CSE 460
Software Analysis and Design
(Fall 2021)

Course Project

Assigned: November 06, 2021
Due: November 27, 2021

If you need help locating your Posting ID, please refer to the post on Discussions board.

Posting ID |__|__|__|__|-|__|__|__|

REVISION HISTORY

Version	Release Date	Major Changes
1.0	11/06/2021	Initial release of description and scope of project
1.1	11/20/2021	See Rev. 1.1 notes/updates

Please ask early any questions or concerns about this document and the project itself – it not only helps you but all your classmates and us!

Rev. 1.1: Updates and notes for the Availability Demand Software System (ADS)
The following restricts the design, implementation, and testing:

1. The Availability Demand System notifies the customers about room availabilities only. That is, the system does not support booking rooms or making reservations.
2. If a customer has received the notification for some availability period and location he has subscribed to from a certain BNB provider, then the customer should not receive any duplicate notifications (e.g., the provider publishes the same room again).
3. For testing purposes, it is expected that all the dates either published by BNB providers or requested by customers are later than a default date provided by ADS. The default date is 11/27/2021.
4. The system tracks all published events. Whenever a new customer uses the system, he receives notifications for published events that satisfy the subscription.
5. All published and subscription events are unique.

6. When a B&B announces a room availability, any subsequent room published by this same B&B is discarded if its availability period overlaps with that of the room published earlier.
7. Any customer can have subscriptions with overlapping stay periods, and the subscriptions are unique.
8. The AvailabilityDemandTests.java file is provided. It has sample test cases.
9. The AvaiblityDemand.java file is provided. It serves as a template.

Availability-Demand Software System for Bed&Breakfast

Bed & Breakfast accommodations, called B&Bs, can broadly be defined as small independent properties offering overnight lodging and breakfast in a home-like setting. Customers seek accommodations at different locations for different stay periods (arrival and departure dates). B&Bs should provide information on their properties to potential customers and, therefore, the opportunity to have their business prosper. Customers should only know about B&B availability at least one day prior to their arrival dates. An **Availability-Demand Software System (ADS)** is crucial to serving both B&Bs and customers. It is expected for existing B&Bs to stop offering accommodations while other properties start to offer their properties to potential customers.

For example, a customer wants a B&B in New York City for four nights starting on Dec. 1st, 2021. The customer should be able to know all the B&Bs that have availability for his preference through Dec. 1st, 2021. The customer can then decide to stay at a B&B if at least one is available. For instance, several B&Bs are available in NYC but for the dates after Dec. 10th, 2021. In this scenario, these B&Bs are of no use to this customer. The customer does not need to know about these B&Bs.

Every B&B can announce **(*publish*)** its location and stay period. The customers can look for **(*subscribe*)** B&Bs that have availability for their desired locations and stay periods. When there are one or more B&Bs available for what a customer is looking for, then the customer is notified. Customers receive notifications *only when the* ADS finds matches (location and stay periods) at the B&Bs that match customers 'needs. Customers can choose to stop receiving notifications (i.e., unsubscribe for what they have previously subscribed to) at any time.

Scope

Design and implement the Availability-Demand Software System. This system should support three types of events: "*publish*", "*subscribe*", and "*unsubscribe*". Each subscribe event includes a location and a stay period in which a subscriber is interested. Similarly, unsubscribe event includes a location and stay period. Each publish event includes the B&B location and the dates for which it is available.

When a publisher publishes an available B&B for a certain location and a stay period, all subscribers (customers) that are currently subscribed to that location will receive a notification if and only if the availability of B&B falls within the customer's stay period need. In this project, this notification to any customer is one or more B&Bs (additional information can be found in the **Coding and Testing** section). In addition, the stay periods and locations are not known to the customers prior to subscription.

Once a customer unsubscribes, it will no longer receive any notifications unless the customer re-subscribes to the same B&S location and stay period. The subscribe, unsubscribe, and publish events are to be processed in the strict first-come-first-serve sequential order. This is required to have a fixed order of notification events sent to subscribers.

Analysis and Design

The design for the system must follow the *publisher-subscriber* design pattern. This pattern has a **broker** to manage the interactions among any number of publishers and subscribers**. The design and implementation must guarantee the publishers and subscribers do not have any direct relationship with one another.** Furthermore, every publisher is independent of all other publishers,

just as every subscriber is independent of all other subscribers. This means that both the structure of your system and the behavior of the components of your software system strictly conform to the publisher-subscriber design pattern. Include useful pre- and post-conditions, at least for the operation needed for notifying subscribers. The report must include the specification and code for this operation.

You need to develop your UML specifications using Astah with its forward and reverse engineering features. You should include an appropriate number of use-case, class, sequence, and state machine diagrams.

The B&B systems should implement the provided interface, **IPublisher**, shown in Figure 1.

