CSE 460
Software Analysis and Design
(Fall 2019)

Programming Assignment

Assigned Date: October 28, 2019
Due Date: November 18, 2019

Submit your report (see the report template), source files, and Astah files in a single ZIP file (no other archive formats will be accepted) to Blackboard. The report name is CSE460_postid.zip (replace "postid" with your ASU posting ID). You must include your student ID number as a comment at the beginning of every source file. Submit your solution in the following format: **CSE460_postid.zip** (replace "postid" with your ASU posting ID).

Description of a Social Networking System

A social networking website allows setting up social gatherings of interest (e.g., Earth Day) by its members. A member acting as an *author* can post social gathering announcements for any number of members. Every member of the website can sign-up to receive announcements that have one or more of his/her keywords of interest. In other words, the members of the website can choose to be notified of new topics related to the keywords of their choosing. The website notifies every member who has an interest in any keyword of an announcement. A member may choose to opt-out of notifications for specified keywords at any time. Both Authors and members may create keywords. Members only receive notifications for announcements sent after they have chosen to be notified.

Scope
Design and implement a software system that sends announcements to website members based on the keywords. Authors advertise topics through the interface `IPublisher`. Authors who post topics are *publishers*. Members find about the advertised topics of interest through the interface `ISubscriber`. Members who are interested in keywords are subscribers. The actions of the authors and members may have an arbitrary order. If a keyword in a command (i.e., publishing or subscribing) does not already exist, it is to be added to the system. When an announcement is added, every subscriber to any of the matching keywords should be notified. The subscriber will then add a message to its log that has the name of the member who received the message, the person who posted, the text of the message, and all ~~matching~~ keywords from the message. To simplify and clarify the assignment, log entries should have all keywords from the message, in the same order, as stated in the format for a log entry below. The message has this format:

[Member name] received an announcement from [Name of author].
Text: [Text of announcement]
Keywords: [Keywords of announcement].

A newline (\n) separates the lines of the log entry. The list of keywords must be comma-separated, with one space after each comma, and no comma after the last keyword. The order of the keywords in the list matches the order of keywords in the announcement.

The design for the system must follow the *publisher-subscriber design pattern*. This pattern has a *broker* to manage the operations of the publishers and subscribers while ensuring they do not have any direct

relationship to one another (i.e., all publishers and all subscribers are independent of each other). This means that both the structure of your system and the behavior of the components of your system must match the publisher-subscriber design pattern.

No error message occurs if a member tries to unsubscribe to a keyword to which the user is not subscribed, even if the keyword does not exist. There is also no error message if a member subscribes multiple times to a keyword, but there is no such thing as a duplicate subscription. There are no duplicate announcements.

The `Author` class must realize the `IPublisher`, as shown in Figure 1.

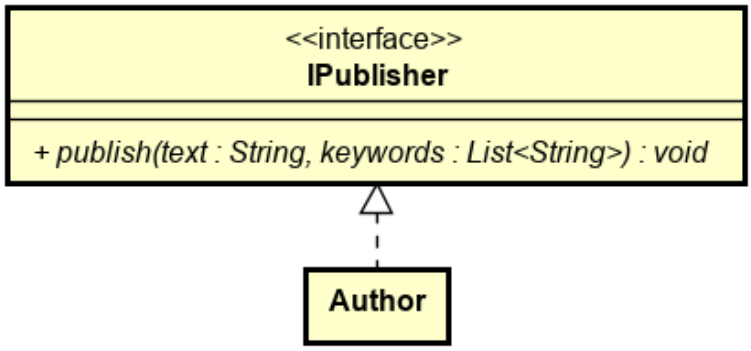

Figure 1: Partial class diagram for the publisher in the Publish-Subscribe design pattern

The `Author` class must have a public constructor with the signature

```
Author(String name)
```

Every announcement topic text with its keywords are posted using the `publish(…)` method.

The `Member` class must realize the `ISubscriber` as shown in Figure 2.

