Assignment One (A1): Object-Oriented Software Design (Group Work)

Liam O'Reilly

Number of Credits: 30% of the 15 credit module
Electronic Submission Deadline:
Monday, Nov. 6th @ 11am.

Learning Outcome(s): To gain experience in designing larger scale software.

1 Problem Statement and Overview

In this assignment, each group will complete an **object oriented design** of an application that has eBay-like functionality specialising in artwork called Artatawe (pronounced Art-A-Tawe). The electronic version of this system shall adhere to the full description of the functional requirements. A detailed list of the functionality and requirements specifications is given in this document (Functional Specification). *Hold on to this description. You will need it for Assignment A3. For those of you on the Software Engineering course, you will need it for assignments in CS-235 as well!*

1.1 User Information

Each user of Artatawe must have an account, which we refer to as the user's profile. The following information is linked to a user's profile:

1. A username that allows access to the profile.
2. A first name and a last name.
3. A valid UK mobile telephone number.
4. A full UK address complete with postcode.
5. The date and time of the last time the user was logged in.
6. A profile image.

You should design and implement classes to manage this information. Also, you should create an graphical user interface that allows the input of this information to create a new account.

1.2 The Artatawe System

The main purpose of Artatawe is to allow users to auction artwork to other users. Such a system would normally run as a web application, but to make this assignment manageable (at least for the initial version), Artatawe runs as a stand alone PC application. Only one user can be logged in at a time. All data will be stored in files on the local computer.

Artworks to be auctioned on Artatawe fall into two categories, namely, paintings and sculptures. Each artwork (that is placed for auction on the systems) has:

- Title of artwork.
- Description of artwork (optional)
- A main photo of the artwork.
- The name of the creator.
- The year the art was created.
- Reserve price.
- Number of bids allowed.
- The date and time the artwork was placed on auction/entered into the system.

Each painting also has:

- Dimensions (width, height).

Each sculpture also has

- Dimensions (width, height, depth)
- Main material (e.g., marble).
- Additional photos of the sculpture (optional).

Unlike eBay this auction system is not based on time, but based on a set number of bids (see Section 1.4).

When a user places an artwork up for auction they must provide all the data above depending on the type of artwork (with the exception of optional data). Other users will be able to login to the system and browse artworks

that are currently up for auction. A user will be able to bid on a piece of artwork (see Section 1.4).

It is up to you to design the data structures involved in realising this system.

In your design and implementation, you must have files that store the artwork entries, the bid histories and all the relevant data used in the system. For images, you can store file paths to local image files.

1.3 Browsing Artworks

Users shall be able to browse the artworks. That is, the user shall be able to look through the images of the artworks and have the title and description (if available) displayed. The user shall be able to choose an artwork and the system will provide more detailed information. From here users shall be able to mark the seller as a favourite (see Section 1.5) and also bid on artworks.

The user shall be able to filter artworks based on the type of artwork to only show paintings, sculptures, or both (i.e., disable the filter). The user shall also be able to view artworks the users they have marked as favourites.

Users shall also be able to search for artworks based on entering partial information in a typical search mechanism.

1.4 Bidding

Unlike eBay this auction system is not based on time, but based on a set number of bids.

When a user is viewing an artwork, they may place a bid. They will enter the amount of the bid and the following actions will take place.

- If the user placing the bid is already the highest bidder then the bid is refused and the user is informed of this.
- If the bid is lower than the artwork's reserve price then the bid is refused and the user is informed of this.
- If the bid is lower or equal to the current highest bidder then the bid is refused and the user is informed of this.
- If the new bid is higher than the current highest bid and also higher than the reserve price then that bid is accepted. The bid becomes the new highest bid on that artwork. The artwork's *number of bids allowed* is then decreased by one. If this number reaches is then zero, the auctioning of this particular artwork is complete and the user who just bid (and is the highest bidder) wins the auction. The user is informed of the outcome of the action (either the bid is accepted; or the bid is accepted and they win the auction).

