**What each class and function do: (legend at end of the document)**

HotelSystem.java:

- Contains a new HotelMenu object and runs that object.

- This is the part of the system that starts the program.

HotelMenu:

- Constructs a HotelMenu object

- Prints out the user’s available actions to the command line.

- Takes user input using scanners.

**-Public void run():**

- contains 3 types of Boolean attributes; more (while more is true the system will run, if more becomes false the system will stop, this occurs when the user selects the sign out option), isAuth (isAuth determines if the user is allowed to run the full program by comparing an inputted username and password, to a stored username and password in a CSV. If the username and password match,, the user is allowed to continue on with the program, if they don’t they will receive an error message saying they have inputted an incorrect username/password and then they will have the original option returned to them again for a new input), and isAdmin (isAdmin compares the authority of the user in the CSV, after isAuth has confirmed they are allowed to use the system, if the user has admin in the CSV, isAdmin is positive and allows the user to use admin commands, else the user can only use customer commands.

- contains if loops to check the authority of the user (see 3 types of Boolean attributes)

- originally the user cannot be logged in, so is presented with two options:

- login (which outputs text asking for username and password, these inputs are passed through a Login object that determines if the user is a valid user and if the user has admin privileges)

-Quit (which terminates the program)

- the user is then presented with one of two options:

- admin commands which contain:

- remove rooms (removes rooms from a list of available rooms)

- add rooms (adds room to a list of available rooms)

- cancel a booking (cancels any customers booking from the reservations)

- sign out (logs the user out and is isAuth and isAdmin become false)

- customer commands which contain:

- cancel a booking (cancel a booking that this current user has made previously)

- new booking (creates a new booking for this current user)

- sign out (logs the user out and isAuth becomes false)

- once the user is signed out, they are presented with the login/quit options again.

- if the user inputs an invalid command they will receive an error stating they have inputted an invalid command and they will then be prompted to input a new valid command.

**- private void adminOptions(String currentLoggedInUser, String command):**

-if the user has been verified as an authorized user, and they have admin privileges, they’re shown the admin commands.

- once the user has inputted a command, the command string is ran through the adminOptions() method.

-the admin can add or remove rooms from the available rooms list, they are prompted with confirming their alteration. If the alteration is confirmed, the calendar is updated with the relevant data inputted.

- if the user has not been verified as an authorized user, they will be shown the customer commands.

- once the user has inputted a command, the command string is ran through the customerOptions() method.

- the customer can create a new booking and is prompted for a checkin and checkout date, which are then broken down user a parseDates() method, to validate the dates.

- if the dates are valid the customer is then prompted to enter the amount of rooms and the amount of guests that are required.

- the availability of the required options is then checked, if there is availability a new calendar object is created that retrieves the data from the hotels CSV, breaking up the information by “,” into Strings for hotel, roomType, costPerNight, and then the costPerNight is parsed into a Double for cost.

- these values are then inputted into String booking reference that is a calendar.makeBooking() method.

- if the bookingRefernce length is greater than zero, then the booking was successful, and the customer is shown a message declaring the booking a success. Else if the booking was not greater than 0, the booking was unsuccessful, and the customer is shown a message stating so.

- if any incorrect commands or dates are entered throughout the customerOptions() method, the customer is shown an error stating why their input was incorrect.

-private ReservationDate parseDates(String dateToParse):

- this method takes the users inputted date in the format of DD/MM/YYYY and separates the values by “/” using a StringTokenizer.

- the values are parsed individually as integers, and are then passed one by one into a new ReservationDate(day, month, year).

- **private Boolean isDateValid(String dateToCheck):**

- uses a new DateFormat object to check if the date entered by the customer is in the correct format of DD/MM/YYYY.

- if incorrect it throws a parseException, else the dates are parsed in the parseDates() method.

- private Object getChoice(Objecct[] choices):

- runs through the amount of available choices for the chosen command, and prints out a letter A-Z (depending on the amount of choices available) and compares the users inputted letter to the list of choices, and returns the corresponding choice.

Login:

- contains two Strings, username and password, and contains a Boolean isAdmin.

