

CSE 460 Programming Assignment

Computer Science Program

Arizona State University, Tempe, AZ

Matthew Sarantos

Spring 2016

Assumptions

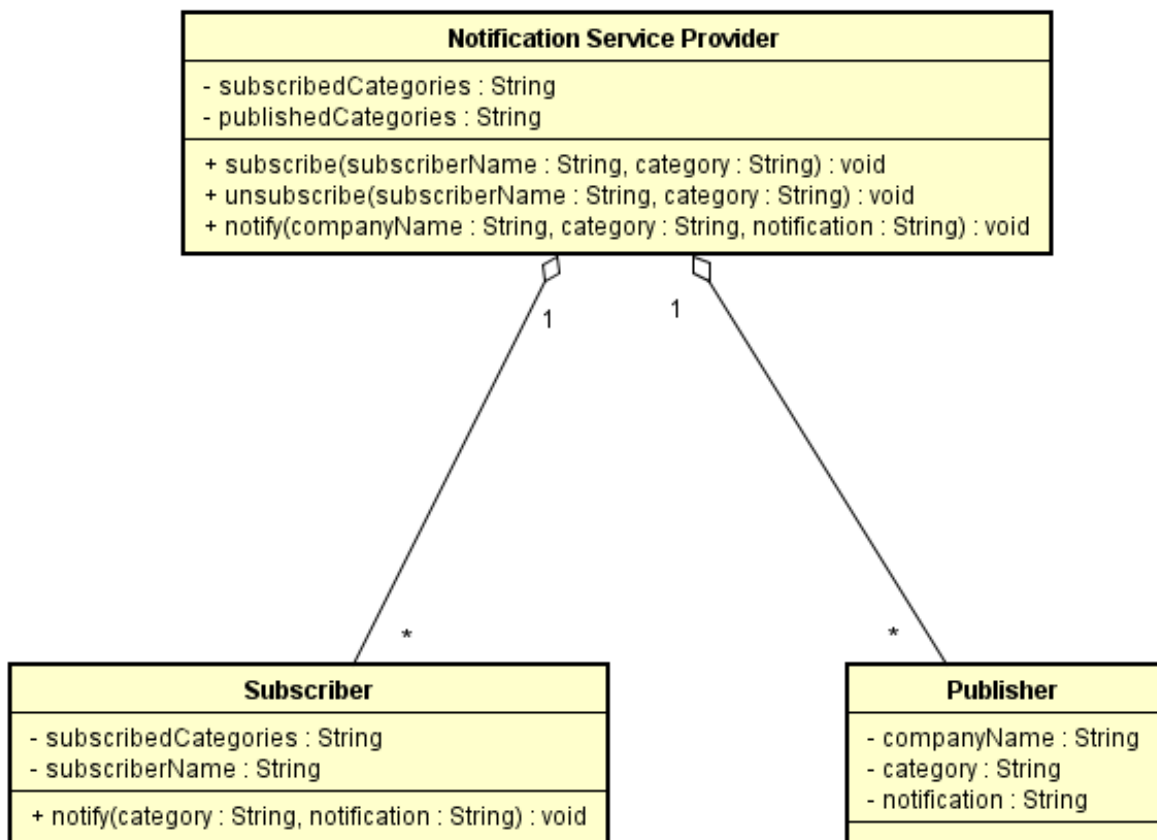
There were a few different assumptions made when designing the model and coding it as well. In the implementation side if a publisher or a subscriber entered a name (either their own or a category) The capitalization of characters did not matter. “Jared” was the same as “jared” and “Rock and Roll Hall of Fame” was the same as “rock and roll hall of fame”. Another implementation assumption was that the events would output the same order they were processed in the input to avoid confusion and have one certain output instead of many different possible combinations. The last assumption made for the implementation process was that the subscriber would only receive notifications that were output after they had subscribed to a category. So if a publisher put out a notification before they subscribed to a category the user would not receive that notification or any previous notifications.

1. Publisher-subscriber

Some of the thoughts that went into the whole design process revolved around the publish-subscribe model used here. It was a data-centric approach where Podcast.java listens from publishers with Publisher.java attributes and then outputs the information to Subscribers who are subscribed with Subscriber.java attributes and does all of this with the notify() method. The advantages of this, as was told in CSE 460, is that you have an enhanced response time, the model is scalable to larger numbers of publishers and subscribers, and there is a loosely coupled relationship between publishers and subscribers.