1.5 Favourite Users

Users shall be able to browse the limited profiles of other users (username and profile image only) and mark them at favourites. Users shall also be able to unmark a user as a favourite. When browsing the artworks on auction and looking at a particular artwork, the user shall also have the ability to mark the seller as a favourite.

1.6 Bid Histories

A user will be able to see two sets of bid histories.

Firstly, as a buyer, they will be able to see a list of bids (in chronological order) they have placed (and the artworks associated with those bids). This will be an empty list if they have never placed any bids. Each entry will show the date and time of the bid, the artwork that was bid upon and the amount.

Secondly, as a seller, for each artwork a user has placed on the system for auctioning, they will be able to see a list of bids (in chronological order) placed on the artwork by other users. Each entry will show the date and time of the bid, the username of the person who placed the bid and the amount.

1.7 Won Artworks

A user will be able to see a list of artworks for which they placed the winning bid.

1.8 Completed Auctions

A seller will be able to see a list of artworks which have successfully been auctioned. This will display the artwork name, the winning bidder's username and the winning bid.

1.9 Information Since Last Login

The user will have the ability to look at a variety of lists which will contain information that has changed since the last time they logged in. Specifically:

- New artworks that are now on auction that were not on auction last time the user logged in.

- A list of artworks which the user is selling that have had new bids on them since the last time the user logged in.

1.10 Profile Images

Each user must have a profile image. Two kinds of profile images are supported:

- Avatars
- Custom Drawings

The system will have a set (say 5 or 6) built in Avatar images. The user can select one of these as their profile picture.

Alternatively, the user will be allowed to create a custom drawing. The system will provide the user with a built-in basic drawing environment that is similar to a paint program with at least the following functionality:

- Ability to draw straight lines
- Ability to draw a particle trace (i.e., clicking and dragging will draw "filled" circles at the location of the pointer).

Once you have the basic functionality working to a high standard, you are welcome to expand upon it to earn higher marks.

1.11 Other Properties of Artatawe

Initially, all the information shall be stored in files on disk. More specifically, the Artatawe must have the following property:

- For all users, their profile information, artworks they are auctioning, bid histories and favourite user list must be saved to disk locally. When you restart the program, these details will be loaded from a series of files.

The first version of Artatawe that you create should operate on a single machine. A user logs in to the program and performs actions which are saved locally to disk on that machine. The user logs out. A new user can then log in to view what has changed. When you start the implementation part of the project, please do not proceed to something more complicated until you have all Artatawe functionality working well locally on one machine.

1.12 Extra Artatawe Features

This section is not technically part of A1, but it will be part of A3. It will be helpful for you to plan for extensibility and hence know about this section.

To achieve top marks (in A3), I would strongly encourage you to be creative. At a minimum, all the functionality of the specification should be completed to a high standard. All features should adhere strictly to the specification. You need to get all this working very well in order to get a low first class mark (in A3). In order to get higher marks, you would be required extend the implementation in a novel way. All extensions that do not violate the specification will be considered. Substantial extensions to the software, extra reading and learning, will be required to achieve a high first class mark (in A3).

For this assignment (A1, object-oriented design), do not include these extra features. Simply design without the extra features, but be aware that they will be required for the implementation (in A3).

2 A1 Tasks

Your team is asked to provide a complete **design** for the entire system. This design must include at least one complete class hierarchy. Each team member is required to contribute to at least one class in the design. The designer of the class does not necessarily need to implement it in A3.

2.1 Submission Requirements

Only electronic submission via Blackboard will be accepted. Please submit your files in the following formats:

- Design Document (PDF)
- Meeting Minutes (txt or PDF)
- Contributions Report (PDF)

All group members must review the document prior to submission. Each individual group member is responsible for understanding the contents of the entire document. Submission indicates that all group members have read and approved the document unless a conversation that has been documented by your tutor indicates otherwise.

2.2 Design Document

The Design Document (report) proposes an object-oriented design for the *entire* application. In other words, your team incorporates the full functional specification into the design. The Application Design Report

is no more than 30 pages including text and diagrams and consists of the sections as detailed below:

2.2.1 Candidate Classes and Responsibilities:

Provide a list of candidate classes and their responsibilities. This list is like the CRC cards in lectures but with a bit more detail. For each candidate class the team has identified, the following information is provided:

1. **Class Name:** (in bold)
2. Author:
3. SuperClass:
4. SubClasses:
5. Responsibilities: a list of services this class provides.
6. Collaborations: a list of classes with which the class communicates.

