COSC1295 Advanced Programming
Computer Science and Information Technology
School of Science, RMIT

Assignment 1 - Semester 2, 2019

Submission due date: end of week 5, 11:59 PM on Sunday 25th of August, 2019.

Introduction

This assignment is worth 15% towards your final grade.

NOTE

- This assignment is of a size and scope that can be done as an individual assignment. Group work is not allowed.
- A more detailed marking rubric will be provided closer to submission.

You are required to implement a basic Java program using Java Standard Edition 8.0 or higher. This assignment is designed to:

- Practise your knowledge of design classes in Java
- Practise the implementation of various kinds of classes in Java
- Practise the use of polymorphism

Academic Integrity

The submitted assignment must be your own work. For more information, please visit http://www.rmit.edu.au/academicintegrity.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

Overview

NOTE: Carefully read this document. In addition, regularly follow the Canvas assignment discussion board for assignment related clarifications and discussion.

For this assignment you need to write a console application in the Java programming language which allows a hotel named CityLodge to manage the renting and maintenance of two types of hotel rooms: standard hotel rooms and suites.

Common Features of All Hotel Rooms

Each hotel room managed by CityLodge has the following attributes:

- Room id: a string which uniquely identifies each room. A room id must begin with R_ if the room is a standard hotel room and S_ if the room is a suite. For example R_108 or S_559.
- Number of beds: a standard room can have either 1, 2, or 4 beds. A suite always has 6 beds.
- Feature summary: a short string (maximum 20 words) to summarise all the room features. For example "air conditioning, cable TV, Wifi, fridge, electric kettle".
- Room type: CityLodge currently has two types of rental rooms: standard room and suite. For simplicity, we use two values for room type: i.e. Standard and Suite. The different characteristics of those types of rooms are described in the sections below.
- Room status: employees of CityLodge inspect this attribute to determine whether the room is currently available for rent or being rented or under maintenance. For simplicity, we use three values for room status: i.e. Available, Rented and Maintenance.

Furthermore, each room also keeps its own collection of Hiring Records, i.e information about the 10 most recent times that room has been rented. Details of a hiring record is shown below:

Hiring Record

Each Hiring Record has the following attributes:

- Record id: a string which uniquely identifies each hiring record. A hiring record id is constructed by concatenating the following three attributes

  roomId_ + customerId_ + rentDate (8 digit format: ddmmyyyy)

  Note: for simplicity, each customer is simply represented by a unique string of customer id of your choice. There is no need to implement a class to store customer information.

- Rent date: the date when a customer rents the room

  Source code of the **DateTime.java** is provided. Please click here to access the code. Some examples of how to use the DateTime.java class is here.

- Estimated return date: the calculated date given the number of days a customer wants to rent the room and the rent date shown above

  Example: a customer wants to rent a room on 16/07/2019 for 3 days, hence the estimated return date will be 19/07/2019

- Actual return date: the date when the customer actually returns the room
- Rental fee: the fee calculated based on the type of room, the rent date and the estimated return date.

- Late fee: the additional fee which must be calculated when the actual return date is after the estimated return date.

    Note: standard room and suite have different formulae to calculate rental fee and late fee, which will be shown in the next sections

Standard Room

As mentioned above, CityLodge has two types of rooms for rent. The first type is standard room, which has the following characteristics:

- A standard room can have either 1, 2, or 4 beds.

- Each standard room can be rented for:
    - a minimum of 2 days if the rental day is between Monday to Friday
    - or a minimum of 3 days if the rental day is Saturday or Sunday
    - and a maximum of 10 days

- A standard room has the following rental rates:
    - $59 per day if the room has 1 bed
    - $99 per day if the room has 2 beds
    - $199 per day if the rooms has 4 beds

- If a standard room is returned earlier than the estimated return date, there is no additional fee applied and the rental fee is calculated based on the rent date and the date the standard room is returned (actual return date)

- Late fee: if a standard room is returned later than the estimated return date, then the rental rate of each late day is 135% of the normal daily rate for that standard room type. For example, a standard room with 2 beds has a daily rate of $99 as shown above. Therefore the rental rate of each late day is 135% of 99 = 135/100 * 99 = $133.65

- A standard room has no fixed maintenance schedule. CityLodge can perform maintenance on a standard room at any time that there is no customer renting the standard room, i.e. when the standard room is available.

Suite

The second type of room CityLodge offers for rent is suite. A suite has the following characteristics:

- A suite always has 6 beds

- The rental rate of a suite is $999 per day

- Late fee: if a suite is returned after the estimated return date, then the late fee is calculated at $1099 per day

- Each suite has a strict maintenance schedule because CityLodge wants all their suites to be in the best possible conditions. Therefore they specify the following requirements:

    - All suites must have a maintenance interval of 10 days.