- **public Login(String username, String password):**

- constructor that changes the username and password to the two inputted values (inputted username changes private String username in class, inputted password changes private String password in class)

- **public Boolean getIsAdmin():**

- returns isAdmin,if true the user gains admin commands, if false the user gains customer commands.

**- public String getUsername():**

- returns the username of the current user.

**- public Boolean getAccess():**

- uses a FileReader object to read the usernameAndPasswords CSV and stores the values in a String arraylist called userAndPass

- a Boolean isAuth is created

- the userAndPass arraylist is then searched through, checking if the inputted username and password, from the Login() constructor, matches any paired username and password (paired meaning the two values are not just any random username and password in the arraylist, and that they correspond to one another).

- if the inputted username and password match with a paired username and password, isAuth is returned and the user is then checked for admin access.

- if no match is found the user is prompted with an error stating an incorrect username/password was inputted.

- there is then a FileNotFoundException that outputs an error stating that the filepath for the FileReader was incorrect/ not found.

FileReader:

- contains three String arraylists, info, Reservations, and UserPassword

**-public String [] [] readHotels() throws FileNotFoundException**:

- creates a new java.io.File object that reads the hotels CSV.

- this method contains a Scanner input, a String line, an integer z and initiates the String arraylist info.

- a while loop reads through each line of the CSV, adding the data to the String arraylist info.

- once there are no more lines to be read the while loop ends, the scanner is closed, and info is returned.

- **public String[] [] readRes() throws FileNotFoundExceeption:**

- creates a new java.io.File object that reads the reservations CSV.

- this method contains a Scanner input, a Scanner count, a String line, a String c, and an integer z.

- a while loop counts the amount of lines the file has.

- the Reservations arraylist is then initiated based on the length of the length of the file that was counted in the while loop.

- a second while loop cycles through each line in the CSV and copies the CSV into the Reservations arraylist, separating each value by “,”.

- once the while loop is finished the second scanner is closed and Reservations is returned.

- **public String [] [] readUserPasswordFile() throws FileNotFoundException**:

- creates a new java.io.File that reads the UsernameAndPasswords CSV.

- this method contains a Scanner input, a Scanner count, a String line, a String c, and an integer z.

- a while loop counts the amount of lines the file has.

- the UserPassword arraylist is then initiated based on the length of the length of the file that was counted in the while loop.

- a second while loop cycles through each line in the CSV and copies the CSV into the UserPassword arraylist, separating each value by “,”.

- once the while loop is finished the second scanner is closed and UserPassword is returned.

**ReservationCalendar:**

- contains a private Arraylist<Reservation> reservations, a private Arraylist<Hotels> hotels, and private int booking number.

- **public ReservationCalendar():**

-this is an empty constructor that initiates the Arraylists reservations and hotels, while also calling the methods AddReservations() and AddHotels().

- **private void AddReservations():**

- creates a new FileReader that uses the readRes() and loads the data into a String[] [] reservationCSV.

- a for loop goes through the length of the reservationCSV collecting the data and passing it through a new reservation object. A new booking is created then using the addBooking() method.

- an if loop then determines the value of the bookingNumber for the newly made booking.

- finally there is a FileNotFoundException that prints the stack trace, in case the FileReader cannot locate the file through the file path.

**- private void AddHotels():**

- a FileReader uses the readHotels() method to transfer data into the String[] [] hotelCSV.

- a for loop then loops through this and adds the cost for each different night into a double array called rates. A new room object is created, and the minimum and maximum capacity of the room, as well the rates of the room is declared.

- an if loop is then used to check if the name of the hotel has been declared, if it hasn’t, the previous hotel name is used.

- a new hotel object is then created, and the hotel type and number of rooms is declared, then the new hotel object is added to the hotels arraylist.

- finally there is a FileNotFoundException that prints the stack trace, in case the FileReader cannot locate the file through the file path.

- **public void addBooking(Reservation r, Boolean initialLoad):**

- adds a new reservation to the reservations arraylist.

- if the variable initialLoad is false the method UpdateReservationFile() is declared.

- **public void cancelBooking(Reservation r):**

- removes the reservation r from the reservations arraylist.

- the method UpdateReservationFile() is then declared.

