THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON, ONTARIO, CANADA

Computer Science 2212b
Introduction to Software Engineering - Winter 2014

Team Assignment 2

Due February 28 at 11:59 PM

1 Overview

In this assignment, your team will develop a prototype, determine how data will be saved, and what classes will be required.

Since we are following an agile development methodology, you must complete some of the project requirements for this assignment. Specifically, for your prototype, you will be asked to complete six of your user stories.

The TA must be able to test your prototype, so the code in your repository that is tagged `asn2` must compile using `mvn package` and the TA must be able to run the resulting JAR file.

In addition to the code, you will need to include instructions for the TA on which user stories you completed, and how he/she can test the completed stories.

2 PDF Report

You must submit a PDF report containing the following:

- Title page, in the same format as that required for Team Assignment 1
- Section 1: Revised UML Class Diagram
- Section 2: Revised User Stories
- Section 3: Class Descriptions
- Section 4: Project Plan Documentation

2.1 Revised UML Class Diagram

This section should include a revised and updated version of the UML class diagram submitted for team assignment 1. Your diagram should capture ALL the system's functionality, as specified in the project specification document.

In addition, your diagram should include details such as attribute types, method parameters, and return types, and should identify which attributes and methods are public, private, etc.

In order to make it easier for your teaching assistant to evaluate your new UML class diagram, **make sure to use a different colour and/or font and/or weight in your diagram** to distinguish between the updates you have made to the new class diagram versus the class diagram you submitted for team assignment 1.

This class diagram should match up with the class descriptions in section 2.3. That is, any methods listed in this class diagram must be also listed in the class descriptions.

2.2 Revised User Stories

You must update your PivotalTracker and indicate which stories are complete. Additionally, if you were missing stories on Team Assignment 1, you must add the appropriate stories to your tracker.

Finally, take a screenshot of your tracker so that we can see:

- Done
- Current
- Backlog
- Icebox
- Epics

Include this screenshot in this section, along with a textual link to your tracker. Furthermore, this section must list the user stories that have been completed since the last assignment, and provide instructions on how the TA can test each user story in your prototype (e.g. what to click on, how to navigate to the implemented functionality, etc.).

2.3 Class Descriptions

This section should show how you are going to translate your class diagram into Java code. Thus, a many-to-one association between, say, *Employee managed by Manager*, will mean that the Employee class will contain a Manager object in it.

No actual code should be provided – just the headers and comments for all the methods/attributes that you believe will be required within each class. This JavaDoc is not a contract (not “carved in stone"), so don't worry if you make a few changes to the classes in the future.

Using JavaDoc, show what each of your Java classes will look like. Each class should start on a separate page, be shown as a JavaDoc file, and should have the following information:

- The name of the class
- An overall description of the purpose of the class
- A description of each of the attributes, constructors, accessor methods, mutator methods, and private helper methods within the class. The methods should consist of the method header (the name, return value, and parameters, if any), and a descriptive comment describing each method.

The classes should be listed in alphabetical order, for ease of marking. To hand them in, view each rendered JavaDoc class in a browser and then print each web page to a PDF file (remove the headers and footers that your browser adds). You can then use a variety of tools that exist to merge the JavaDoc PDFs with your assignment report PDF.

The following example is a subset of a class description for a Person class from a hypothetical high-level design of a payroll system. Note that this example is included to illustrate the format of what your JavaDoc would look like in your Java classes. You will, however, be handing in the rendered JavaDoc – not the raw code – in your PDF file. Furthermore, the comments below do not necessarily constitute an endorsement on the size of your descriptions. For example, more detailed descriptions may be needed for complex classes and methods.