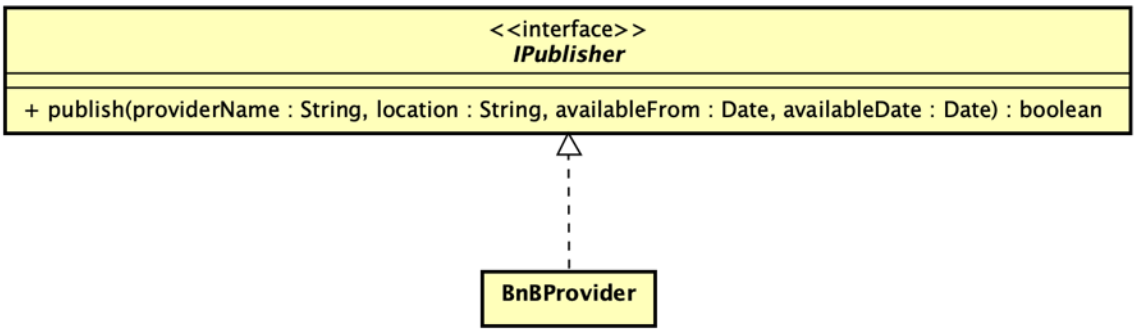

Figure 1: Partial class diagram for the publisher in the Publish-Subscribe design pattern

Similarly, the customers (for different request parameters) should implement **ISubscriber**, as in Figure 2. You may implement additional methods, but they will not be used in the testing process.

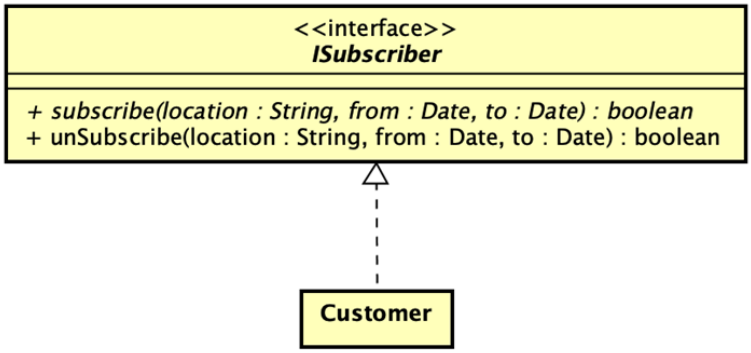

Figure 2: Partial class diagram for the subscriber in the Publish-Subscribe design pattern

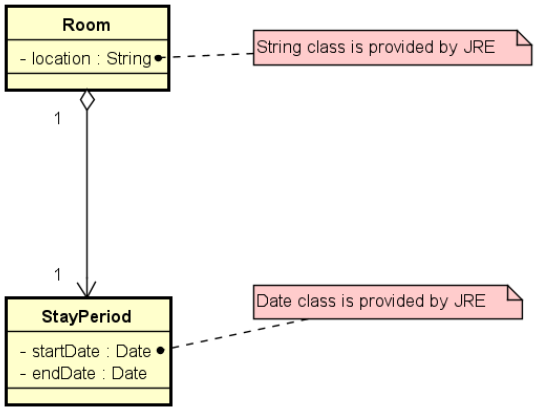

Figure 3: Partial class diagram for Room and StayPeriod classes

Coding and Testing

You must use forward engineering to export your UML design specification to Java code. Then, you must complete the partially generated Java code by adding your own code to the stubs generated by Astah. Do not delete or change any of the Java code produced by Astah. If you need to change any line of code from Astah has generated, such as for a return statement, leave a commented copy of the original line in the program.

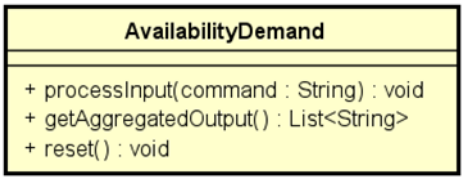

Figure 4: AvailabilityDemand class skeleton

Your program will be tested using a class called **AvailabilityDemand**. The following is a general testing procedure for running a set of test scenarios.

1. An AvailabiltiyDemand object is instantiated.
2. The processInput() operation can be called sequentially multiple times, each time with a single input (see below). The three publish, subscribe, and unsubscribe inputs can be interleaved. All inputs are for the Availability-Demand software system. Note that not every processInput() operation may necessarily result in a notification sent to one or more subscribers.
3. The getAggregatedOutput() operation returns a list. This output list can have zero or more entries depending on the specific processInput() operations included test scenarios. This operation also acts as input for testing the Availability-Demand software system. The output lines in the returned value of this input must be sequentially ordered according to the order of the processInput() operations (see included test scenario). Do not add trailing newline characters (\n) to the output.

4. The result from the previous step will be compared against a reference answer.
5. Finally, the reset() operation will be called to clean up all information in the system. After resetting, the system starts anew. The program should **NOT** carry over information from previous rounds to the next one.
6. Repeat from step 2. Each round of execution corresponds to one test case.

The input and output formats for the Availability-Demand software system:

Input: [command], <arguments>

- publish, [BNB provider],[location], [available from date], [available until date]
- subscribe, [customer name], [location],[fromDate], [toDate]
- unsubscribe, [customer name], [location],[fromDate], [toDate]

Output: [customer] notified of B&B availability in [location] from [available from date] to [available until date] by [B&B provider name] B&B.