- Each suite must keep its last maintenance date. Maintenance operations for a suite must be done no more than 10 days (as specified by the maintenance interval above) after its last maintenance date.

- Customers will not be allowed to rent a suite for a time period which exceeds the date on which maintenance operation must be done.

  Example: a suite is available and last underwent maintenance on 15/07/2019. Maintenance must be done for that suite no later than 25/07/2019. Therefore, if a customer wants to rent that suite on 21/07/2019 for 5 days, CityLodge system will reject that request.

Implementation Requirements

Your Hiring Record class must meet the following requirements:

- Override the **public String toString()** method to return a string containing the details of a hiring record in the following format:

  `recordId:rentDate:estimatedReturnDate:actualReturnDate:rentalFee:lateFee`
  (notice how the colon is used as a separator)

  Example 1: When a standard room is being rented, but hasn't yet been returned, calling the toString method of the latest Hiring Record object of that room will return a String as shown below:

  `R_108_CUS239_11072019:11/07/2019:16/07/2019:none:none:none`
  (notice how ***none*** strings are used for actualReturnDate, rentalFee and lateFee because these values cannot be determined when the room hasn't been returned)

  Example 2: When a room has been returned, the latest hiring record of that room will be updated with the actualReturnDate, the rentalFee and any lateFee. Therefore, calling the toString method of the latest Hiring Record object of that room will return a String as shown below:

  `R_108_CUS239_11072019:11/07/2019:16/07/2019:16/07/2019:495.00:0.00`

- Implement a **public String getDetails()** method. This method should build a string and return that string. The returned string should be formatted in a human readable form as shown below. This method MUST NOT do the actual printing to the console. Please refer to the following examples:

  Example 1: When a room is being rented, but hasn't yet been returned, calling the getDetails method of the latest Hiring Record object of that room will return a String as shown below:

  ```
  Record ID:             R_108_CUS239_11072019
  Rent Date:             11/07/2019
  Estimated Return Date: 16/07/2019
  ```
  (the actualReturnDate, rentalFee and lateFee have no value yet and therefore are not included in the returned string)

  Example 2: When a room has been returned, the latest hiring record of that room will be updated with the actualReturnDate, the rentalFee and any lateFee. Therefore, calling the getDetails method of the latest hiring record object of that room will return a String as shown below:

```
Record ID:              R_108_CUS239_11072019
Rent Date:              11/07/2019
Estimated Return Date: 16/07/2019
Actual Return Date:     16/07/2019
Rental Fee:             495.00
Late Fee:               0.00
```
(notice that rentalFee and lateFee are printed with 2 decimal places)

Implementation requirements for all room classes (standard room and suite)

Each room must maintain its own collection of hiring records. These records store information about the 10 most recent times that room has been rented.

The following methods can be called on any object of type standard room or suite:

(Hint: implementing these methods is a good chance for you to apply inheritance and polymorphism in your code)

public boolean rent(String customerId, DateTime rentDate, int numOfRentDay)

This method is called on a room object (a standard room or a suite) to perform the operations required to rent the room.

This method should check for pre-conditions to determine if the room can be rented. For example, this method returns false when the room is currently being rented or is under maintenance. You should check any other possible conditions which would make this method return false.

If the room is available for rent, this method will perform all necessary operations to update the information stored in the room object based on the input parameters. For example, updating the room status, creating a new hiring record, updating the hiring record collection of the room, and any other operations you consider necessary.

This method returns true to indicate that the room has been rented successfully.

public boolean returnRoom(DateTime returnDate)

This method is called on a room object (a standard room or a suite) to perform the operations required to return the room.

This method should check for pre-conditions to determine if the room can be returned. For example, this method returns false when the given returnDate is prior to the rentDate stored in the hiring record. You should check any other possible conditions which would make this method return false.

If the room can be returned, this method will perform all necessary operations to update the information stored in the room object based on the input parameter. For example, updating the room status, updating the corresponding hiring record with the rental fee, the late fee, and any other operations you consider necessary.

This method returns true to indicate that the room has been returned successfully.

public boolean performMaintenance()

This method is called on a room object (a standard room or a suite) to execute the operations required to perform the maintenance of the room.

This method should check for pre-conditions to determine if maintenance operations can be performed in that room. For example, this method returns false when the room is currently being rented because maintenance operations cannot be performed when there is a customer renting the room. You should check any other possible conditions which would make this method return false.

If the room is ready for maintenance, this method performs all necessary operations to update the information stored in this room object when a maintenance happens. This method returns true to indicate that the room is under maintenance.

public boolean completeMaintenance(DateTime completionDate)

This method is called on a room object (a standard room or a suite) to perform the operations required when the maintenance of the room is finished.

