Parking Lot Java Project

By Colin Kula

Table of contents:

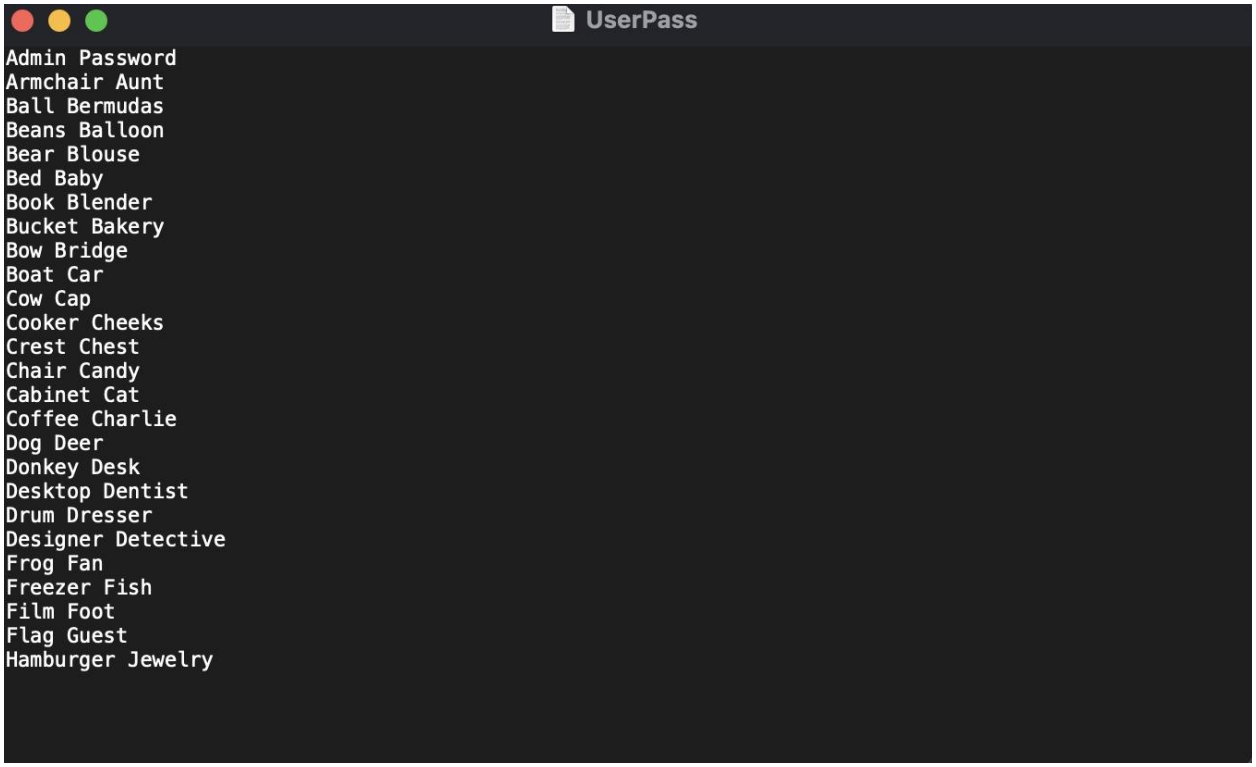
Introduction

This project is designed to represent a parking lot management system. A login screen is shown once the program is run and only specific usernames and passwords will let the user log in. A text file of all the usernames and passwords is searched through to ensure the information put in by the user will let them access the next screen. After logging in, the user is shown a screen that provides many options that can be used to manage a parking lot. Buttons serve as the input for many different tasks: generating a parking report, parking a vehicle, removing a vehicle, holding a parking space, calculating the total fees, generating a parking status of the current parking lot, changing parking lots and changing users.

The goal of this project is to serve as a back and front end service that allows the user to manage a parking lot. The FrontEnd class creates a GUI that allows the user to choose which of the previously stated tasks they want to perform. Classes called User, ParkingLot and ParkingSpace provide the backend for the actions being performed, and the Manageable interface is used to provide abstract methods for the User class.