Below is an example for testing the design and its implementation:

Customer (name of a customer who subscribes to receiving B&B availability notifications): Jon Doe, Jane Doe;

Location (location the customer is interested in): New York City;

Available from date, (date in MM/DD/YYYY format from which B&Bs are available and customer is interested in): 12/01/2021;

Available until date, (date in MM/DD/YYYY format from which B&Bs are available and customer is interested in): 12/15/2021;

B&B (name is Hight-Mountains)

Sample input	`processInput("subscribe, John Doe, New York City, 12/01/2021, 12/05/2021");`
	`processInput("publish, High-Mountains, New York City, 11/30/2021, 12/15/2021");`
	`getAggregatedOutput();`

Sample output (a String list with one entry)	`John Doe notified of B&B availability in New York City from 11/30/2021 to 12/15/2021 by High-Mountains B&B`

Table 1: Sample input and output

Customer/B&B provider names and location names may contain any character except comma and leading/trailing spaces. It is your responsibility to handle the upper-case/lower-case names (i.e., treating them to be the same, such as "High-Mountains" and "High-mountains"). You do not have to preserve the letter casing in your output. However, the characters should exactly match our output (no extra space, lines, or additional characters) as all programs will be tested automatically.

Additional sample test cases are provided. AvailabilityDemandTests.java contains some test cases as well as a complete JUnit test setup similar to the one used in automated tests. To run it, you need to place the file in the test folder under AvailabilityDemand and set up JUnit 4.12. These sample test cases, in addition to others, will be used in the grading of the program.

Your program should not produce any output other than notifications sent to customers. Therefore, please do not include a welcome message or a menu in your output. Additionally, the program should **NOT** produce any error messages during the entire execution. For example, when an illegal command is entered or a customer is trying to unsubscribe from a location they are not subscribed to. Your program should not crash throughout the execution.

Important! All classes must be in a package named AvaialbilityDemand (i.e., all *.java files are in folder AvailabilityDemand). Figure 3 shows a directory structure (including the JUnit test) under this requirement. *Note that this is for illustration only and does NOT imply you have to have this number of classes or that you must have a Utils sub-package.* **Failure to comply with this requirement will result in failure of compilation in the automated grading system.**

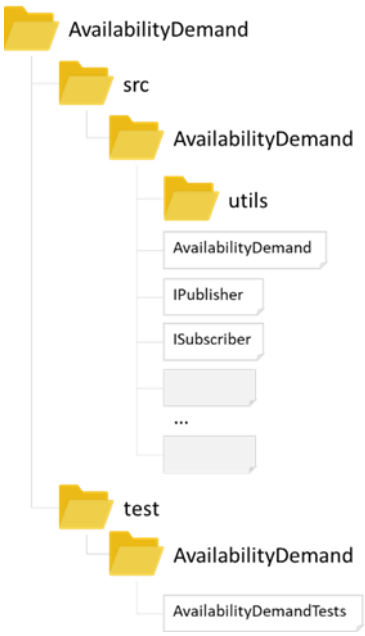

Figure 5: Directory structure for implementation

Some notes regarding the testing environment:

- J2SE must be used — J2EE is not allowed.
- No compiled code or third-party APIs are allowed.
- Source code can be implemented only using the Java programming language.
- Your submission will be tested automatically on GradeScope in an Ubuntu 18.04 virtual machine. Your source code will be compiled and executed using JRE 11.
- After each submission, you will see detailed reports. You may notice some random letters and numbers in place of publishers, subscribers, etc. – this is normal. Test inputs are randomly generated each run.
- You should not need to worry about the actual input, but rather the design with its implementation works under different B&B supply and demand scenarios.

Make sure your code compiles and matches the given interfaces.

Submission

- Submit your **source code** in a zip file. It should contain a folder named AvailabilityDemand, in which AvailabilityDemand.java and other classes are located (in other words, when you open the zip file, you should only see AvailabilityDemand folder). Upload this zip file to the project **GradeScope** page (you can also find the link to GradeScope in the project page on Canvas).
- Submit your **project report** to **GradeScope**. A new Canvas page named Project Report Submission was created and it will take you to the right GradeScope page. Note that you need to mark up the sections accordingly.
- Submit your **Astah file** to Canvas. You should name this file *postingid*.asta (e.g. 1234-987.asta). See the rubric below for information on the project report. Submit this Astah file to **Canvas** - Project Astah Submission.

Project Rubric

Parts	Points
Uses publisher-subscriber model	20
Use case diagrams	5
Class diagrams	15
Sequence diagrams	10
State machine diagrams	10
Design quality	5
Forward engineering (partial code is generated automatically)	15
Produces correct outputs	20
Code documentation	5
Project report quality: this is for the full project report which contains UML specifications with *their descriptions* and other supporting materials. The report is a single PDF file. A template for this document is provided.	5
Total	110

Note: 10 out of 110 are bonus points.

Important note: be sure to check the Canvas Discussions for updates, questions, answers, and hints. You are responsible for knowing any posted requirements or guidelines. If you are not certain about some aspects of this programming assignment, it is very important to ask early, not a few days before the project due date.