper \*/

private JButton okButton;

/\*\*button to cancel the check in \*/

private JButton cancelButton;

/\*\*int that holds either 1 or 0 to determine if window is open \*/

private int closeStatus;

/\*\*the site being checked in \*/

private Site unit;

public static final int OK = 0;

public static final int CANCEL = 1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*The constructor to set up and display the window for checking in

\*a tent

\*@param paOccupy the Jframe

\*@param d the site being added

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public DialogCheckInTent(JFrame paOccupy, Site d) {

super(paOccupy, true);

setTitle("Check In Tent");

closeStatus = CANCEL;

setSize(400, 200);

unit = d;

// prevent user from closing window

setDefaultCloseOperation(WindowConstants.DO\_NOTHING\_ON\_CLOSE);

// instantiate and display text fields

JPanel textPanel = new JPanel();

textPanel.setLayout(new GridLayout(6, 2));

textPanel.add(new JLabel("Name:"));

name = new JTextField("John Doe", 30);

textPanel.add(name);

textPanel.add(new JLabel("Number of Tenters: "));

Tenters = new JTextField("1", 30);

textPanel.add(Tenters);

textPanel.add(new JLabel("Site Number: "));

siteNumber = new JTextField("1", 30);

textPanel.add(siteNumber);

textPanel.add(new JLabel("Ocupied on Date: "));

OccupyedOn = new JTextField("03/18/2015", 30);

textPanel.add(OccupyedOn);

textPanel.add(new JLabel("Days Planning to Stay: "));

staying = new JTextField("1", 30);

textPanel.add(staying);

getContentPane().add(textPanel, BorderLayout.CENTER);

// Instantiate and display two buttons

okButton = new JButton("OK");

cancelButton = new JButton("Cancel");

JPanel buttonPanel = new JPanel();

buttonPanel.add(okButton);

buttonPanel.add(cancelButton);

getContentPane().add(buttonPanel, BorderLayout.SOUTH);

okButton.addActionListener(this);

cancelButton.addActionListener(this);

setSize(300, 300);

setVisible(true);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the method for all actions. ie buttons and menuItems.

\*@param e the action

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void actionPerformed(ActionEvent e) {

JButton button = (JButton) e.getSource();

// if OK clicked the fill the object

if (button == okButton) {

// save the information in the object

closeStatus = OK;

SimpleDateFormat df = new SimpleDateFormat("MM/dd/yyyy");

Date d;

try {

d = df.parse(OccupyedOn.getText());

GregorianCalendar opened = new GregorianCalendar();

opened.setTime(d);

int DayStay = Integer.parseInt(staying.getText());

int tenters = Integer.parseInt(Tenters.getText());

int sitenum = Integer.parseInt(siteNumber.getText());

unit.setNameReserving(name.getText());

unit.setCheckIn(opened);

unit.setDaysStaying(DayStay);

unit.setSiteNumber(sitenum);

((Tent) unit).setNumOfTenters(tenters);

} catch (ParseException e1) {

e1.printStackTrace();

}

}

// make the dialog disappear

dispose();

}

public int getCloseStatus() {

return closeStatus;

}

}

**package** package1;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.PrintWriter;

**import** java.text.DateFormat;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.Date;

**import** java.util.GregorianCalendar;

**import** java.util.Scanner;

**import** javax.swing.table.AbstractTableModel;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The generic model for the Jtable to utilize to display information in

the GUI.

**@author** Zach Hopman

**@version** March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **class** SiteModel **extends** AbstractTableModel {

/\*\*arraylist of all the campers \*/

**private** ArrayList<Site> listSites;

/\*\*array of strings with all the coulumn names for the table \*/

**private** String[] columnNames = { "Name Reserving", "Checked in",

"Days Staying", "Site #", "Tent/RV info" };

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*the defualt constructor for the siteModel

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** SiteModel() {

**super**();

listSites = **new** ArrayList<Site>();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*checks in the camper and calls the checkIn method from the site

\***@return** the amount owed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **double** CheckIn() {

**return** getSites(listSites.size() - 1).CheckIn();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*sorts the JTable based on the name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** ArrayList<Site> Sort(){

Collections.*sort*(listSites, **new** Comparator<Site>(){

@Override

**public** **int** compare(Site site1, Site site2) {

**return** site1.getNameReserving().compareTo

(site2.getNameReserving());

}

});

**return** listSites;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*checks if the site is empty

\***@return** true if empty otherwise false

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **boolean** isEmpty(**int** site) {

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

**if** (getSites(i).getSiteNumber() == site)

**return** **false**;

}

**return** **true**;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*removes the site

\***@param** i the site number

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** removeSite(**int** i) {

listSites.remove(i);

fireTableRowsDeleted(0, listSites.size());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*adds the site

\***@param** the site number

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** addSite(Site a) {

listSites.add(a);

fireTableRowsInserted(0, listSites.size());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the site

\***@param** the site number

\***@return** the site

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** Site getSites(**int** i) {

**return** listSites.get(i);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the amount of campers

\***@return** the size of the arraylist

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **int** getSize() {

**return** listSites.size();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the coulumn name for the JTable

\***@param** col the coloumn number you want

\***@return** the name of the coloumn

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** String getColumnName(**int** col) {

**return** columnNames[col];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the number of coulumns

\***@return** the length of column array

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **int** getColumnCount() {

**return** columnNames.length;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns the number of rows

\***@return** the size of the arraylist of sites

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **int** getRowCount() {

**return** listSites.size();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*returns each cell of the JTable based on the paramaters