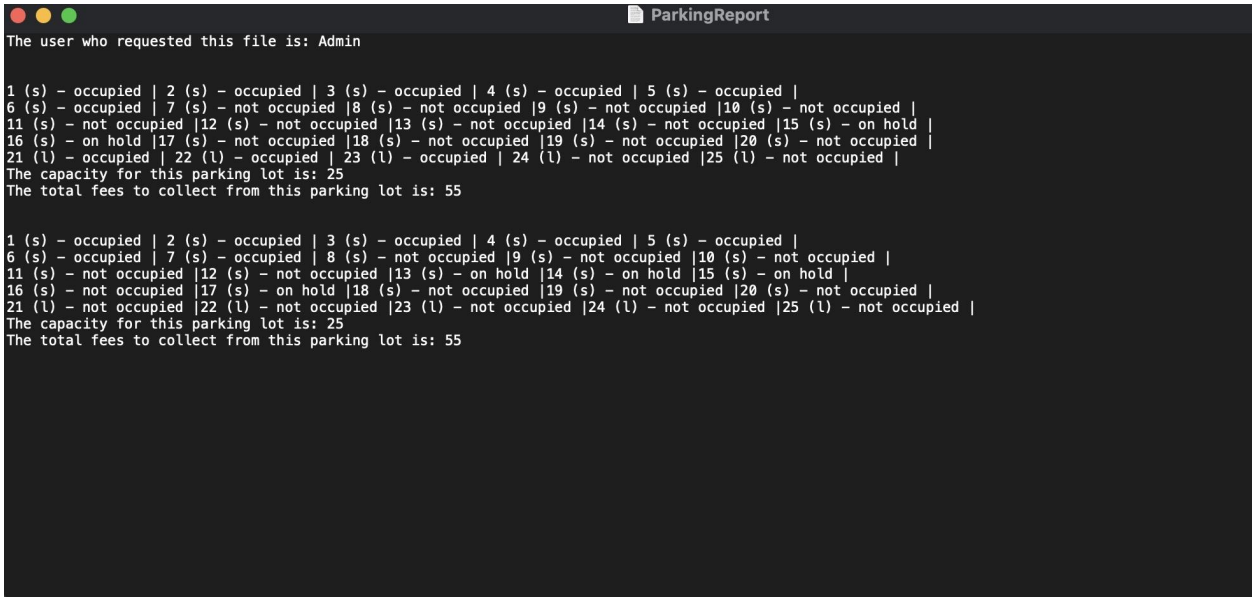
Description of text files

UserPass.txt (to be imported): The UserPass text file stores all of the username and password combinations that will allow the current user of the program to login. This text file can be changed to hold specific employee login information, but currently uses two random words for each user's username and password. Text fields on the login screen are used to access the information the user wants to login with, and the method checkUserAndPass(), in the User class, ensures that the information is found somewhere in the text file. The following image shows how the current text file looks.



ParkingReport.txt (to be exported): The ParkingReport text file is used to hold all of the information regarding the user, the parking lot and the parking spaces. The method generateParkingReport(), in the User class, loops through all of the parking lots accessed and

prints their toString() methods to the text file. Whenever the user decides to generate the parking report, they will be able to view information such as who the current user is, each parking lot, each parking space in the parking lot, which parking spaces are which size, if the parking spaces are occupied or being held, the capacity of each parking lot and the total fees that need to be collected. The following image shows how a parking report might look.



Manageable Interface

The manageable interface is used to

Constructors: N/A

Instance variables: N/A

Abstract methods:

Public void storeAllUsers(ArrayList<User> users) throws FileNotFoundException - accesses the UserPass text file, gets each of the user's username and password, and creates User objects and adds them to an array list

Public User changeUser(String username, String password, ArrayList<User> users) - returns the User object with given username and password if it exists

Public void generateParkingReport() - fills the ParkingReport text file with information like who the current user is, the parking lot information and all the information related to the parking spaces in each parking lot

Public String toString() - returns a String representation of the User object

Public Object clone() - returns a deep clone of the user object

User class implements Manageable

The User class is used to create an instance of a User object. Each User has a username and password and can perform many different actions including generating a new password for the current user, resetting the current password, checking if the login information typed into the login textboxes are correct, importing all of the usernames and passwords from the UserPass file, changing to a different user and generating a parking report. The purpose of this class is to create instances of a user that can access the rest of the features of the program. Once logged in, the user can perform all the tasks on the home screen. This class also implements the manageable interface so that it can incorporate the abstract methods that will allow the user to manage the parking lot.