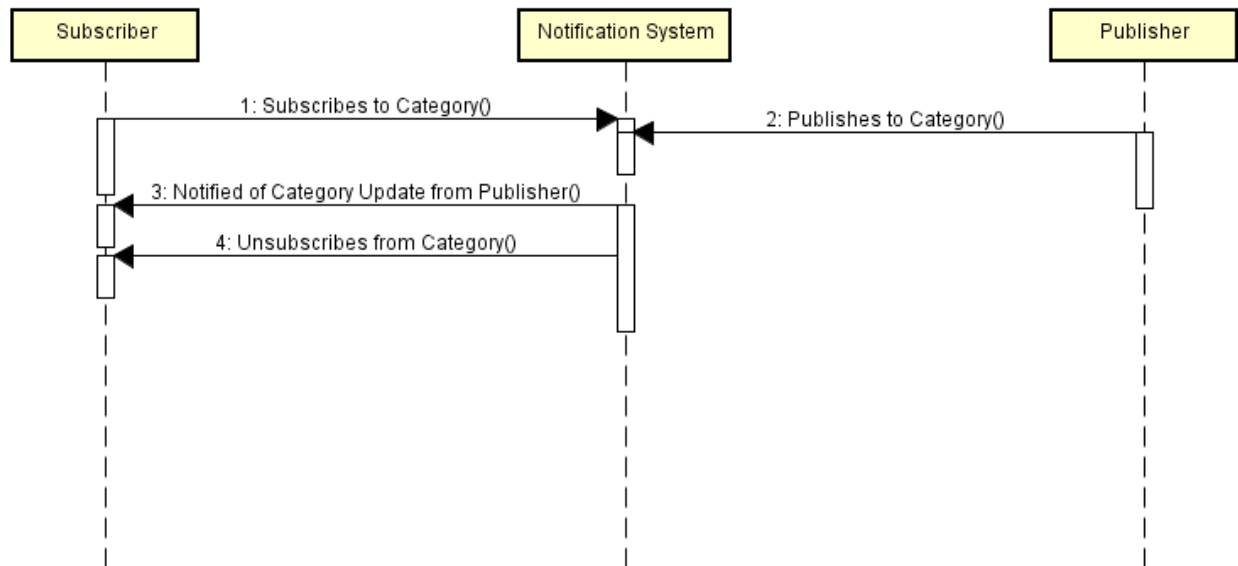
2. Class diagrams

For the class diagram I based it off the power point given to the class for the project. There would be one main Notification Service Provider and then there could be many different Publishers and Subscribers. This way the project becomes easily scalable since you can just add and remove Subscribers and Publishers. The Notification Service Provider would also serve as a way to store Publishers attributes and the Subscriber Attributes and then notify certain Subscribers with the Publishers companyName and notification Strings depending if their categories matched and the Subscriber was subscribed.



