Contribution report – Group 32

Bradley Tenuta - 902559

My job for implementing Artatawe was to implement the Artatawe class, which creates all the GUI for the program except the GUI made for custom drawings. I created the fileWriter class, which saved all the data to a text file and I created the CSS for the program in order to style it and make it look nice and easier to read. Furthermore, I also provided help and assistance for my group members who ran into any difficulty.

Mainly I created nice GUI, which displayed all the data based on the requirements in the way that was wanted. For example the information on all the bids a user has placed had to be in chronological order based on date so the most recent bid was at the top of the table. I also implemented most of the error checking for logging into the system and when you create a bid. I had to make sure there was checks for if the user entered in characters rather than numbers and that the number were higher than any previous bid and so on, for each error I have a different error message stored in a label and tells the user what they did wrong. I also had to save the data so I made a method call which saves data every time the user logs out and every time the users pressed the red x that closes the window and the program.

Extra features that I was able to add to the program was the ability to log in and out of the system and so it didn’t require the user to quit the program to change user. Another was the ability for the GUI to auto update when changes were made, for example when making a bid it automatically adds it to the table and updates the amount of bids made on an auction without having to reload the page.

Problems that I came across when making the Artatawe class and the fileWriter class was that my first implementation of tables on the auction page and the account page wasn’t working the way I wanted it to. I first used a tableView, which displayed data from a certain class; however, I realised shortly after that I wanted to view data from more than one class and so I couldn’t use table view. In the end, I made my own table using HBox’s and VBox’s and styled it in CSS for the table to look nice. Another issue I came into was that In order for the fileWriter and FileLoader to work well together they have to save and load in the same format. Therefore, a lot of communication was required in order to get it working and in the end, we just let one of us complete their class first and then the other person just used the format they used so the classes would work and communicate together. If we had to do it, again we probably would have assigned the file writer and file loader class to the same person to avoid this confusion.

Elias Nemr – 961625

My job for implementing in CS230, A3, Artatawe project was to create the profile picture system/GUI. I had two requirements:

1. Default avatars for the users.
2. Custom Drawing with particle trace of circles and drawing straight lines.

To begin with, I started with a main GUI which the user could choose to change their profile picture on their “My Account” GUI. After opening the main Profile Picture GUI, you are allowed to choose between picking an avatar or creating your own custom drawing. If you click on the choose avatar button, you will get another scene which will display for you 6 default pictures (including the default one you get by creating your account) which then you will choose to set as profile. Otherwise if you choose to create a create a custom drawing you will have to go back to the Profile Picture GUI and choose custom drawing, which you’d be able to draw anything you like and after having saved that picture, it will automatically override your default avatar which you chose prior. It should do the same vice versa.

Furthermore, I also had to capture the features of our system once it was finished and I had a volunteer from my team, Cormac, help out with the voice over and the rest of the team collaborated with how we should layout the demo video. After we recorded all the features I had to edit out the videos and add subtitles to help understand better.

Some issues I came across while creating my GUI part of A3 was the saving and loading part of custom drawing. It worked just fine when we used the command line to run the program. However, when trying to use an IDE it would crash sometimes as it would not allow saving onto the “src” folder. I also had to do some extra research to figure out how to particle trace circularly rather than rectangularly.

Liam Cooper – 916153

My job for the implementation of assignment 3 was to implement a hierarchy which is used to create the artworks which are placed on auction by users. For this I created 3 classes, the first class was the artworks class which contained basic data that both types of artworks contained so this meant that the other classes (painting and sculpture) had to inherit from this class. I also decided that the users should not be able to create an artwork object as it doesn’t make sense in our application so this class was made into an abstract class. The next class I created was the sculpture class which contained the specific sculpture data and inherited from the artworks class. The final class I created was the painting class which contained specific data for the paintings and also inherited from artwork.

These classes are a main part of this application as it’s an artwork auction application, there are a lot of classes that collaborate with mine. This meant that I had to create these classes early on during the implementation stage to allow my fellow team mates to finish off their classes as they could not be fully implemented or tested without mine.

The first problem I had encountered was the data and time variable which contained the date and time the artworks were placed on the auction. I was initially setting this within the artworks class by importing the date library, however this was not required as this was a responsibility of another class. So, I changed this variable to a string as the date was imported and created in another class and then a string of the date and time was passed in when a painting or sculpture is created.

Another problem was the confusion we had of where to store the maximum number of bids that each artwork allows when it is on auction. This was not included in my initial design of the classes so we decided that it would be best if a variable of bid amounts was added to the artwork class during the implementation. This required me to create the variable in order to set and get the maximum bid amount.

Jay Markey - 828820

My contribution to the implementation of A3 was the FileLoader class. This was the class that loaded the information from two text files. One which held information about the users, the other contained information on artworks. The class contained four methods, this doesn’t comply with the class design I created in A1 this was mainly due to me not entirely thinking about how I would be creating the objects. There were also two more variables that I added once implementation had started, these were both array lists used to store profile and auction objects.

The method LoadProfile() was used to create profile objects and returned an array list of these objects. This method was the most straightforward to implement. The layout for the profiles was simple to load. Once this had been finished and id moved on we realised that we hadn’t provided a way of showing the users favourites. I had to go back to this method to make that happen. It took a little while longer than anticipated to implement a work around for adding favourites without mistakes being made when creating objects.

The loadAuction() method was originally meant to be the only other method in this class but when it came to making it I realised that it would be easier to have this method called from another class and then it would decide what method to call either loadPainting() or loadSculpture() depending on the type of artwork that needed to be added to an auction. A problem that I encountered when testing this method was that for some reason the if else statement didn’t like when I had another conditional in the else statement. I thought about this and realised as it would only be us writing to file then there was no need for the condition in the if else. This method returned an array list of auction objects.

The loadPainting() and loadSculpture() methods are pretty much the same they just created the different artwork objects, then created an auction object using the artworks and finally made and added bibs to the arraylist of bids. Originally the file used to load these objects was set out in the same format as the file used to store the profile information. However, when trying to distinguish between what objects were to be created I found that it was much simpler to have all of the information about an auction on one line and use the first element of the line as a guide for creating what objects. It took a fair bit of communication between myself and Bradley to decide on a layout that was good for the both of us.