Constructors:

User(String username, String password) - creates a User object and initializes the username and password instance variables

User(User user) - creates a User object based off the attributes of another User object

Instance variables:

username [String] - holds the username of the user

password [String] - holds the password of the user

currentUserIndex [int] - holds the index of where the current user is in the array list that contains all of the users capable of logging in

parkingLots [ArrayList] - holds a list of all the ParkingLot objects that have been created

Methods:

Public String getUsername() - returns the username of the user

Public String getPassword() - returns the password of the user

Public String generateNewPassword() - returns a String containing a randomized new password for the current user

Public String resetPassword() - returns a String containing a randomized password

Public boolean checkUserAndPass(String username, String password, ArrayList<User> users) - returns a boolean that is true when the given username and password are correct and false when incorrect

Public void storeAllUsers(ArrayList<User> users) throws FileNotFoundException - accesses the UserPass text file, gets each of the user's username and password, and creates User objects and adds them to an array list

Public User changeUser(String username, String password, ArrayList<User> users) - returns the User object with given username and password if it exists

Public void generateParkingReport() - fills the ParkingReport text file with information like who the current user is, the parking lot information and all the information related to the parking spaces in each parking lot

Public String toString() - returns a String representation of the User object

Public Object clone() - returns a clone of the User object

ParkingLot class extends User

The ParkingLot class is used to create an instance of a ParkingLot object. Each ParkingLot has a capacity and array list of parking spaces and can perform many different actions including finding the number of available spaces left in the parking lot, finding the next available parking space and calculating the fees that need to be paid. The purpose of this class is to create instances of multiple parking lots that the User can manage. Since ParkingLot extends the User class, a parent and child relationship is formed and inheritance can occur.

Constructors:

ParkingLot(int capacity) - creates a ParkingLot object and initializes the capacity and spaces instance variables

ParkingLot(ParkingLot) - creates a ParkingLot object based off the attributes of another ParkingLot object

Instance variables:

spaces [ArrayList <ParkingSpace>] - holds all the parking spaces in the parking lot

capacity [int] - holds the total capacity of the parking lot

Methods:

Public int getCapacity() - returns the total capacity of the parking lot

Public int getNumberOfAvailableSpaces() - returns an int that represents the available parking spaces in the parking lot

Public String getAllSpecificAvailableSpaces() - returns a String that contains each parking spot and its attributes (parking spot number, size, and status)

Public int getNextAvailableSpace(char size) - returns the index of the next available parking space in the spaces arraylist

Public int calculateFees() - returns an int that represents the fees that currently need to be calculated

Public String toString() - returns a String representation if the ParkingLot object

Public ParkingLot clone() - returns a ParkingLot that has the same attributes of the current parking lot

ParkingSpace class

The ParkingSpace class is used to create an instance of a ParkingSpace object. Each ParkingLot has a space number and size and can perform many different actions including making the space occupied, putting it on hold, parking a vehicle in it, removing a vehicle from it and ensuring a vehicle fits in the spot. The purpose of this class is to create instances of multiple parking spaces that are held in a parking lot.

Constructors:

ParkingSpace(int spaceNumber, boolean isOccupied, boolean isHeld, char size) - creates a ParkingSpace object and initializes the spaceNumber, isOccupied, isHeld and size instance variables

ParkingSpace(ParkingSpace) - creates a ParkingLot object based off the attributes of another ParkingSpace object

Instance variables:

spaceNumber [int] - holds the unique number of each parking space

isOccupied [boolean] - holds whether the parking space is occupied or not

isHeld [boolean] - holds whether the parking space is held or not

size [char] - holds the size of the parking space

Methods:

Public char getSize() - returns a char that represents the size of the parking space

Public void setSize(char size) - sets the size of the parking space

Public int getSpaceNumber() - returns an int that represents the specific parking space number

Public boolean isOccupied() - returns a boolean that is true when there is a vehicle in the parking spot and false if not

Public void occupy() - sets the parking space to occupied

Public void free() - sets the parking space to not occupied

Public boolean isHeld() - returns a boolean that is true when the parking space is on hold and false if not

Public void holdSpace(ParkingLot parkinglot, int spaceNumber) - sets the parking space to on hold