This method should check for pre-conditions. For example, when the room is currently being rented, it does not make sense to call completeMaintenance method on this room object, and therefore this method should return false. You should check any other possible conditions which would make this method return false.

If it is possible to complete maintenance, this method will perform all necessary operations to update the information stored in this room object when the maintenance has finished. Finally, this method returns true to indicate that the room maintenance has finished successfully.

public String toString()

This method should build a string and return it to the calling method. The returned string should be formatted in a pre-defined format as shown below:

```
roomId:numberOfBeds:roomtype:status:featureSummary
```

If the object is a suite, the returned string should include the last maintenance date in the format DD/MM/YYYY. The format of the string representation of a suite is shown below:

```
roomId:numberOfBeds:roomtype:status:lastMaintenanceDate:featureSummary
```

Please notice the following:

- The colon (:) is used as a separator.
- If that room is a suite, the attribute lastMaintenanceDate is included as shown above.
- You can create your own feature summary string with maximum 20 words limit as shown in the Common Features of All Hotel Rooms section above.

More examples are shown below:

Example 1: The CityLodge hotel has a standard room with id R_108, which has 2 beds, and is now ready for rent. The room has facilities such as air conditioning, cable TV, Wifi, fridge, electric kettle. Calling toString method of that room object returns the following string:

```
R_108:2:Standard:Available:air conditioning, cable TV, Wifi,
fridge, electric kettle
```

Example 2: A suite with id S_559 is currently being rented. It has 6 beds with outstanding features such as a large seating area with a flat screen TV together with outdoor balconies. The suite has its last maintenance date 22/07/2019. Calling toString method of the suite object returns the following string:

```
S_559:6:Suite:Rented:22/07/2019:large seating area, flat screen
TV, outdoor balconies
```

public String getDetails()

This method should build a string and return it to the calling method. This method SHOULD NOT do the actual printing. The returned string contains all information about the room, including details about up to 10 most recent hiring records of that room. The returned string should be formatted in a pre-defined human readable format. See the examples below:

Example 1: a standard room is available for rent for the first time (no hiring record). Calling getDetails method of that room object returns the following:

```
Room ID:              R_111
Number of beds:  2
Type:                 Standard
Status:               Available
Feature summary:      air conditioning, cable TV, Wifi, fridge
RENTAL RECORD:        empty
```

(no need to keep track of maintenance date of a standard room)

Example 2: a suite is currently being rented for the first time. Calling getDetails method of that suite object returns the following:

```
Room ID:              S_599
Number of beds:       6
Type:                 Suite
Status:               Rented
Last maintenance date: 22/07/2019
Feature summary:      large seating area, outdoor balconies
RENTAL RECORD
Record ID:            S_599_CUS001_24072019
Rent Date:            24/07/2019
Estimated Return Date: 27/07/2019
```

(notice that the last maintenance date is shown because this is a suite and part of the first hiring record are shown because the suite is being rented for the first time)

Example 3: a suite is currently being rented. It was rented and returned one time in the past.

```
Room ID:                S_773
Number of beds:         6
Type:                   Suite
Status:                 Rented
Last maintenance date:  21/07/2019
Feature summary:        large seating area, outdoor balconies
RENTAL RECORD
Record ID:              S_773_CUS003_25072019
Rent Date:              25/07/2019
Estimated Return Date:  28/07/2019
-------------------------------------
Record ID:              S_773_CUS001_13072019
Rent Date:              13/07/2019
Estimated Return Date:  16/07/2019
Actual Return Date:     17/07/2019
Rental Fee:             2997.00
Late Fee:               1099.00
```

(notice how the latest hiring record is shown first, although it's still not complete until the room is returned. The first rental record is shown below the current rental record, separated by the line ----------)

Implementing the CityLodge application class

You are required to implement a class named CityLodge which contains a single collection to store up to 50 objects of both types standard room and suite. Objects of type standard room and suite will be added at runtime by the user of your program via Console input using a menu system described below.

Your CityLodge class must present the following menu for employees of CityLodge company to manage rental rooms. The menu is shown below:

```
**** CITYLODGE MAIN MENU ****

Add Room:                   1
Rent Room:                  2
Return Room:                3
Room Maintenance:           4
Complete Maintenance:       5
Display All Rooms:          6
Exit Program:               7
Enter your choice:
```

User should be able to enter a number from the menu above to select an option. If the input is outside of that range, an error message should be displayed and the menu should be re-displayed. When a valid number is entered, your program should execute the corresponding method and then return to the menu. Your program should exit if the user selects the Exit Program option.