There should be one of these cards for every class that appears in your class diagrams (see Section 2.2.2).

2.2.2 Class Diagrams

After specifying the information in Section 2.2.1, draw the classes using UML Class Diagram notation. Your class diagrams must use appropriate arrows and UML syntax to represent the relationships such as collaborations (i.e., associations, compositions, and aggregations) and generalisations/specialisations (i.e., inheritance). The drawing style is to be modelled after the UML drawing style used in lectures. You should carefully think about navigability and multiplicities when drawing the collaboration relationships (associations, compositions and aggregations). If you would like, you can use UML tools such as Papyrus[1] (or others).

When providing details about the attributes and operations, you must think carefully about type information and your design. The classes and methods should fit together and function to achieve the intended behaviour.

Each class in your design (See Section 2.2.1) must appear in at least one UML Class Diagram. Each class hierarchy identified during your design process must be depicted in a hierarchy in a UML Class diagram.

2.2.3 Hierarchy Descriptions

Each generalisations/specialisations (i.e., inheritance) relationship (See Section 2.2.2) must be accompanied by a short textual description that describes the "is-kind-of" relationships used. Make sure that your team provides a justification that backs up its choices. Make sure that abstract classes are distinguishable from concrete classes in your diagrams. Your team should be able to identify two (or more) class hierarchies.

2.2.4 Collaboration Descriptions

Each composition and aggregation relationship (See Section 2.2.2) must be accompanied by a short textual description that describes the "is-made-up-of" relationship used. Make sure that your team provides a justification that backs up its choices.

2.2.5 Subsystems Diagrams and Descriptions

Each subsystem identified during your design process should be depicted in a diagram. Again, the drawing style is to be modelled after the drawing style used in the lectures. Each diagram is to be accompanied by a short textual description that describes the subsystem and justifies how the classes work closely together. Again, make sure that your team provides a justification for each subsystem, in other words, why do you think a certain set of classes or objects belongs together in a subsystem. Your team should be able to identify two (or more) subsystems.

2.3 A1 Submission: Design Report and Contributions Report

Electronic copies of each file and PDF report are to be submitted to Blackboard. One member of each group, the **Planning and Quality Manager**, should lead the submission of all files and reports on behalf of the group, such that they are all in one place (and not scattered amongst several group members). Points will be deducted for those submissions that do not follow the file naming conventions and required file formats.

2.3.1 Design Document

85% of A1 (25% presentation, 60% content) Your Design Document is to be included in your submission to Blackboard. The file must be named *"XXGroupGN-DesignReport.pdf"* where XX is replaced by either "CS" or "SE" depending on if you are a Computer Science Group or a Software Engineering Group and GN is replaced by your group number. PDF format is required. Word document format (.doc or .docx) is not acceptable. Attention: you will be assessed on your conformity to the instructions.

[1] `http://www.eclipse.org/papyrus/`

2.3.2 Minutes Protocol

10% of A1 A copy of your group minutes for three (or more) meetings held before the assignment deadline must be included in your submission to Blackboard. Your minutes files must be names *"XXGroupGNMinutes-yyyy-mm-dd.txt"* where XX is replaced by either "CS" or "SE" depending on if you are a Computer Science Group or a Software Engineering Group, GN is replaced by your group number, $yyyy - mm - dd$ is replaced by the calendar date your group met (with $yyyy$ replaced by the year, mm with the month, and dd by the day), e.g., $Group3minutes - 2017 - 10 - 23.txt$. (You will be assessed on this.) A **minimum of three** minutes of meeting protocol is required and are submitted on behalf of the group by one of its members. The minutes format should follow *Bob's Minutes of Meeting Protocol*. See blackboard for a copy of this document.