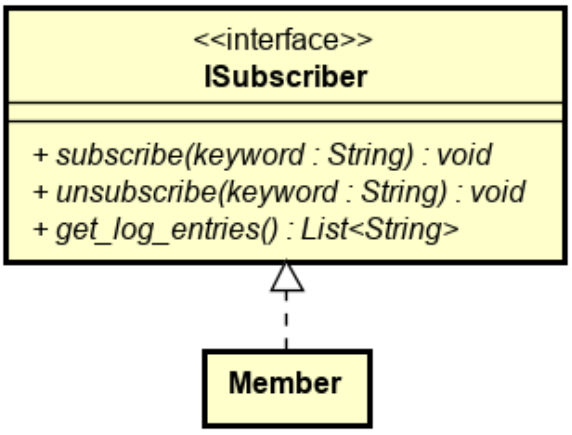

Figure 2: Partial class diagram for the subscriber in the Publish-Subscribe design pattern

The `Member` class must have a public constructor with the signature

```
Member(String name)
```

The `subscribe(…)` method causes the subscriber to receive all future messages with the given keyword. The `unsubscribe(…)` method removes a subscription for the keyword if the subscription exists. ~~The `notify(…)` informs the subscriber that a message has been sent.~~ The `get_log_entries(…)` method returns a list of log entries for the subscriber's messages, in the order the messages were received.

For the sake of simplicity, you may assume that separate Member and Author objects are used if one person acts as both a publisher and subscriber.

Analysis and Design
You need to develop *analysis and design specifications* using the UML standard and the Astah tool prior to implementing them. You should include an appropriate number of class and sequence diagrams and at least one state machine diagram.

You must use forward engineering to convert your UML design specification to Java code. Then you must complete the partially generated Java classes by adding your own code to the stubs generated by Astah. Manually added code must be identified using **// Begin** and **// End** comments. The UML diagrams must completely match your code to the degree possible. Do not delete or change any of the Java code produced by Astah. If you need to change any line of code from Astah, such as for a return statement, leave a commented out copy of the original line in the program.

The system will be given commands through the interfaces. During the execution of the program, the Member objects must each keep a log of all the messages the member has received. The log entries can be retrieved at any time by using the `get_log_entries()` method. If no entries exist in the log, `get_log_entries()` should return an empty list (not a null object). The log entries may be considered the outputs of the software system. The entries must have the exact format stated above.

Automated test cases will be used to evaluate the programs. Please be sure that all of the above specifications are met. Spelling, punctuation, and spacing are important for method signatures and output messages.

Coding and Testing
A set of JUnit test cases will be used to evaluate the correctness of your system for different situations. These test cases will instantiate objects of `Author` and `Member` and call methods by using the interfaces. The log entries from the `Member` objects will be compared to expected entries. The test passes only if all the entries are exactly the same as expected. Any type of exception, error, or unexpected output results in a failure for the test.

A few sample test cases are provided. These test cases, in addition to others, will be used in the grading of the program.

- **Note that J2SE must be used — J2EE is not allowed.**
- We will execute your source code on Windows 10 OS.
- Source code can be implemented only using the Java programming language.
- We will execute your source code using JRE 1.8. No compiled code or third-party APIs is allowed.
- Make sure your code compiles and matches the given interfaces and class constructors.

Rubric

Parts	Points
Uses publisher-subscriber model	25
Use case diagrams	5
Class diagrams	8
Sequence diagrams	8
State machine diagram	8
Design quality	6
Forward engineering (partial code is generated automatically)	10
Produces correct outputs	20
Code documentation and quality	5
Documentation: this is for the full project report which contains the class, sequence, and state machine specifications with *their descriptions*, a readme file (including the name of the main class), and any other supporting materials. A single PDF or Microsoft Word file must be used. A template for this document is posted.	10
Total	105

Note: 5 out of 105 points is bonus.

Be sure to check the Canvas discussion board for updates, questions, answers, and hints. You are responsible for knowing any posted requirements or guidelines. If you are not certain about some aspect of this programming assignment, it is good to ask.