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

Assignment 1 - Semester 2, 2018

Submission due date: by 11:59 PM on Sunday 19th of August (end of Week 5)

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 company called FlexiRent to manage the renting and maintenance of various types of rental apartments in

Melbourne CBD. Unlike traditional hotels, FlexiRent offers stylish 1, 2, and 3-bedroom Apartments and Premium Suites for short-term stays in Melbourne.

Rental Property

Each rental property managed by FlexiRent has the following attributes:

- Property id: a string which uniquely identifies each rental property, the id should start with A_ if the property is an apartment and S_ if the property is a Premium Suite

  Example of an Apartment ID: A_700BSMEL for 700 Bourke Street, Melbourne
  Example of a Premium Suite ID: S_633WMSB for 633 Whiteman Street, Southbank

  Note: You are free to use your own format, as long as each rental property is uniquely identified and an Apartment id starts with A_ and a Premium Suite id starts with S_

- Street number
- Street name
- Suburb
- Number of bedrooms of the property
- Property type: FlexiRent currently has two types of rental properties; Apartment and Premium Suite, whose different details are shown further down
- Property status: employees of FlexiRent will inspect this attribute to determine whether the property is currently available for rent or being rented or under maintenance

Furthermore, each rental property also keeps its own collection of Rental Records. These store information about the 10 most recent times that property has been rented.

Rental Record

Each Rental Record has the following attributes:

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

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

  Example 1: Customer with id CUS0011 rented an apartment A_668BSMEL on 14/07/2018, then the rental record id will be: A_668BSMEL_CUS0011_14072018

  Example 2: Customer with id CUS0039 rented a premium suite S_63WMSB on 12/07/2018, then the rental record id will be: S_63WMSB_CUS0039_12072018

  Note: In Assignment 1, 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 property

Source code of the DateTime.java is provided to you. Please click here to access the code.

- Estimated return date: the calculated date given the number of days a customer wants to rent the property (provided when a customer wants to rent that property) and the rent date shown above

  Example: a customer wants to rent a property on 14/07/2018 for 3 days, hence the estimated return date will be 17/07/2018

- Actual return date: the date when the customer actually returns the property

- Rental fee: the fee calculated based on the type of property, 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: Apartment and Premium Suite have different formulae to calculate rental fee and late fee, which will be shown further down

Apartment

As mentioned above, FlexiRent has two types of properties for short-term rental. The first type is Apartment, which has the following characteristics:

- Each Apartment can have 1, 2 or 3 bedrooms

- Each Apartment can be rented for:
  - a minimum of 2 days if the rental day is between Sunday and Thursday inclusively
  - or a minimum of 3 days if the rental day is Friday or Saturday
  - and a maximum of 28 days

- Apartment type has the following rental rates:
  - $143 per day for a 1-bedroom apartment
  - $210 per day for a 2-bedroom apartment
  - $319 per day for a 3-bedroom apartment

- If an Apartment 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 Apartment is returned (actual return date)

- About late fee:

  - If an Apartment is returned later than the estimated return date, then the rental rate of each late day is 115% of the normal daily rate for that Apartment type. For example, a 1-bedroom Apartment has a daily rate of $143 as shown above. Therefore the rental rate of each late day is 115% of 143 = 115/100 * 143 = $164.45

- An Apartment has no fixed maintenance schedule. FlexiRent can perform maintenance on an Apartment at any time that there is no customer renting the Apartment

Premium Suite

The second type of rental property FlexiRent offers for short-term rent is called Premium Suite. It is even more spacious, with an excellent view of Melbourne CBD. Each Premium Suite has the following characteristics:

- Each Premium Suite always has 3 bedrooms
- Each Premium Suite can be rented for a minimum of 1 day
- The rental rate of a Premium Suite is $554 per day
- About late fee: if an Apartment is returned after the estimated return date, then the late fee is calculated at $662 per day
- Each Premium Suite has a strict maintenance schedule because FlexiRent wants all their suites to be in the best possible conditions. Therefore they specify the following requirements:
  - All Premium Suites must have a maintenance interval of 10 days.
  - Each Premium 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 Premium Suite is available and last underwent maintenance on 15/07/2018. Maintenance must be done for that suite no later than 25/07/2018. Therefore, if a customer wants to rent that suite on 21/07/2018 for 5 days, the FlexiRent system will reject that request.