\***@param** row the row number

\***@param** col the column number

\***@return** the site at the specified cell

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

**switch** (col) {

**case** 0:

**return** (listSites.get(row).getNameReserving());

**case** 1:

**return** (DateFormat.*getDateInstance*(DateFormat.*SHORT*)

.format(listSites.get(row).getCheckIn().getTime()));

**case** 2:

**return** (listSites.get(row).getDaysStaying());

**case** 3:

**return** (listSites.get(row).getSiteNumber());

**case** 4:

**if** (listSites.get(row) **instanceof** Tent)

**return** ((Tent) listSites.get(row)).getNumOfTenters();

**else**

**return** ((RV) listSites.get(row)).getPower();

**default**:

**return** **null**;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*saves JTable as a serizable file

\***@param** filename the name of the file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** saveDatabase(String filename) {

**try** {

FileOutputStream fos = **new** FileOutputStream(filename);

ObjectOutputStream os = **new** ObjectOutputStream(fos);

os.writeObject(listSites);

os.close();

} **catch** (IOException ex) {

ex.printStackTrace();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*loads the serizable file into the JTable

\***@param** filename the name of the file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** loadDatabase(String filename) {

**try** {

FileInputStream fis = **new** FileInputStream(filename);

ObjectInputStream is = **new** ObjectInputStream(fis);

listSites = (ArrayList<Site>) is.readObject();

fireTableRowsInserted(0, listSites.size() - 1);

is.close();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*saves the JTable as a text file

\***@param** filename the name of the file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **boolean** saveAsText(String filename) {

**if** (filename.equals("")) {

**return** **false**;

}

**try** {

PrintWriter out = **new** PrintWriter(**new** BufferedWriter(

**new** FileWriter(filename)));

out.println(listSites.size());

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

Site site = listSites.get(i);

out.println(site.getNameReserving());

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

.format(site.getCheckIn().getTime()));

out.println(site.getDaysStaying());

out.println(site.getSiteNumber());

**if** (site **instanceof** Tent)

out.println(((Tent) site).getNumOfTenters());

**else**

out.println(((RV) site).getPower());

}

out.close();

**return** **true**;

} **catch** (IOException ex) {

**return** **false**;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*loads the JTable from a text file

\***@param** filename the name of the file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** loadFromText(String filename) {

listSites.clear();

fireTableRowsDeleted(0, listSites.size());

**try** {

Scanner scanner = **new** Scanner(**new** File(filename));

// should clear the arrayList and screen....

**int** count = Integer.*parseInt*(scanner.nextLine().trim());

**for** (**int** i = 0; i < count; i++) {

// gets name

String nameReserving = scanner.nextLine().trim();

// gets checkIn

GregorianCalendar checkIn = **null**;

**try** {

DateFormat formatter = **new** SimpleDateFormat

("MM/dd/yyyy");

Date date = formatter.parse(scanner.nextLine().

trim());

checkIn = **new** GregorianCalendar();

checkIn.setTime(date);

} **catch** (ParseException ex) {

ex.printStackTrace();

}

// gets daysStaying

**int** daysStaying = Integer.*parseInt*(scanner.nextLine().

trim());

// gets siteNumber

**int** siteNumber = Integer.*parseInt*(scanner.nextLine().

trim());

// gets Tenters

**int** numOfTenters = Integer.*parseInt*(scanner.nextLine().

trim());

**if** (numOfTenters < 20) {

Tent t;

t = **new** Tent(nameReserving, checkIn, daysStaying,

siteNumber, numOfTenters);

listSites.add(t);

fireTableRowsInserted(listSites.size() - 1,

listSites.size() - 1);

} **else** {

RV r;

**int** power = numOfTenters;

r = **new** RV(nameReserving, checkIn, daysStaying,

siteNumber,

power);

listSites.add(r);

fireTableRowsInserted(listSites.size() - 1,

listSites.size() - 1);

}

}

scanner.close();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

}

package package1;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The window that opens when checking status of campers.

@author Zach Hopman

@version March 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class DialogGetDateStatus extends JDialog {

/\*\*textfield for the name of camper \*/

private JTextField name;

/\*\*textfield for the date checked in \*/

private JTextField OccupyedOn;

/\*\*textfield for the number of days staying \*/

private JTextField staying;

/\*\*textfield for the sitenumber \*/

private JTextField siteNumber;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Constructor for the status menu. displays all campers in a list

\*and calculates the days remaining.

\*@param paOccupy the Jframe to add to.

\*@param date the checkOut date entered

\*@param model the JTable representation of the camper

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public DialogGetDateStatus(JFrame paOccupy, String date,

SiteModel model) throws ParseException{

setTitle("Status");

JPanel textPanel = new JPanel();

textPanel.setLayout(new GridLayout(8, 1));

for(int i = 0; i<model.getRowCount(); i++){

SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");

String checkin = sdf.format((model.getSites(i).getCheckIn().

getTime()));

final long DAY\_IN\_MILLIS = 1000 \* 60 \* 60 \* 24;

int diffInDays = (int)((sdf.parse(date).getTime()-sdf.parse(

checkin).getTime()) / DAY\_IN\_MILLIS);

int dayremain = (int) model.getValueAt(i, 2) - diffInDays;

textPanel.add(new JLabel(model.getValueAt(i, 0)+

" Checked in: "+model.getValueAt(i, 1)+

" Site #: "+model.getValueAt(i,3)+

" Estimated days: "+model.getValueAt(i, 2)+

" Days remaining: "+dayremain));

}

getContentPane().add(textPanel, BorderLayout.CENTER);

setSize(500, 300);

setVisible(true);

}

}