Public boolean ensureVehicleFits(char size) - returns a boolean that is true if the vehicle fits and false if not

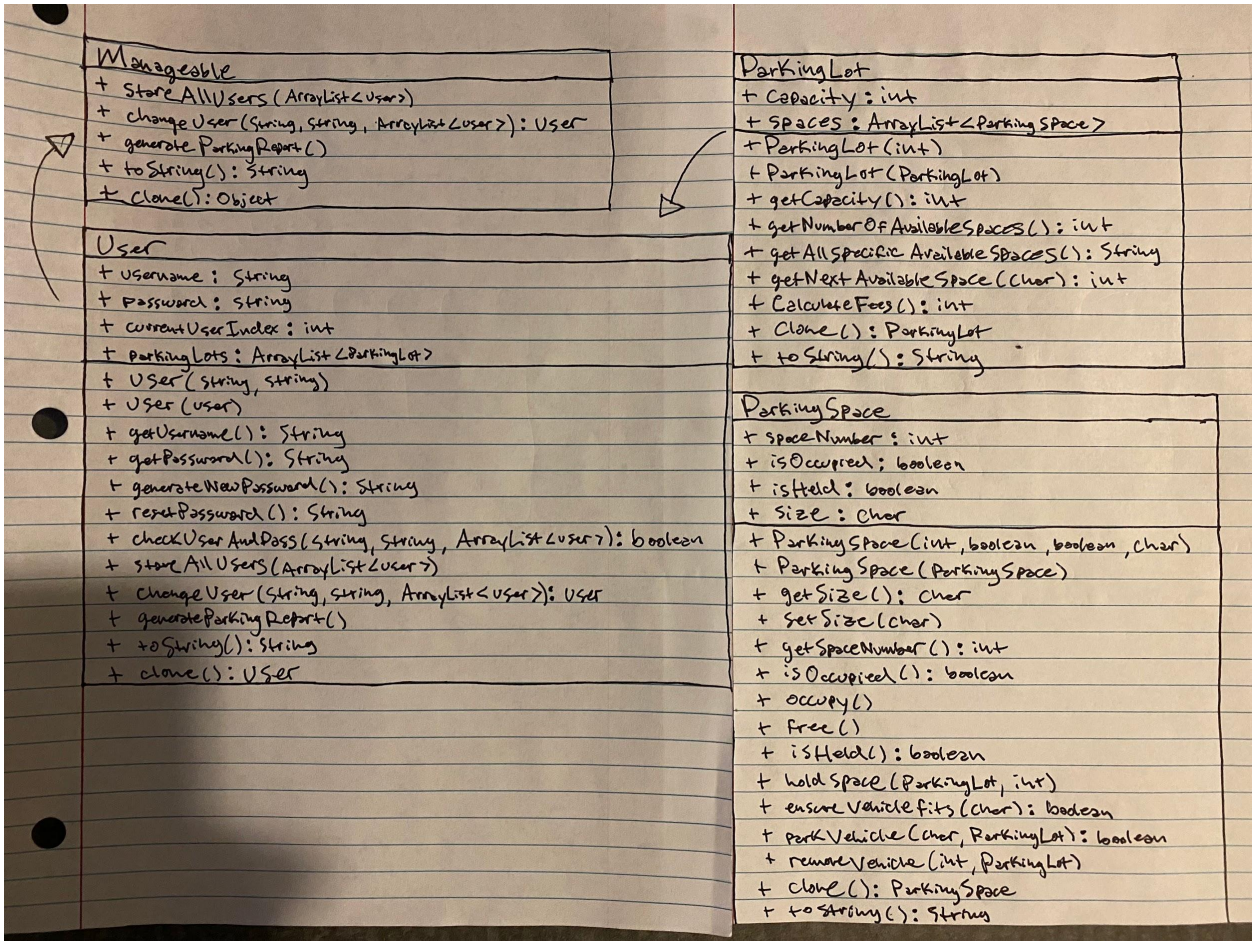
Public boolean parkVehicle(char size, ParkingLot parkingLot) - returns a boolean that is false when a specified vehicle can fit in an available parking space and true when it cannot

Public void removeVehicle(int spaceNumber, ParkingLot parkinglot) - removes a vehicle from the specified parking space

Public String toString() - returns a String representation if the ParkingSpace object