Implementation Requirements

General Implementation Requirements

- Although you are not required to use more than one class per task, 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.
- It is not necessary to use dynamic data structures to store the input data (i.e. it is fine to define a fixed size data structure taking into account the maximum possible amount of input data).
- Your programs will be marked with Java SE 8.0. Make sure you test your programs with this setting before you make the submission.

Main Implementation Requirements

Your Rental Record class must meet the following requirements:

- Override the **public String toString()** method to print the details of a rental record in the following format:

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

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

  ```
  A_108CRSB_e73581_15072018:15/07/2018:16/07/2018:none:none:none
  ```
  (notice how none strings are used for actualReturnDate, rentalFee and lateFee)

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

  ```
  A_108CRSB_e73581_15072018:15/07/2018:16/07/2018:16/07/2018:350.00:0.00
  ```
  (notice that now all attributes have value)

- 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 SHOULD NOT do the actual printing to the console. Please refer to the following examples:

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

  ```
  Record ID:             A_108CRSB_e73581_15072018
  Rent Date:             15/07/2018
  Estimated Return Date: 16/07/2018
  ```
  (the actualReturnDate, rentalFee and lateFee have no value yet and are therefore not shown)

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

  ```
  Record ID:             A_668BSMEL_CUS0022_15072018
  Rent Date:             15/07/2018
  Estimated Return Date: 18/07/2018
  Actual Return Date:    18/07/2018
  Rental Fee:            957.00
  Late Fee:              0.00
  ```
  (notice that rentalFee and lateFee are printed with 2 decimal places)

Implementation requirements for all rental property classes (Apartment and Premium Suite)

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

- In assignment 1, you are required to use an array to implement that rental record collection, in which the first element of the array is always the latest rental record, the second element of the array is always the second latest rental record, and so on.

- If the rental record array is full, then the oldest rental record (the one at array index number 9) will be removed.Then all rental records in the array will be shifted to the next position in the array while their orders are preserved, and the latest record will be inserted at the beginning of the array (at array index number 0).

The following methods can be called on any object of type Apartment or Premium 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 rental property object (either an Apartment or a Premium Suite) to perform the operations required to rent this property.

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

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

Finally, this method will return true if the property can be rented successfully.

public boolean return(DateTime returnDate)

This method is called on a rental property object (either an Apartment or a Premium Suite) to perform the operations required to return this property.

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

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

Finally, this method will return true if the property can be returned successfully.

public boolean performMaintenance()

This method is called on a rental property object (either an Apartment or a Premium Suite) to perform the operations required to perform the maintenance of that property

This method should check for pre-conditions to determine if maintenance operations can be performed in that property. For example, this method will return false when the property is currently being rented. You should check any other possible conditions which would make this method return false.

If the property is ready for maintenance, this method will perform all necessary operations to update the information stored in this property object when a maintenance happens. Finally, this method will return true if the property is now under maintenance.

public boolean completeMaintenance(DateTime completionDate)

This method is called on a rental property object (either an Apartment or a Premium Suite) to perform the operations required when the maintenance of that property is finished.

This method should check for pre-conditions. For example, when this property is currently being rented, it does not make sense to call completeMaintenance method on this property 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 property object now that maintenance has been finished. Finally, this method will return true to indicate that the maintenance of this property has finished.

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:

```
propertyId:streetNumber:streetName:suburb:propertyType:numOfBedRoom:status
```

(Notice how the colon is used as a separator. If that property is a Premium Suite, the attribute lastMaintenanceDate is appended.)

Example 1: A 2-bedroom Apartment is available for rent, having id A_108CRSB and address 108, City Road, Southbank. Calling toString should return the following line:

```
A_108CRSB:108:City Road:Southbank:Apartment:2:Available
```

Example 2: A Premium Suite is currently being rented, having id S_63WMSB and address 63, Whiteman Street, Southbank, and its last maintenance date is 22/07/2018. Calling toString should return the following line:

```
S_63WMSB:63:Whiteman Street:Southbank:Premium Suite:3:Rented:22/07/2018
```

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 rental property, including details about up to 10 most recent rental records of that property. The returned string should be formatted in a pre-defined human readable format. See the examples below:

Example 1: a new Apartment is available for rent

```
Property ID:     A_108CRSB
Address:         108 City Road Southbank
Type:            Apartment
Bedroom:         2
Status:          Available
RENTAL RECORD:   empty
```

Example 2: an Apartment is currently being rented for the first time

```
Property ID: A_108CRSB
Address:     108 City Road Southbank
Type:        Apartment
Bedroom:     2
Status:      Rented
RENTAL RECORD
Record ID:             A_108CRSB_ABC1234_25072018
Rent Date:             25/07/2018
Estimated Return Date: 28/07/2018
--------------------------------------
```

