Project document

Reservation system for hotel

Jack Bergkulla 705855

Automaatioteknologia

5.5.2022

1. **General description**

   The program is a reservation system for a hotel. The difficulty level of the project is easy. In the reservation system the customers are able to book rooms for a specific date. The user is able to see what customer has stayed at the hotel and in which room at what date. The user is able to save and load a file containing records of previously saved data.

2. **Instructions for the user**

   When the program is started the user is shown different options of what can be done. Available options are Booking, Customers, Record, Load File, Save File and Exit. These are furthermore explained in the readme file and below.

   By typing "1" the user is redirected to reservation class. Then the user is asked for input for one's name. If it's a new customer one is asked to input phone number, if the customer is already a registered customer there is no need for phone number as it is already in the system. The customer fetch/creation happens through customer class where a customer is either fetched from the system or created. The customer class creates an object that has "name" and "phone number" properties. All customers are stored in the "customer_list". After the customer is fetched, the user is asked to input date of the stay at the hotel. Then the customer object and reservation_date is returned to home class where it is fetched to "Schedule" class. There the booking is reserved through "insert_schedule" function into the hotel schedule, if there is available rooms that date.The function returns a value to "Home" class depending on this, where the customer is informed whether the booking was successful and what room was booked.

   By typing "2" the user gets a print of what customers is/has been at the hotel, the customer name and phone number is shown. This happens through "Reservation" and "Customer" classes.

   By typing "3" the user is shown a record of all bookings on the hotel, both past and future. The record shows "Booked date" and "Customer". This happens through "print_schedule" function in class "Schedule".

   By typing "4" the user can load a textfile containing a record of customers and bookings. After loading the file the user can see everything that was booked

and all customers on the file. The loading happens through the Class "File" and function "load_file" and "final_load".

By typing "5" the user can save a textfile containing all records, bookings and customers, that have been made. The savinf happens through the Class "File" and function "save_file".

The saving and loading is implemented by writing and reading a textfile that contains the information from all 6 lists in "Rooms", customer_list, customer_name and customer_phonenumber. The loading of said textfile was challenging to implement. The saving of a file uses the classes "Reservation" and "File" while loading the file uses all 5 classes.

By typing "0" the program is exited and the customer is shown "Thank you for choosing HOTEL KAARL, welcome back!"

3. **External libraries**

   Only external library that is used in the program is the “time” library, this to make the program give a smoother experience to the user by implementing small delays between menus.

4. **Structure of the program**

   Class Home: contains the main function without any other functions. Used to present the menu to the user and the different menu options there is.

   1. Booking – goes into class Reservation class
   2. Customers – goes into Customer class through Reservation class. This had to be done due to not wanting to “loop” import classes.
   3. Record – goes into Schedule class
   4. Load – Uses mainly File class but implements every one of the 5 classes available
   5. Save – uses mainly File class but also Reservation class
   0. Exit – exits program in Home class

Home class

Imports Reservation, Schedule and File classes

Creates menu and overall runs program as sort of a “super”.

Reservation-class

Imports Customer class

Functions:

Booking

Used to create/fetch customer through customer class.

Make_customer_lists

Used when saving file to provide customer names and phone numbers.

Create_customer_from_lists

Used to create customers by customer names and phone numbers when loading a file

Print_customers

A redirect function to Customer class when printing customer records.

Customer class

__init__

Used to create customer, needs to get name and phone number.

fedge_customer

Used to check if customer already exits in system, if exists then returns customer. If customer do not exist it creates a new customer.

Get_customer_info

Returns customer name and phone number to make fewer lines in program

Print_customers

Function that gets redirect from Reservation print_customers, prints info of all customers.

Schedule class

Imports customer class

__init__

Used to create schedule of hotel containing 6 lists. Either used when starting new program file or loading an old textfile containing data.

Insert_schedule

Inserts booking into schedule

Print_schedule

Prints what rooms have been booked in past and future and what date and what customer booked it.

File class

Imports Schedule class

Save_file

Saves all data that has been stored by the user in a textfile. All data is stored on different rows split by a \n. Lists are separated by a string to get right data into right lists.