Public ParkingSpace clone() - returns a ParkingSpace that has the same attributes of the current parking space

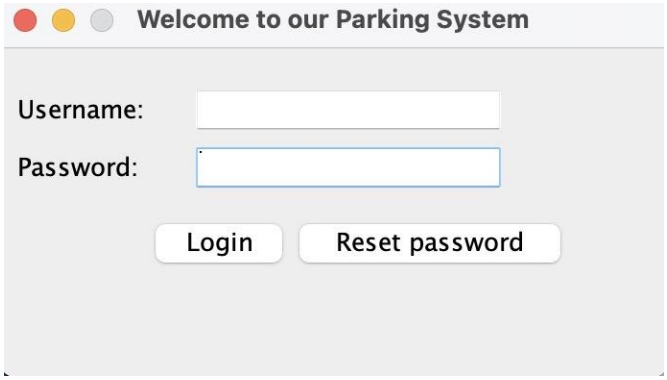
UML

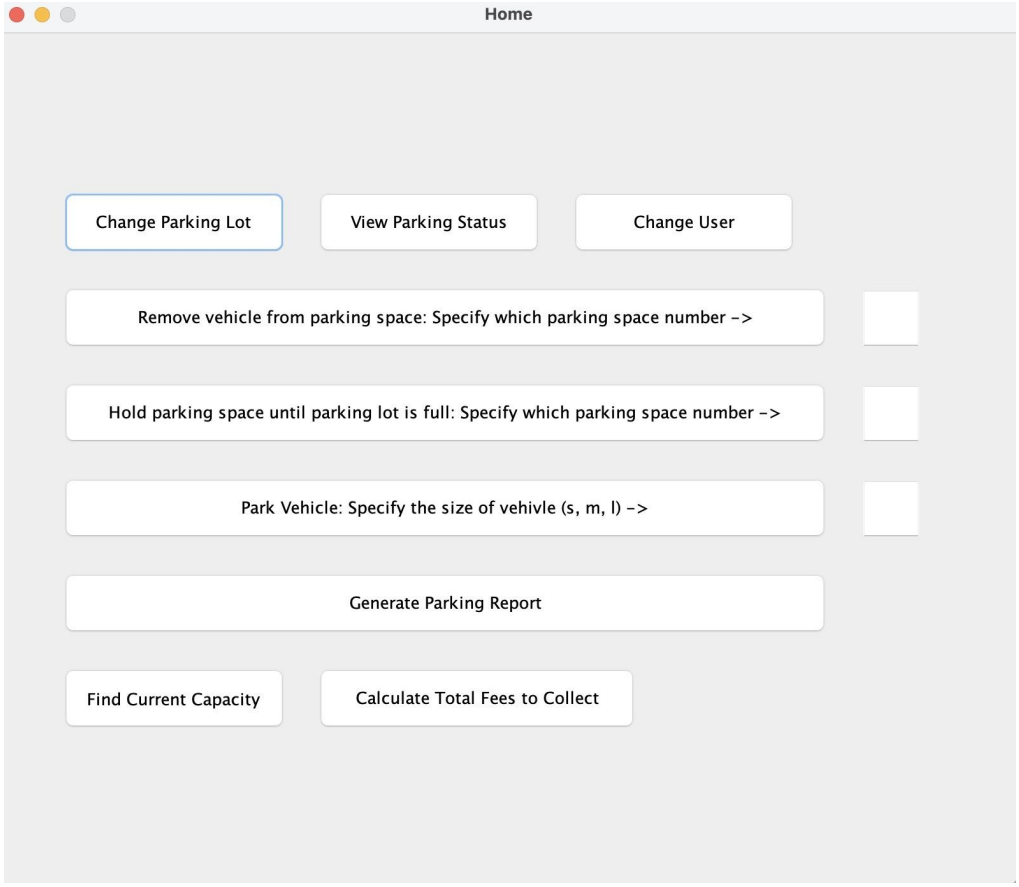

Manageable
+ storeAllUsers(ArrayList<User>)
+ changeUser(String, String, ArrayList<User>): User
+ generateParkingReport()
+ toString(): String
+ clone(): Object
User
+ username: String
+ password: String
+ currentUserIndex: int
+ parkingLots: ArrayList<ParkingLot>
+ User(String, String)
+ User(User)
+ getUsername(): String
+ getPassword(): String
+ generateNewPassword(): String
+ resetPassword(): String
+ checkUserAndPass(String, String, ArrayList<User>): boolean
+ storeAllUsers(ArrayList<User>)
+ changeUser(String, String, ArrayList<User>): User
+ generateParkingReport()
+ toString(): String
+ clone(): User
ParkingLot
+ capacity: int
+ spaces: ArrayList<ParkingSpace>
+ ParkingLot(int)
+ ParkingLot(ParkingLot)
+ getCapacity(): int
+ getNumberOfAvailableSpaces(): int
+ getAllSpecificAvailableSpaces(): String
+ getNextAvailableSpace(char): int
+ calculateFees(): int
+ clone(): ParkingLot
+ toString(): String
ParkingSpace
+ spaceNumber: int
+ isOccupied: boolean
+ isHeld: boolean
+ size: char
