Critical Reflection

The design we were given for the system had many significant floors particularly in consistency of ideas and in security protocol. The main security issue was that when a user logged in to the token machine, their entire user account was passed through and stored locally in the machine. The presented a problem which required a large-scale redesign of the account variables and methods. For this reason, a Proxy design pattern was used to hide this, and only the account ID was returned; thus, allowing a more secure account system. There was then a function created which allowed specific data to be accessed from the account by passing through the accountID integer and a string which tells the function what it needs to return. This subsequently meant that some other functions were not used such as GetCardId as they were made obsolete by the new template function:

(GetXByAccountId from the CustomerAccount class)

public T GetXByAccountId<T>(int accountId, string x)

{

var accs = ReadFromBinaryFile<List<CustomerAccount>>(@"Accounts.txt");

           foreach (var account in accs)

           {

               if (accountId == account.\_accountId)

               {

                       switch(x){

                           case "balance":

                               return (T)Convert.ChangeType(account.\_balance, typeof(T));

                           case "cardid":

                               return (T)Convert.ChangeType(account.\_cardId, typeof(T));

                           case "fullname":

                               return (T)Convert.ChangeType(account.\_fullName, typeof(T));

                           case "username":

                               return (T)Convert.ChangeType(account.\_username, typeof(T));

                           case "paymentoptions":

                               return (T) Convert.ChangeType(account.GetSavedPaymentDigits(), typeof(T));

                           case "savedpaymentmethods":

                               return (T)Convert.ChangeType(\_savedPaymentMethods, typeof(T));

                       }

               }

           }

           return default(T);

}

Many of the classes within the class diagram were lacking basic variables which were required for the class to function correctly. Some examples of this are classes like LanguageList which needed to contain a List<Language> variable, and the scanner location becoming a Station rather than just a string. This allows the scanner to have access to a list of tickets which are valid which is crucial to the scanner’s ability to open and close the barrier that it is attached to.

Another issue with the design was the way that the payment system had been implemented. This process had been detailed inside one of the sequence diagrams.

Maybe talk about problems with university PCs not having SVN originally?

Talk about times where all of us was working on one piece / (slight waste of time as other people could have been doing other things)?

Should have focused on core functionality first then built around it. Eg. Ticket system should have been implemented weeks ago; rather than last minute as it affected a lot of things.