BIT106 Programming in Java 1
Assignment 2

Due date: **14th November, 2015**

Value: 15%

Rationale

This assignment has been designed to give students the opportunity to demonstrate their skill in:

- solving a fairly complex problem involving the design of more than one user defined class;
- using container classes to maintain an array of objects and allow management of them.
- managing an application which involves a collection of objects
- writing and using methods which enable objects to show desired behaviours
- using and complying with a supplied specifications for classes to be written.
- using good programming style.

Expected Learning Outcomes Assessed

- LO1: use the fundamental control structures and data types used in the Java language
- LO2: be able to apply conceptual and practical skills involved in writing computer programs
- LO3: be able to use software objects and classes, both standard library classes and user defined classes
- LO4: apply the object-oriented approach to design and develop Java applications
- LO5: display a good working knowledge of a Java programming environment

SUBMISSION REQUIREMENT

Your assignment has to submit to TurnItIn, with the following all contain in a single file:

1. All your Java source files, printed in Word document format
2. Printed output (showing your interactivity with your program) is to be included at the end of your Java source files, in Word document created in (1)
3. A Turnitin Report, again to be attached within the Word document created in (1)

Turnitin Report (http://www.turnitin.com)
Register yourself in BIT106 (STIKOM) using the following details:

class ID : **10860736**
Enrollment password: **210915**

Overview

For this assignment you will write a small application which can be used by a hotel staff to manage hotel operations such as booking a room, un-booking a room, displaying vacant

rooms, searching for a specific guest's room number, and displaying summary information about the rooms. A hotel may be considered as containing an array of rooms. The program will allow a hotel staff to interactively do various tasks as stated above.

You are to write the following three classes:

- The first is a class called `Room` which defines a simple object type representing a room.
- The second class called `Hotel` defines objects which are containers of `Room` objects.
- The final class, called `HotelMain` defines a Java application which creates one `Hotel` object and allows the various methods of `Hotel` to be called. This class will be an interactive application using the keyboard and the screen to interact with a human operator. It will not do calculations itself but will immediately pass user inputs as arguments to methods of `Hotel` class.

NOTE: The final application will only execute correctly when all three classes have been defined completely and correctly but don't wait until you have completely written all three before you start compiling and testing your code. It is recommended that you save all three source code files in the same directory on your file system and compile and test each class as you develop it using small separate programs to create and test objects of each class.

The files

The files you will require are:

Room.java

This file defines a class of `Room` objects. The objects have the following instance variables:

- number of beds in the room, of type integer;
- guest's name, of type String;
- booking status of a room – if the room has been booked, the status is 'true', otherwise it is 'false';
- room tariff, i.e. cost of using the room for one night, of type double;

The methods of class `Room` should include:

- A default constructor which does not accept any arguments. This constructor should initialise a Room object with the number of beds as two, the guest's name as "Nobody", the booking status as false, and the room cost to 100.00. This is the default state of a Room object.
- A writer method for the number of beds. This method accepts one argument which is used to set the number of beds. It must ensure that the number of beds stored in the Room object remains in the range 1..4 inclusive.
- A writer method for the room tariff. This method accepts one argument representing a new tariff value and must ensure that the tariff is never negative.
- A method called book which accepts a String argument representing a guest's name. It sets the booking state to true and assigns the parametric string to the guest's name variable.
- A method called unbook which sets the booking state to false and sets the guest's name variable back to "Nobody".
- A reader method for each of the number of beds, the tariff and the guest's name - these return the appropriate value.

- A boolean method called isBooked which returns the booking status.
- A 'toString' method which return a single String containing the details of a room with format as described below:

  `Room with <numOfBeds> beds, tariff <roomTariff>, and guest named <guestName>.`

  or

  `Room with <numOfBeds> beds, tariff <roomTariff>, and is vacant.`

  Example:

  `Room with 2 beds, tariff 100.00, and guest named James Bond.`

  or

  `Room with 2 beds, tariff 100.00, and is vacant.`

It is recommended that once you have written the `Room` class, you create a tiny program to test it. The testing program should be placed in the same working directory as the `Room` class and be used to create one or two `Room` objects and call some of the `Room` methods. Compile the `Room` class and compile and run the test program to check your work.

Hotel.java

This file declares a class of object which maintains a collection of `Room` objects. It will contain methods which enable the collection to show the appropriate behaviours as required by the menu. This file should be saved into the same working directory as `Room.java`.

The `Hotel` class should declare only an array of Room objects, no additional attribute is allowed.