+ ParkingSpace(int, boolean, boolean, char)
+ ParkingSpace(ParkingSpace)
+ getSize(): char
+ setSize(char)
+ getSpaceNumber(): int
+ isOccupied(): boolean
+ occupy()
+ free()
+ isHeld(): boolean
+ holdSpace(ParkingLot, int)
+ ensureVehicleFits(char): boolean
+ parkVehicle(char, ParkingLot): boolean
+ removeVehicle(int, ParkingLot)
+ clone(): ParkingSpace
+ toString(): String

Testing and results

How to use the program -

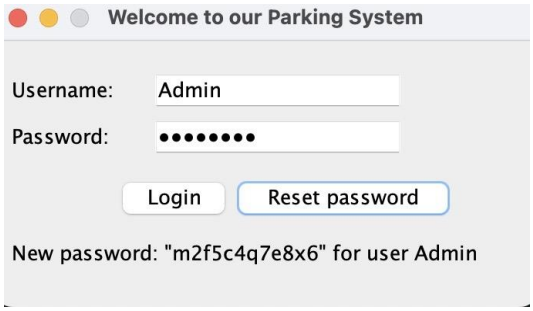
After initially running the program, a login screen is created. To login, a username and password from the text file “UserPass” must be entered or else you cannot make it to the home screen. The initial username and password to use is “Admin” and “Password”, respectively. There are multiple buttons that can be pressed to perform different tasks on both the login screen and home screen. Below is a picture of what the login and home screen look like.





What each button does -

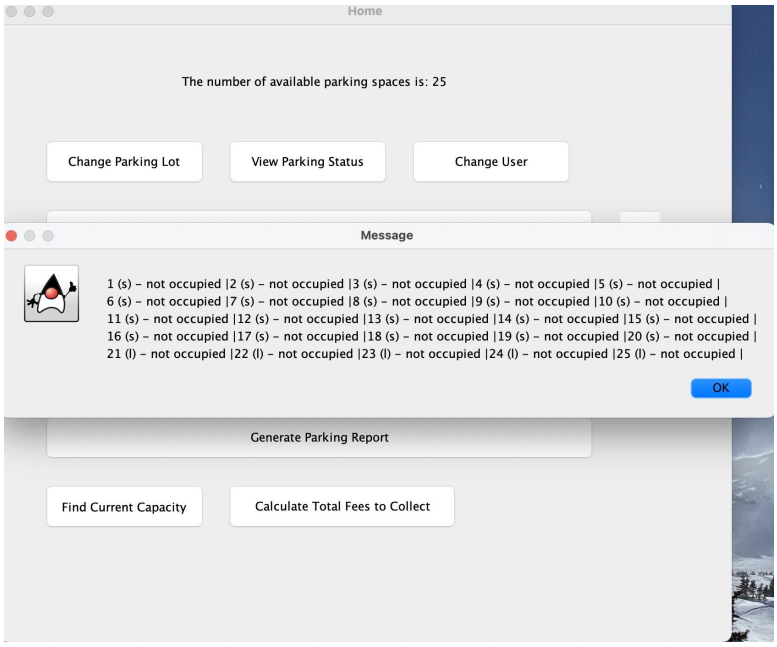
Login: Lets the user log in depending on what is put in the username and password text boxes.



Reset Password: Gives the user a new password and prints it to an output label.

Change Parking Lot: Creates a new parking lot that holds more parking spaces.