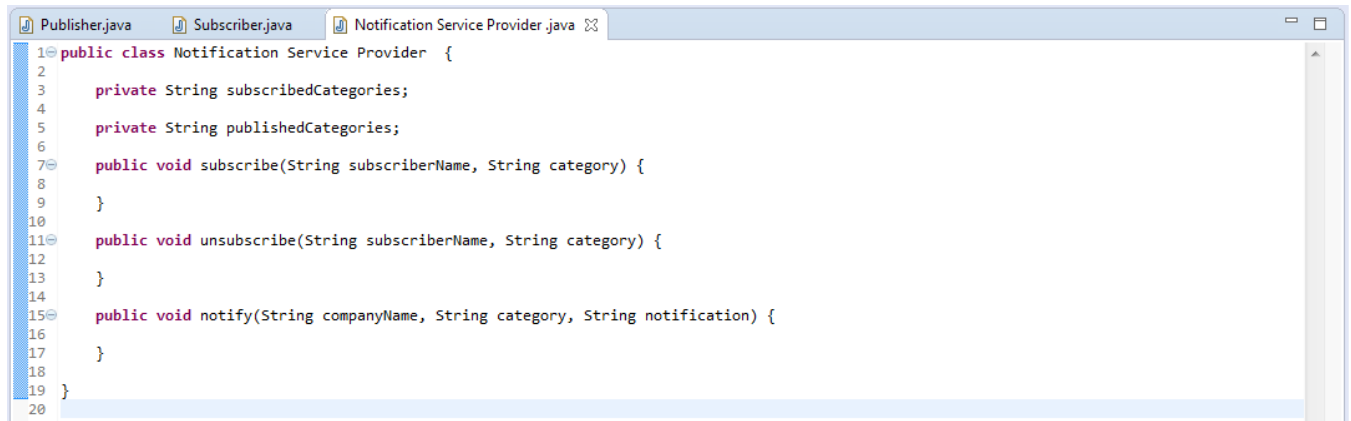
3. Sequence diagrams

Since the Sequence Diagram is just going to describe the behavior the model created just shows the basic functions of each class and what each class can actually do. The subscriber can subscribe to a category or even unsubscribe if they no longer choose to receive notifications from publishers with this category. Publishers can only publish to a category with a companyName String and a notification String. After the notification is sent to the notification system the subscribers are then notified of the update via the notify() method. As stated before, after subscribing the subscriber can unsubscribe at any time.

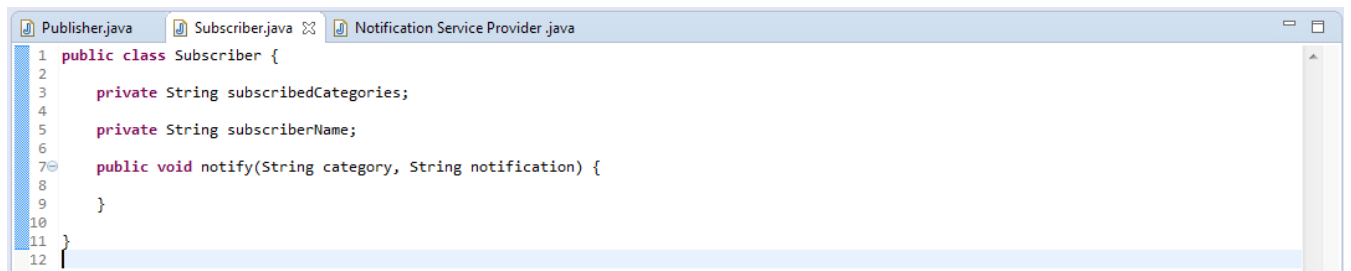


4. Forward engineering


To export Java code stubs for every class and forward engineer a basic skeleton of variables, methods, and classes all that was done was [Tool] - [Java] - [Export Java] in the main menu of Astah. The following three classes were the ones generated from the UML design. All that needs to be done to this code skeleton is the main Notification Service Provider class needs to be renamed to Podcast.java for the automated test cases.



```
1 public class Notification Service Provider {
2
3     private String subscribedCategories;
4
5     private String publishedCategories;
6
7     public void subscribe(String subscriberName, String category) {
8
9     }
10
11    public void unsubscribe(String subscriberName, String category) {
12
13    }
14
15    public void notify(String companyName, String category, String notification) {
16
17    }
18
19 }
20
```



```
1 public class Subscriber {
2
3     private String subscribedCategories;
4
5     private String subscriberName;
6
7     public void notify(String category, String notification) {
8
9     }
10
11 }
12
```



```
1 public class Publisher {
2
3     private String companyName;
4
5     private String category;
6
7     private String notification;
8
9 }
10
```

5. Readme

Podcast.java is the main class and also uses classes Subscriber.java and Publisher.java. To compile the program from the terminal use the following two commands:

```
javac Podcast.java
java Podcast <inputFileName>
```

The input file has to be in the form of one of the following:

```
Subscribe, <name>, <category>
Unsubscribe, <name>, <category>
Publish, <company name>, <category>, <notification message>
```

The program will take the input and store the values in separate tokens depending on the first keyword and will store the token values in the proper new object. The object, if it is a subscriber, will be stored in an arrayList containing the name and category of the user. If the Publish keyword is used a notification message and company name will go out to all subscribers who are subscribed to that topic.

Appendix

The files in the zip include the following:

- This PDF (Programming Assignment CSE 460 S16 Matthew Sarantos.PDF)
 - A document which contains the read me and design process
- 2 Astah files
 - 1 for the UML
 - 1 for the Sequence Diagram
- 3 Java Source Code Files
 - Podcast.java
 - Subscriber.java
 - Publisher.java