Example 3: a Premium Suite is currently being rented. It was rented and returned one time before.

```
Property ID:      S_63WMSB
Address:          63 Whiteman Street Southbank
Type:             Premium Suite
Bedroom:          3
Status:           Rented
Last maintenance: 23/07/2018
RENTAL RECORD
Record ID:             S_63WMSB_CUS1108_29072018
Rent Date:             29/07/2018
Estimated Return Date: 01/08/2018
--------------------------------------
Record ID:             S_63WMSB_CUS6237_23072018
Rent Date:             23/07/2018
Estimated Return Date: 26/07/2018
Actual Return Date:    26/07/2018
Rental Fee:            1662.00
Late Fee:              0.00
```

--

Implementing the FlexiRentSystem application class

You are required to implement a class named FlexiRentSystem which will contain one single array to store up to 50 objects of both type Apartment and Premium Suite. Objects of type Apartment and Premium Suite will be added during runtime by the user of your program in the command line using a menu system described below.

Your FlexiRentSystem class should present the following menu for employees of FlexiRent company to manage rental properties. The following menu should be presented to the users:

```
**** FLEXIRENT SYSTEM MENU ****

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

User should 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 Property

The user (an employee of FlexiRent) selects this option to create a new rental property. The user can enter all relevant details of a new property, such as property id, property type, street number, street name, suburb, number of bedrooms, (last maintenance date if it is a Premium Suite)

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

Rent Property

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

Example 1: rent a property which is available for rent

```
Enter your choice: 2
Enter property id: A_668BSMEL
Customer id: CUS0022
Rent date (dd/mm/yyyy): 22/07/2018
How many days?: 3
Apartment A_668BSMEL is now rented by customer CUS0022

****** Rental Company ******
Add Property:          1
Rent Property:         2
Return Property: 3
...............
```
(menu is shown again)

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

```
Enter your choice: 2
Enter property id: A_668BSMEL
Apartment A_668BSMEL could not be rented

****** Rental Company ******
Add Property:          1
Rent Property:         2
Return Property:       3
...............
```
(notice how the menu is shown again, without asking user any further details)

Return Property

By selecting this option, an employee of FlexiRent can then enter a property ID and a return date to return that property. If a return is successful, your program should print all relevant information about that property, including information about the latest rental record. See the example below.

You should perform all necessary data validation, for example, if that property is currently under maintenance, it's not reasonable to return that property.

Example: a property is returned successfully

```
Enter your choice: 3
Enter property id: A_668BSMEL
Return date (dd/mm/yyyy): 26/07/2018
Apartment A_668BSMEL is returned by customer CUS0022
Property ID: A_668BSMEL
Address:     668 Bourke Street Melbourne
Type:        Apartment
Bedroom:     3
Status:      Available
RENTAL RECORD
Record ID:             A_668BSMEL_CUS0022_22072018
```

```
Rent Date:             22/07/2018
Estimated Return Date: 25/07/2018
Actual Return Date:    26/07/2018
Rental Fee:            957.00
Late Fee:              366.85
--------------------------------------

****** Rental Company ******
Add Property:          1
Rent Property:         2
...............
(menu is shown again)
```

Property Maintenance

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

Example: perform maintenance of an Apartment which is not currently being rented

```
Enter your choice: 4
Enter property id: A_668BSMEL
Apartment A_668BSMEL is now under maintenance

****** Rental Company ******
Add Property:          1
Rent Property:         2
...............
```

Complete Maintenance

By selecting this option, an employee of FlexiRent can then enter a property ID to complete maintenance of that property. From the ID, if that property is a Premium Suite, then the system will prompt the employee to enter a maintenance completion date, as shown in the example below:

Example: complete maintenance of a Premium Suite

```
Enter your choice: 5
Enter property id: S_63WMSB
Maintenance completion date (dd/mm/yyyy): 24/07/2018
Premium Suite S_63WMSB has all maintenance completed and
ready for rent

