Java 2 Final Project

# Part 1 – Hotel

Create an abstract HotelRoom class that has a int capacity, roomType(enum works well here) Boolean hasMicroFridge, double/ float averageNightlyPrice, and Boolean isVacant fields.

Create a DoubleHotelRooms that extends the HotelRoom the capacity should always be 4. Create a KingHotelRooms that extends the HotelRoom the capacity should always be 3. Create a DoubleSuiteHotelRooms that extends the DoubleHotelRooms the room always has a mircofridge. Create a KingSuiteHotelRooms that extends the KingHotelRooms the room always has a mircofridge.

Create a Hotel class with an ArrayList of HotelRooms. Create a no-arg constructor that will initialize in the ArrayList with 5 rooms of each type. It will set the averageNightlyPrice for a DoubleHotelRooms to $95.50, KingHotelRooms to $96.50, DoubleSuiteHotelRooms to $105.36, and KingSuiteHotelRooms to $104.52. Create bookDoubleHotelRoom, bookDoubleSuiteHotelRoom, bookKingHotelRoom, bookKingSuiteHotelRoom methods that return the averageNightlyPrice of room and sets next available room of that type isVacant field to false. Create a hasVacancy method that will return true if the hotel still has rooms available and false otherwise.

Create getters, setters, equals, and toStrings for all classes.

Create a driver class that will allow the user to book a room of their choosing for x-number of nights and the total for a stay.

# Part 2 – Hotel Revisited

Add to the Part1 of the project the following exceptions: NoVacancyException and OverCapacityException. The NoVacancyException should be thrown when the hotel has no rooms left. The OverCapacityException should be thrown when a guest’s party is too large for the room.

Add the following classes: Guest, Reservation. The Guest class should have the following fields: String firstName, String lastName, String address, int numberChildrenInParty, and int numberAdultsInParty. The Reservation class should have a int reservationNumber, status (Hint: enum works well here), double/float totalCostForTheStay, a roomType (enum works well here), a guest and a hotel room.

Add the following interfaces: BookHotelRoom and ReservationStatus. The BookHotelRoom interface should generate a reservation. (Hint: a Hotel should be able to book a hotel room). The ReservationStatus interface should have a checkIn, and checkOut method. The checkIn method should be called once the guest checkIn and will cause the room to be occupied and set the status of the reservation to valid. Also the room isVacant field should be false. The checkOut method should be called when the user is ready to checkout of a room and will cause the room to be vacant and set the status of the Reservation to invalid.

Create getters, setters, equals, and toStrings for all classes.

Create a driver class that will be menu driven with options to: create a Reservation with a room of their choosing for x-number of nights and print the total for a stay, see all current Reservations for the Hotel, check in and out a guest from a Hotel room. The user should only be using Reservations to book different rooms at the hotel. The programmer should handle any room overcapacity or no vacancies with feedback to the user, and the program should not stop.

# Part 3 – A Reservation Application

Use the classes implemented in Parts 1 and 2 to create a GUI application to accommodate a front desk agent creating reservations, checking in and out guests, and displaying current reservations. Upon startup the application should search for a “hotelRooms.dat”, it should populate an instance of the Hotel Class. (Hint: if the file doesn’t exist create a new instance) Upon shutdown the application should write the current instance of the Hotel Class to a file named “hotelRooms.dat”. The programmer should handle any exceptions and not stop.