**CSE 460/598**

**Software Analysis and Design**

(Spring 2016)

**Programming Assignment**

**Assigned Date:** March 30, 2016

**Due Date:** Apr. 13, 2016

Submit your report, source files, and Astah files in a single ZIP file (no other archive formats will be accepted) via Blackboard. You must include your name in the report and in every source file. Please do not include compiled code.

Submit your solution in the following format: **[FirstName][LastName].zip**

This programming assignment is aimed at applying your knowledge in model specification methods for design and using them to implement and test an exemplar Publisher/Subscriber program for a podcast website described below. All UML diagrams must be created using Astah. You will use forward engineering to convert your UML diagrams to Java code using Astah. Then you will complete the implementation by adding your own code to the stubs provided by Astah. Note that J2SE must be used—J2EE is not allowed.

**Problem description**

Consider a podcast directory website. An example of this would be [www.podcast.com](http://www.podcast.com). Assume such a website makes video podcasts in various categories (e.g., Rock Music) to users. Categories are accessible via channels (publishers). Users can subscribe to and unsubscribe from categories. Users will receive notifications whenever new podcasts become available in the categories to which they have subscribed to. For example, assume that *Ville* has subscribed to the “Rock Music” category. When a new podcast is released in this category, *Ville* will be notified. Develop this software system using the Publisher/Subscriber Design Pattern.

The software system will be tested using events contained in a comma-separated value (CSV) input and output files. There are 3 event types: “Publish”, “Subscribe”, and “Unsubscribe”. Each subscribe event includes a subscriber name and the category name in which the subscriber is interested in. All subscriber names are unique. Similarly, each unsubscribe event includes a subscriber’s name and a category name. Each publish event includes a publisher name, category name, and a podcast name. All publisher names are unique. Subscriber, publisher, podcast category names, and podcast names may contain any character other than comma. The design and implementation must handle the upper-case/lower-case names (i.e., treating “Rock and Roll Hall of Fame” and “rock and roll hall of fame” the same). The output file as in the input file can have lowercase/uppercase names. The characters must exactly match our test output CSV file (no extra space, lines, or additional characters) as tests will be done automatically. A sample input/output is provided in the remainder of this document.

Events in the input file are to be processed in the order given. In particular, a user will not receive notifications for a podcast unless they subscribe to the category (and until they remain subscribed). Also, users will not receive any notifications prior to their subscriptions or if they unsubscribe from categories. Multiple subscribers to a category are notified in the order in which they subscribed (in order to fix the output order) to categories. Repeat subscriptions or un-subscriptions to a category must be disregarded.

**Example run**

Suppose the following events are specified in the input:

Subscribe,Keenan,Space News

Subscribe,Jared,Politics

Publish,PBS,Space News,Space Time e.12

Publish,NASA,Space News,nasa public

Publish,CSPAN,Politics,White House Briefing

Subscribe,Ville,literature

Subscribe,Kirk,Rock Music

Publish,BBC,literature,The Secret Life of Books

Publish,CNN,Documentary,High Profits

Publish,VH1,Rock Music,Behind the Music

Unsubscribe,Keenan,Space News

Publish,BBC,Space News,Wonders of the Solar System

Then the following results are expected (this is in all lower-case):

keenan notified of space time e.12 from pbs

keenan notified of nasa public from nasa

jared notified of white house briefing from cspan

ville notified of the secret life of books from bbc

kirk notified of behind the music from vh1

**Design**

You need to design the software system using UML prior to implementing it. You should include at least a class diagram and a sequence diagram. The code which is manually added is needed to be identified using the following tag: //**@Begin** and **//@End.**

**Testing**

During testing, your program’s input will be provided and the output captured via redirection. The produced output file for the sample input should match the sample output exactly for automated evaluation. Note that the test input(s) will differ from the example input and may be larger.

If the TA has any problem with compiling or running a program locally, it will be tested on general.asu.edu (which has Java 1.6.0\_20-b02).

You can use the “My Files” link within MyASU to transfer files to/from general.

To test your code on general:

Use an SSH client (e.g. PuTTY in Windows) to connect to general.asu.edu with your ASURITE ID and password.

To compile: javac *Podcast*.java

To run: java *Podcast* < *inputFile* > *outputFile* (substitute the file names)

**Format**

Since we are testing your programs using a PHP script, you have to make sure that you comply with the following:

1. Your main class should be named: *Podcast.java* (this is only the main java file. You need other classes in order to implement the entire software system)
2. The Podcast.java receives the names of the input and the output files from command line. The PHP script will compile and run your code using the following commands:

javac *Podcast*.java

java *Podcast* < *inputFileName* > *outputFileName*

1. Make sure your output file does not contain any extra characters
2. Submit all your Java classes as a ZIP file (do not include input/output, .class files, etc.); **File name conversion**: [FirstName][LastName].zip
3. Make sure it is your own work

**Rubric**

20 points: uses publisher-subscriber model

10 points: class diagrams

10 points: sequence diagrams

5 points: design quality

10 points: forward engineering (partial code is generated automatically)

5 points: compiles

10 points: runs without errors on all test inputs

20 points: produces correct output

5 points: code quality

5 points: Documentation: a readme file (including the name of the main class), and any other supporting materials. Acceptable document format is PDF. A template is posted.

Total: 100 Points

**Input file:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Order** | **Subscriber** | **Unsubscriber** | **Category** | **Publisher** | **Product** |
| 1 | Keenan |  | Space News |  |  |
| 2 | Jared |  | Politics |  |  |
| 3 |  |  | Space News | PBS | Space Time e.12 |
| 4 |  |  | Space News | NASA | nasa public |
| 5 |  |  | Politics | CSPAN | White House Briefing |
| 6 | Ville |  | literature |  |  |
| 7 | Kirk |  | Rock Music |  |  |
| 8 |  |  | literature | BBC | The Secret Life of Books |
| 9 |  |  | Documentary | CNN | High Profits |
| 10 |  |  | Rock Music | VH1 | Behind the Music |
| 9 |  | Keenan | Space News |  |  |
| 10 |  |  | Space News | BBC | Wonders of the Solar System |

**Output File:**

|  |  |  |
| --- | --- | --- |
| **Order** | **Subscriber** | **Event** |
| 1 | keenan | space time e.12 |
| 2 | keenan | nasa public |
| 3 | jared | white house briefing |
| 4 | ville | the secret life of books |
| 5 | kirk | behind the music |