Load_file

Used to load data that has been stored in a textfile. All rows are read separately and due to lists being separated by specific strings the reading is smooth. First function of the loading process

Final_load

Second function of the loading process. When all data have been read in the load_file function this function creates the new lists that is to be used in the system when loading is completed. It seemed natural to me to split the loading of data process into two functions, more logical and if errors are identified the error searching can be much faster.

5. **Algorithms**

   I think my best algorithm is in the File class used to save and load system data. When creating the saved textfile all lists are consequently read into the file. Every line consists of one datapoint and lists are furthermore separated by a string. This makes the saved file readable and the loading process more smooth. The loading of a file is a bit more complicated but works in a similar way. I am particularly happy I got to in the final_load function implement a for-in-range algorithm inside another for-in-range algorithm. They always demand a bit of thinking but run very smoothly when implemented. The double for-in-range algorithm consisted of first looking through the Schedule-object in the saved file data and when finding the right reservation then adding the new created customer-object to the newly created Schedule-object.

6. **Data structures**

   Types and data structures used in the program are mainly class objects and lists. Strings are also much used, not many ints at all.

   Lists were overall easily the most suitable for this program. A data structure in class Schedule was created, it is a class object that consists of 6 lists. The class object was the main part of the data stored and was very useful.

7. **Files**

   Text files are handled when saving and loading data.

   The data is stored on separate lines, one line holds one datapoint. Furthermore different lists data is separated by a specific string.

8. **Testing**

   Some basic testing was done, like creating customers and bookings. Also tests of loading files was created. Result of testing was to put in some more fail safeties.

9. **The known shortcomings and flaws in the program**

   One thing is that customers are stored 2 times in the textfile, this to make a distinction between which customers are booked in which rooms. This could be reworked to make them only be stored once, although it would demand a major change of the code. The separating lines between lists in the textfile could also be improved, this would make the code cleaner, also would take away more errors happening, although errors are not happening when implementing the program on a not so big scale.

   Another shortcoming is that when printing the customer list the program at the moment has to go through the Reservation class and not directly to Customer class, this could cause troubles if one would continue to work on the project.1

10. **3 best and 3 worst areas**

    Best

    Saving and loading implementation is good.

    It is a user friendly program.

    All information that is stored and used in the program is available for the user to print out.

    Worst

    Not implementing the print_customer function directly from Home to Customer.

    Too many functions that only have one usage, could probably be compress 2 functions into 1.

    Load_file function is very long, could probably be implemented in half the rows if an algorithm would be made.

11. **Changes to the original plan**

    I added the class “File” for saving and loading data, I had not thought about this part enough to figure out it needed an own function. I also removed the class “Company” that originally was in the project plan, this because it was easily merged with another class.

    I pretty much stuck with my original planning, though much of the work happened 2 weeks before due to heavy work load in other courses/work.

12. **Realized order and scheduled**

    The Home class was first implemented. Secondly I implemented the Cstomer class because it was a pretty basic class. Reservation file then to be able to make a date and connect it with a customer and then Schedule class to piece together all the bookings. Lastly the File class was implemented, this was important because then all data types and structures were completed and the implementing of a good textfile could be done.

13. **Assessment of the final result**

    I am happy with my project; it is a functional (though simple) system for hotel reservation management. I worked the planned amount of hours on the project and when I had understood and built up the structure the project took a leap. I would assess for the easy level of this project it is a 4/5. Enhancements could be to minimize the amount of code and implement more algorithms instead. More functions could be added as different lengths of stay, more customer info and things as payment and room service. The program structure is good in some parts of it to making changes and expanding, as saving/loading file and the Schedule class could easily get expansions. Although some parts would need reworking to make it better for expanding.

14. **References**

    What books, web pages, or other material have you used? All sources must be reported, even if they included only the course textbook and API description of the base class libraries.

    A+, https://plus.cs.aalto.fi/

    Stack Overflow, https://stackoverflow.com/

15. **Attachments**

    Attachment of trial of program attached in email.