- public String makeBooking(String name, ReservationDate checkIn, Reservation checkout, int numberOfRooms, int numberOfPeople, String hotel, String roomType, double cost, String reservationType):

- creates a String currentReservation.

- increases the bookingNumber value by one, and creates a new Reservation object res, with the variables bookingNumber, name, roomType, checkIn, checkout, numberOfRooms, numberOfPeople, cost, reservationType.

- a new booking is created using addBooking using the Reservation object res that was just created.

- at the end of the method there is an Exception, that decreases the bookingNumber in case the reservation failed.

- finally the currentReservation String is returned.

- **public ArrayList<String> checkAvailability(ReservationDate checkIn, ReservationDate checkout, int numberOfRooms, int numberOfPeople)**:

- creates a new ArrayList<String> object called res.

- a for loop goes through the hotels arraylist and ensures the amount of guests staying is under the max occupancy for the rooms, and that the amount of guests is greater than the minimum occupancy.

- it then checks if the number of rooms requested is less than the total amount of rooms in the hotels.

- then it is calculated if the booking will take place in seven or more days, if it is, then the type of payment will be advance pay, if it is less than seven days it will be standard pay.

- there is a ParseException that prints the stack trace if failure occurs.

- the cost of the stay is then calculated by calling the GetRates() function that gets the rates for the room in the hotel, adds up the total cost for the length of the stay, multiplies it by the number of rooms booked. The final cost is stored in a double called rate.

- if the purchase is advance purchase, the double rate is multiplied by 0.95 to get a 5% discount for advance purchase, if the purchase is standard purchase, it is not changed.

- the reservations arraylist then has the new reservation added to it.

- finally the arraylist res is returned.

- **public ArrayList<Reservation> getReservations(String username, Boolean isAdmin):**

- a new arraylist<reservation> object called res is created.

- a for loop goes through the reservations arraylist, and if the user has admin privileges or the user has the same username as the user who created the booking, the reservation can be added into the res arraylist.

- res is returned in the end.

- **public Reservation getReservation(Integer reservationID):**

- a for loop goes through the reservations arraylist

- if the reservation number is equal to the reservationID that is inputted then the reservation at that point is returned. Else null is returned.

- **private ReservationDate parseDates(String dateToParse):**

- a new StringTokenizer object is created that will separate the inputted dateToParse by “/”.

- the digits before the first “/” are entered into the int date field.

- the digits between the two “/” are entered into the int month field.

- the digits after the last “/” are entered into the int year field.

- those three ints are then passed into a new ReservationDate object.

**- private Double GetRates(Hotels h, ReservationDate checkIn, ReservationDate checkout):**

- a double rate and DateFormat object are created.

- the DateFormat object formats the Reservation into a pattern of dd/mm/yyyy

- the difference between check in and the checkout date is determined in milliseconds and then converted into days.

- a for loop cycles through for each day, and determines the cost per day, adding each day onto the final total.

- a ParseException prints out the stack trace in case of a failure.

- finally the rate is returned.

- **private Integer CheckNumberOfRoomsAvailable(Hotels h, Reservations checkIn,ReservationDate checkout):**

- integer is declared that gets the max amount of rooms available.

- a for loop then cycles through the reservations arraylist and formats checkin and checkout dates into the format dd/mm/yyyy.

- its then checked if the date ranges overlap and then the count of the available rooms is reduced.

- a ParseException prints out the stack trace if a failure occurs.

- finally the maxRoomsAvailable is returned.

- **private void UpdateReservationFile():**

- a buffered writer updates the reservations.csv file

- a for loop goes through the length of the reservations arraylist and the buffered reader goes through and writes the name, reservation type, checkin date, checkout date, number of rooms, room type, number of people staying and the total cost of the stay, into the reservations csv.

- once the for loop is finished the buffered writer is closed

- an IOException prints out the stack trace if an error occurs.

Hotels:

- contains a private String name, private integer rooms\_available, private Room roomType.

- public Hotels():

- empty constructor.

- public Hotels(String hotel, Integer rooms\_available, Room roomType):

- constructor that alters the private data fields in the class with the inputted data fields.