The `Hotel` class must also contain some methods which allow the collection of rooms to be managed. These methods should include:

- A constructor which accepts an integer which is used to set the size of the Room array. If the integer value passed in is invalid, then an array of Room objects of size 50 is to be created. If the parametric integer is valid, that is between 20 and 100, inclusive, then a Room array of the specific parametric value will be created. Next you need to perform some initialisation tasks for the rooms as described below. Each task can be defined as private method, and the constructor will then invoke these methods to complete the task:
  - ✓ First task is to traverse the array and instantiates a default Room object referenced by each array cell. After each Room has been instantiated, we will assume that the array index will represent the room number in the hotel. For example room numbered 2 will be in array cell with index 2, room numbered 5 will be in array cell with index 5, etc.
  - ✓ Second task is to traverse the array and set the room tariff of all the even numbered rooms to $150.00, except room numbered 0, which is set to $1500.00 as it is the penthouse suite.
  - ✓ Third task is to set the number of beds to 1 for the last 5 rooms, and set the number of beds to 4 for rooms 1 though 5 inclusive.
- A method named `getRoom` which accepts an integer parameter representing a room number and returns a reference to the Room object in that cell of the array. If the parametric integer is illegal, a null reference should be returned.
- A method named `numOfBookedRooms` which does not accept any parameter, and returns the number of rooms which are booked.

- A method named `numOfVacantRooms` which does not accept any parameter, and returns the number of rooms which are not booked.
- A method named `totalTariff` which does not accept any parameter, and returns the total value of all the tariffs of all the booked rooms. This simulates one day's income for the hotel.
- A method named `getAvailableRooms` which accepts an integer representing a number of guests which need a room. This method should return a String in which there is a list of all the unbooked rooms which have enough beds for the prospective guests.
- A method named `findGuestRoomNumber` which accepts a String representing a guest's name and searches through all the rooms looking for the first guest whose name is the same as the parametric name. The method should return the number of the room when a match is found. If the name cannot be found, the method should return -1.

When you have written the `Hotel` class - test it by creating a `Hotel` object and invoking the methods from a small Java program.

HotelMain.java

The aim of this class is to provide a user-interface for a modest application which uses a `Hotel` container class and should be saved in the same working directory as the previous files. It is recommended that this user-interface be written as a 'console' application using the normal screen and keyboard to interact with a user via a simple text-based menu.

The user-interface should create a single `Hotel` object and provide a menu of choices to the user with the following choices:

1 See available rooms for 'n' guests

The operator enters the number of guests needing accommodation. This value should then be passed to the getAvailableRooms method of the Hotel object, the returned String captured and displayed. This tells the operator which rooms can be booked.

2 Book a room

The operator enters name of the guest, then fetches the Room object of an appropriate empty room (using the 'getRoom' method) and books it with the guest's name.

3 Unbook a room

The operator enters the number of the room to be unbooked, the Room object with that number is obtained using 'getRoom' and unbooked. If the room is not booked, display an appropriate message.

4 Find which room a guest is in.

The operator enters a guest's name and this is passed to the 'findGuestRoomNumber' method and the room number is displayed. If no such guest is found, display an appropriate message.

5 Print a report

Display

- the number of booked rooms
- the number of empty rooms
- the total tariff of all booked rooms

6 Other options
You may add any other options you like to this list and you will be given credit for those which are implemented correctly.

7 Quit the program.
Each time the user selects one of the previous options, and the program does that task, the menu should be presented again. If they choose to quit, the program should end.

Documentation

- You should include comments in your code stating what each method does and explaining any complex sections of code.
- You should also include your student ID as comments within the code.
- You should of course use meaningful variable names so that your code is to some extent self-documenting.

What To Submit

You should submit the following:

- A cover-sheet stating your student number.
- Printouts of your code;
- Printouts of your program execution.
- A soft copy of a compressed file containing the three files **Room.java**, **Hotel.java** and **HotelMain.java** is submitted to the elearning.
- Write your student number and subject code on the disk.

Marking scheme

Application compile and execute with no errors	80 marks
Quality of code and documentation.	15 marks
Assignment Presentation	5 marks

NOTE: Refer to the Excel file, *ms106A2_SS_15.xlsx*, for detailed breakdown of the marks allocated for each level, as well as the requirement for each level.

Assessment

The most important criteria for assessment are that the three classes compile, produce an application which works and demonstrate that you can manage a project of this size. This will be judged by looking generally at your source code and the printed copy of the output of the program.

Credit will also be given when you can demonstrate that you can maintain the array of objects and write correct methods in the classes.

Credit will be given for a readable coding style which may include a modest amount of inline comments.

Note about testing and plagiarism

It is very important that you complete this assignment alone. You may of course obtain general assistance from the lecturing staff in the subject and your peers, but the coding must be carried out yourself. It is normally quite easy to detect when two or more students work together on their coding.

It is also very important that the demonstration of the results of your program using the given test data is produced using the identical version of the program to the printout of your source code.

Students who hand in substantially similar assignments or whose programs do not match their demonstration of testing will fail the assignment.

Any student suspected of copying, or of not producing the work himself or herself, can be called for **oral examination**, where the student will be expected **to demonstrate sufficient knowledge of the application** to show that it is his or her own original work.