2.3.3 Contributions Report

5% of A1 A Contribution Report is to be included in your submission to Blackboard. This document contains a description of what and how each group member contributed to the project design is required. The file must be named *"XXGroupGNMemberContributions.pdf"* where XX is replaced by either "CS" or "SE" depending on if you are a Computer Science Group or a Software Engineering Group and GN is replaced by your group number.

Each group member is obliged to contribute at least one class, both design and implementation, to assignments 2 and 3. The intention is that the classes each student chooses to design are also the classes they implement. However, students can change between assignments if necessary.

The Contribution Report, no more than 5 pages, describes who contributed to classes, hierarchies, subsystems in the design, and other contributions, e.g., the minutes etc. The Contribution Report is also written collectively. In other words, each group member describes their respective contribution to the project.

The report also also informs the reader if any unexpected problems arose during the course of the assignment. For example, if a group member skipped too many lectures, and as a result didn't have the background necessary to contribute, this should be stated in the group report. Likewise, the group experienced success in some areas, both expected and unexpected, this is included.

3 Issues with Contribution

The group assignments assess the performance of the group when delivering a non-trivial piece of software. As mentioned in the course, real software is developed in teams and therefore group work is a necessary skill that needs to be developed. Individual contribution levels will mainly be assessed at the end of term when the groups are interviewed and each member has the opportunity to demonstrate their understanding of the course material in reference to the project. Under normal circumstances, this interview shall be used to adjust marks and determine the level of contribution. However, we realise that abnormal circumstances during group work may occur.

In the event of any non-contribution, please follow this procedure:

- As early as possible, discuss the situation with your tutor.
- Work with your tutor to try and get the non-contributor participating.
- Keep your tutor posted on the situation.

In the case of significant non-contribution despite these efforts, a confidential non-contribution report can be filed to reduce the marks of non-contributors. This report will only be used to lower the marks of non-contributors. The report will not be used to raise the marks of any group member. The report should contain an explanation of the situation and the members involved. This report will only be accepted if the tutor is aware of the situation and the group has taken all possible and reasonable steps to get the group member contributing.

This document must be sent in a confidential email to the instructor (Dr. Liam O'Reilly, L.P.OReilly@swansea.ac.uk).

Likewise, if any particular group member disruptively dominates group meetings and the assignment outputs in a way that is damaging to the team, the tutor should be notified immediately. Action on the marks of group members that are overly disruptive in this way. Please be supportive of all group members.

4 Project Hints

1. All group members are expected to contribute to all phases of the development including design and implementation. The author fields in both the

class design and implementation reflect individual group member contributions.

2. Ask questions in your tutorial.
3. Do not email the instructor with questions, use the Blackboard forum.
4. You could use Slack for communication as a group. Setup Slack by going to `https://slack.com`. Many software development teams and companies use Slack as a main form of communication.
5. You could Skype for some of your group meetings?

5 A1: In a Broader Context

The software design and implementation is broken down into three major phases. A1 is the first assignment of CS-230. A3 will involve a partial implementation. In CS-235, you will continue working on this software. You should contact the lecturer next term for information and clarifications on A4 - A6. CS-235 is also **100%** coursework based.

- **Assignment 1, Design:** *(i.e., this assignment)* Each group proposes an object-oriented design for the *entire* application.
- **Assignment 3, Phase I Implementation (CS-230):** Each group implements a sub-set of the features which is specified in Assignment 3. The features are a subset of the feature specification. After assignment A3 is submitted, All teams will be interviews in the last week of term to determine individual contribution and understanding of the material.
- **Assignment 4, Phase II Hand-Off and Assessment (CS-235):** Each group then hands their full design and partial implementation to another group. Correspondingly, each group takes over another groups' design and partial implementation. Each group then writes a short report assessing the project work they have taken over with respect to the quality of the design, implementation, and testing.
- **Assignment 5, Phase II Implementation (CS-235):** Each group implements V2.0 of the system, i.e., the entire set of features described in the specification.
- **Assignment 6, Re-Engineering (CS-235):** Assignment 6 is a re-engineering task where each individual implements features which are not yet defined. The full description of assignments 4, 5, and 6 will be provided in future documents.