**COE 528 – Final Project Report**

* **Briefly Explain the UML Use case diagram in one paragraph**
* The UML Use Case Diagram depicts 2 actors: **a customer** and **a manager**. The manager must initially set up an account for the customer by logging in (with username/password as admin) and then creating an account. The customer can then log in and do 4 main actions – **purchase** an item (over $50, along with a transaction fee based on the level), **deposit** or **withdraw** some money and check the account **balance**. While making a purchase or withdrawing money, there will always be a check to see if there are sufficient funds in the account, before performing that action. From the managers perspective, after logging in (note: error checks are done to make sure that the **right combination** of the username and password are entered) the manager can **add** or **delete** customers. **“Extend” relationships** can be seen as methods that are necessary to test edge cases but are **not absolutely crucial** in the main working of the program, while **“include” relationships** **are necessary** for the program to perform its function.
* **Briefly Explain the UML Class diagram in one paragraph**
* The UML Class diagram has 6 main classes. The first 2 classes are the customer and the manager, which are related by an “**association**” relationship. The manager has full control on creating and deleting a customer, thereby completely relating directly with a customer. The customer class has an “**aggregation**” relationship with the account class which means that the variables are shared between the two classes. This can be seen in the customer class, where an Account object is initialized. The **abstract** **state** class is also related to the customer by a composition or “**has-a**” relationship. This means that when each customer account is created, the customer always has a given state. The states can be split into 3**(Silver, Gold and Platinum**). These 3 states inherit (**is-a relationship**) from the abstract state class and allow the user to make purchases with different transactional fees, based on the balance they have remaining on their account.
* **Which class performs requirement #2?**
* **“customer.java”** is the selected file where the Abstraction Function, Rep Invariant and other necessary clauses are stated. The method also contains the toString () and repOk() method.
* **Refer to which parts of the UML Class Diagram form the State design pattern**

State design pattern can be found when the 3 different states (Silver, Gold and Platinum) all inherit from an abstract class state. The states have only 2 methods (makePurchases and validate). The validate method ensures that the customer is in the correct state based on the account balance. This method also helps while making purchases to ensure that the correct level is obtained before a transaction fee is applied on a specific purchase (Example: no fee if you are a platinum member, but a $10 fee if you are a gold member).

* **What major books/papers were used as external sources?**
* No books/papers were used as major references. The main websites that came in handy during the project were tutorialspoint/docs.oracle.com to find java functions and import the right classes, stack overflow to fix errors and geeksforgeeks to understand some fundamental programming concepts.