You may use a sub menu under an option.

All output data should be printed to the standard output.

Following is a description of each feature provided by the menu above:

Add room

The user (an employee of CityLodge) selects this option to create a new standard room or a suite. The user can enter all relevant details of a new room, such as room id, room type, number of bed rooms and last maintenance date if the room is a suite. When a room is first created, it's status is Available.

You should perform all necessary data validation, for example, invalid room id or room id already exists. If there is an error, your program should print an appropriate error message to the console and should go back to the menu immediately without creating or storing a new room.

Rent room

By selecting this option, an employee of CityLodge can then enter a standard room ID or suite ID and enter relevant information for a customer to rent that room. Your implementation should be similar to the following example:

Example 1: rent a standard room which is available for rent

```
Enter your choice: 2
Room id: R_108
Customer id: CUS239
Rent date(dd/mm/yyyy): 19/07/2019
How many days?: 3
Room R_108 is now rented by customer CUS239
****** CITYLODGE MAIN MENU ******
Add Room:           1
Rent Room:          2
Return Room:        3
..............
(main menu is shown again)
```

Example 2: attempt to rent a room which is not available for rent

```
Enter your choice: 2
Room id: R_108
Room R_108 is not available at the moment.
****** CITYLODGE MAIN MENU ******
Add Room:           1
Rent Room:          2
```

```
Return Room:        3
.............
```
(main menu is shown again, without asking user further details)

Return room

By selecting this option, an employee of CityLodge can then enter a room or suite ID and a return date to return that room. If a return is successful, your program should print all relevant information about that room, including information about the latest hiring record. You should perform all necessary data validation, for example, if that room is currently under maintenance, it's not reasonable to return that room.

Room Maintenance

By selecting this option, an employee of CityLodge can then enter a room or suite ID to put that room under maintenance. You should perform all necessary data validation to avoid any unreasonable scenario.

Example: perform maintenance of a suite which is not currently being rented

```
Enter your choice: 4
Room id: S_599
Suite S_599 is now under maintenance.
****** CITYLODGE MAIN MENU ******
Add Room:           1
Rent Room:          2
Return Room:        3
.............
```
(main menu is shown again)

Complete Maintenance

By selecting this option, an employee of CityLodge can then enter a room or suite ID to complete maintenance of that room. From the ID, if the room is a suite, then the system will prompt the employee to enter a maintenance completion date. If the room is a standard room, no need to enter the maintenance completion date.

Example: complete maintenance of a suite

```
Enter your choice: 5
Enter room id: S_599
Maintenance completion date (dd/mm/yyyy): 20/07/2019
Suite S_599 has all maintenance operations completed
and is now ready for rent.
****** CITYLODGE MAIN MENU ******
Add Room:           1
```

```
Rent Room:          2
Return Room:        3
.............
```
(main menu is shown again)

Display All rooms

By selecting this option, an employee of CityLodge can see on the Console all information about all rental rooms stored in the system, including details about up to 10 most recent hiring records of each room. (Hint: calling the getDetails method on each room object).

Start-up Class.

You should create a startup class which contains a main method in which an object of the CityLodge class is created and a single method is called on that object to run the entire CityLodge application.

General Implementation Requirements

- You are required to modularise classes properly. No method should be longer than 50 lines.
- You should aim to provide high cohesion and low coupling.
- You should aim for maximum encapsulation and information hiding.
- Your coding style should be consistent with Java coding conventions (http://www.oracle.com/technetwork/java/codeconventions-150003.pdf)
- You should comment important sections of your code remembering that clear and readily comprehensible code is preferable to a comment.
- Your programs will be marked with Java SE 8.0. Make sure you test your programs with this setting before you make the submission.

Design Requirements

You must work out an object design for the above task. You should take advantage of object-oriented concepts such as composition, inheritance, method overriding, abstract classes, interfaces wherever appropriate.

Class hierarchies, relationships and components must be conceptualised in a relevant manner, based on the problem description and special conditions listed in this document. Furthermore, you must be able to explain how your program design will perform in certain scenarios and circumstances.

It may be necessary for your design to provide more functionality, such as accessors and mutators, than is specified in the above sections here for the mechanics of your design to work.

Submission Details

You must submit a zip file of your project via Canvas. In Eclipse, right click on your project, select Export, select General, then select Archive File to export your project as a zip file.

IMPORTANT

- Before submisison, check the zip file you submit carefully to make sure that it contains all your code, i.e. all .java files.
- After you have submitted the zip file on Canvas, download your submission from Canvas again and unzip, check if all your source code, all .java files exist.

You can submit your assignment as many times as you would like prior to the due date. The latest submission will be graded.

THE END