View Parking Status: Sets the result label with the number of available parking spaces and creates a new screen that shows the status of all the parking spaces.



Change User: Brings the user back to the login screen so that they can switch and use a different username and password

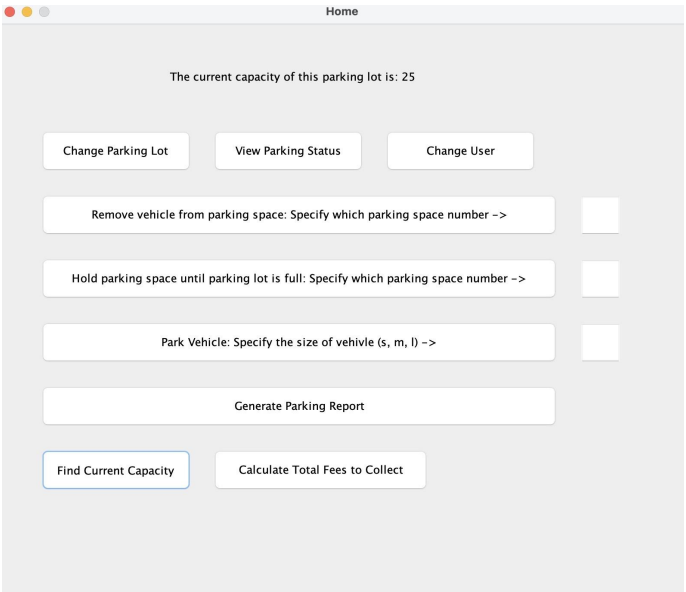
Remove vehicle: Makes the specified parking space empty. The textbox to the right of the button must have the space number in it.

Hold parking space: Puts the specified parking space on hold. The textbox to the right of the button must have the space number in it.

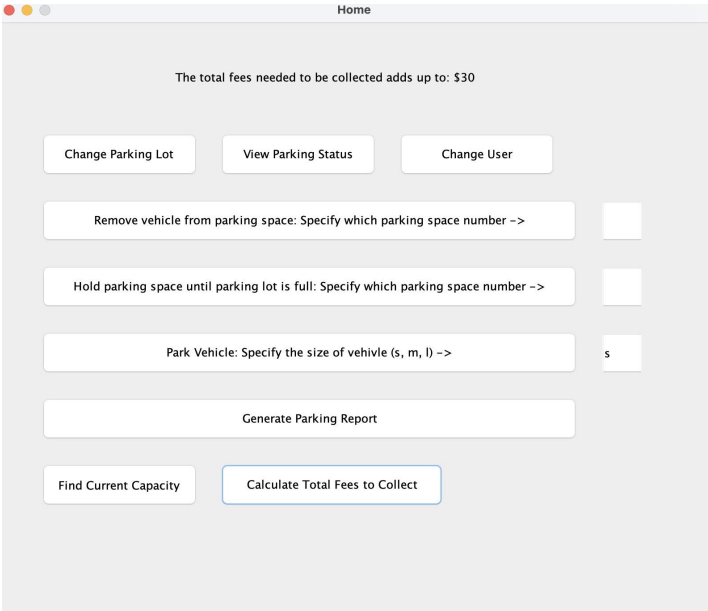
Park vehicle: Parks a vehicle in the next open parking space. The textbox to the right is for the size of the car. Since there are 20 small and 5 large parking spots in each parking lot, a certain amount of cars can be in the parking lot. By default the vehicle is set to small, so the textbox is there to identify if the vehicle is larger than “small”.

Generate Parking Report: Fills the ParkingReport text file with information about all the parking lots. An image of this is provided in the “Description of the Text Files” portion of this report.

Find Current Capacity: Prints the current capacity to the home screen result label.



Calculate Total Fees to Collect: Prints the total fees due to the home screen result label.



Possible errors that can occur:

1. When parking a vehicle the program might crash if the letter isn't s, m or l.
2. If a number that is higher than the capacity of the parking lot is put in the remove vehicle textbox or hold parking space text box, an error might occur.
3. A combination of all the buttons might lead to the program looping and constantly forcing the parking status screen.

References

https://www.javatpoint.com/java-joptionpane

https://7esl.com/list-of-nouns/

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html

https://www.javatpoint.com/java-jpanel