Note: do **not** submit rendered JavaDoc files (e.g. `.html`, `.css`, etc.) in your project repository. Use your `.gitignore` file to ignore them.

```java
/**
* Person is the class that will be used to create a person in our payroll system
*
* @author Laura Reid
* @version Version 1.1
*/
public class Person
{
 /************************************************************
  * Instance Variables
  ************************************************************/

  // The person's first name
  private String firstName;

  // The person's last name
  private String lastName;

  // The person's social insurance number
  private String socialInsuranceNumber;

  // The person's current yearly salary
  private double salary;

 /**
  * Constructor.
  * @param fn First name
  * @param ln Last name
  * @param ssn Social insurance number
  */
  public Person(String fn, String ln, String ssn)
  {
  }

 /************************************************************
  * Accessor Methods
  ************************************************************/

 /**
  * Gets the person's first name
  * @return the first name of the person
  */
  public String getFirstName()
  {
  }

 /************************************************************
  * Mutator Methods
  ************************************************************/

  ...

 /************************************************************
  * Helper Methods
  ************************************************************/

 /**
  * Joins the person's first and last names
  * @return the person's full name
  */
  private String getFullName()
  {
  }
}
```

2.4 Project Plan

This section should be an update of the project plan from the first report. In particular, your group should provide a list of tasks to be accomplished between milestones 2 and 3.

Indicate all the information for each task that was required in team assignment 1. Remember to update the percentage of completion of any previous tasks and any upcoming tasks that you have already started. Make sure your Gantt chart correctly indicates who is/was assigned each task, and who actually completed each task, so that we know that the work is being fairly distributed.

3 Prototype

For this assignment, your team must choose and implement six (6) of its user stories. The stories chosen should be listed in the **Revised User Stories** section of your PDF report, as discussed in section 2.2. Feel free to implement more than six user stories – you will thank yourself later at the end of the course as the deadline nears.

The TA must be able to compile the prototype by cloning your repository and using the `mvn package` command. Additionally, the JAR produced should be named `teamN-gradebook.jar`, where `N` is your team number. The TA should be able to run this JAR by typing `java -jar teamN-gradebook.jar` (that is, it must be an executable JAR and must contain any and all dependencies within it).

Note: the TA must be able to see that when he/she starts your program, enters some data, exits your program, and then restarts it, that the data he/she entered is persistent (i.e. it is not lost when the system is shut down). Not all the data must be stored for this assignment, but the TA must be able to see that your team has figured out how to perform data persistence for the system.

3.1 Unit Tests

We will not be marking your unit tests until the end of the course. However, it is **strongly recommended** that you write your unit tests for your code as you complete it (or *before* you complete it, if you're doing test-driven development). You don't want to be in a situation at the end of the course where you're scrambling to write all your unit tests.

4 Marking

The marks will be tentatively assigned as follows:

- Title page and general formatting of hand-in: 2 marks
- Revised UML class diagram: 3 marks
- Revised user stories: 5 marks
- Class descriptions: 10 marks
- Revised project plan: 3 marks
- Prototype: 20 marks
- Spelling and grammar: 2 marks

5 Assignment Submission

- In the **root** of your team repository, create a directory `deliverables/asn2`
- In the `deliverables/asn2` directory, place **one** file named `teamN-asn2.pdf`, where `N` is your team number
- In the root of your repository, there should be a `pom.xml` file
- Your code should be present in the `src` directory, which should be found in the root of your repository
- Do **not** submit rendered JavaDoc files (e.g. `.html`, `.css`, etc.) in your project repository. Use your `.gitignore` file to ignore them.
- Commit your code and PDF to GitHub and tag the commit with the tag `asn2`
  - Do not forget to push the tag to GitHub – your assignment is not submitted unless the tag is pushed
  - You can verify that the assignment was submitted by clicking the **Releases** link on your team repository page on GitHub. You should see an `asn2` release if you submitted successfully

- Do not submit any other files in the `deliverables/asn2` directory
- Remember that part of your mark is based on the clarity and organization of your report, so make sure it is neat and well organized
- Additionally, remember that poor spelling/grammar will lose you marks, so ensure that one or two people go over the entire assignment once it is finished, prior to submission, to proofread it and correct any errors found

6 Peer Evaluation (to be done individually)

Within four days immediately after this assignment's due date, complete a peer evaluation for each individual (including yourself) on your team. The private course site will soon have a link to complete the peer evaluations.

For each peer evaluation that you fail to submit within four days of the due date, you will individually lose 0.5% off of your final course mark (up to a total of 2.5% for the 5 peer evaluations you must submit).

Note: Peer evaluations can adversely affect the final mark given to an individual, so please complete these peer evaluations constructively and fairly.