****** Rental Company ******
Add Property:          1
Rent Property:         2
Return Property:       3
...............
```

Display All Properties

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

Start-up Class.

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

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.

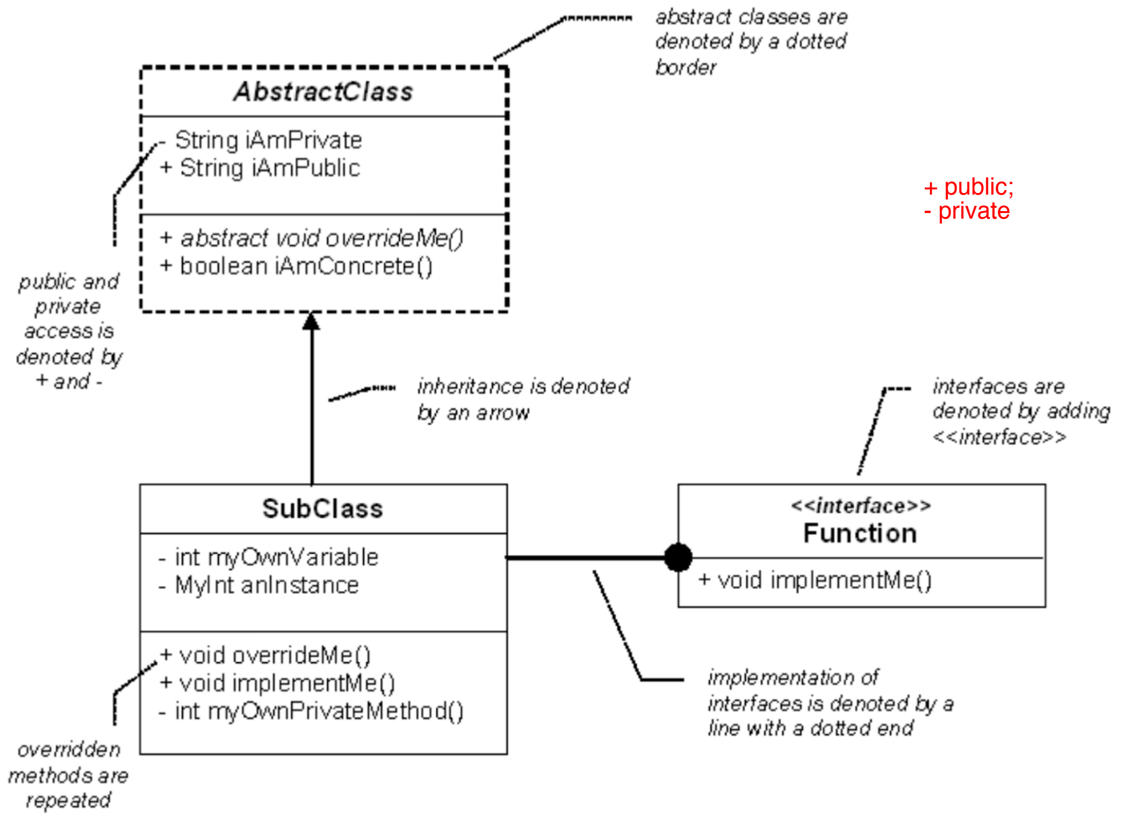
Class hierarchies, relationships and components must be conceptualised in a relevant manner, based on the problem description and special conditions listed in this document. Furthermore, you must be able to explain how your program design will perform in certain scenarios and circumstances.

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.

You will need to answer relevant questions with respect to your design during the demo in Week 5.

Class Diagram Format

The diagram below summarises one way to express common items in a class diagram:



abstract classes are denoted by a dotted border
AbstractClass
- String iAmPrivate
+ String iAmPublic
+ abstract void overrideMe()
+ boolean iAmConcrete()
+ public;
- private
public and private access is denoted by + and -
inheritance is denoted by an arrow
interfaces are denoted by adding <<interface>>
SubClass
- int myOwnVariable
- MyInt anInstance
+ void overrideMe()
+ void implementMe()
- int myOwnPrivateMethod()
<<interface>>
Function
+ void implementMe()
implementation of interfaces is denoted by a line with a dotted end
overridden methods are repeated

Submission Details

You must submit a zip file of your project via Canvas.

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

This assignment has a lab demo component in Week 5. Details will be posted later on the course announcement.

Design documents should be in one of the following formats only and should be named as design (e.g. design.doc, design.pdf, design.gif, design.jpg or design.png)

- Microsoft Word format (must be compatible with Microsoft Word on RMIT CS terminals)
- Adobe PDF format
- GIF, PNG or JPEG image format (please do NOT submit BMP files)

Submission due date: by 11:59 PM on Sunday 19th of August (end of Week 5).

You should compress all your design documents and source code into one zip file and submit only that zip file (no RAR or 7-Zip).

Please do NOT submit compiled files (*.class files), or you will get zero.
You will get zero if the submitted code cannot be compiled.