- public void setRooms\_available(Integer rooms\_available):

- sets what rooms are available to book

- public Integer getRooms\_available():

- returns what rooms are available to book

- public Room getRoomType():

- returns the type of the room

- public String getName():

- returns the name of the hotel.

- public String format(Double rate, Boolean isAdvancePurchase):

- method formats the output of the data about the hotels and returns the hotel name, room type, rate of the room and the purchase type.

- public String toString():

- method to format the output of the data about the hotels, returns the name of the hotel and the type of room.

Room:

- contains private String roomType, private int minOccupancy, private int maxOccupancy, private double[] rates, private ArrayList<Room> function, private ArrayList<Reservation> booked.

- **public Room(String roomType, int minOccupancy, int maxOccupancy, double[] rates):**

- constructor that alters the private data fields using the inputted data.

- **public void addRoom(Room add):**

- method to add rooms.

- **public void removeRoom(Room add):**

- method to remove rooms.

- **public void setMaxOccupancy(int maxOccupancy):**

- sets the max occupancy for a room.

- **public void setMinOccupancy(int minOccupancy):**

- sets the min occupancy for a room.

- **public void setRoomType(String roomType):**

- set the room type.

- **public void setRates(double[] rates):**

- sets the rates for a room.

- **public String getRoomType():**

- returns the room type.

- **public int getMinOccupancy():**

- returns min occupancy.

- **public int getMaxOccupancy():**

- returns max occupancy.

- **public double[] getRates():**

- returns the rates for a room.

- **public String format(Double rate, Boolean isAdvancePurchase):**

- determines if the purchase type is standard or advanced and then returns the room type, with the rate and the rate type.

- **public String toString():**

- returns room type

Reservation:

- contains private int resNumber, private String name, privateString reservationType, private ReservationDate checkin, private ReservationDate checkout, private int numberOfRooms, private String roomType, private int numberOfPeople, private double totalCost

- **public Reservation(int resNumber, String name, String roomType, ReservationDate checkin, ReservationDate checkout, int numberOfRooms, int numberOfPeople, double totalCost,, String reservationType):**

- constructor that alters the private data fields in the class using the inputted data fields.

**- public int getResNumber():**

- method that sets the reservation number.

**- public String getReservationType():**

- method that returns the type of reservation.

**- public void setReservationType(String reservationType):**

- method that sets the reservation type.

**- public String getName():**

- returns the name of the customer who made the booking

**- public void setName(String name):**

- method that sets the name of the customer who created the reservation.

**- public ReservationDate getCheckin():**

- method that returns the checkin date.

**- public void setCheckin(ReservationDate checkin):**

- method that sets the checkin date.

**- public ReservationDate getCheckout():**

- method that gets the checkout date.

**- public void setCheckout(ReservationDate checkout):**

- method that sets the checkout date

**- public int getNumberOfRooms():**

**­**- method that returns the number of Rooms

**- public void setNumberOfRooms(int numberOfRooms):**

- method that sets the number of rooms.

**- public String getRoomType():**

- method that returns the room type.

**- public int getNumberOfPeople():**

- method that returns the number of people.

**- public void setNumberOfPeople(int numberOfPeople):**

- method that sets the number of people who are staying.

**- public double getTotalCost():**

- method that returns the total cost of the stay.

**­- public void setTotalCost(double totalCost):**

- method that sets the total cost of the stay.

**- public String toString():**

- method that formats the output of the required data, it returns string containing reservation number, checkin date, and checkout date

ReservationDate:

- contains private int year, private int month, private int day.

**- public ReservationDate(int day, int month, int year):**

- constructor that alters the private date fields in the class with the inputted data fields.

**- public int getDay():**

- method that returns the day.

**- public void setDay(int day):**

- method that sets the day.

**- public int getMonth():**

- method that returns the month.

**- public void setMonth(int month):**

- method that sets the month.

**- public int getYear():**

- method to get the year.

**- public void setYear(int year):**

- method to set the year.

**- public String format():**

- method that formats the output of the data fields in the class

**Legend:**

**“-“ signifies a new point.**

**A sentence in bold is aa method..**

**A word that’s underlined is the class name.**