Assignment 5 - Design and UML

You may work on this assignment in **groups of 2**, or you may work alone. If you work as a group, both members will receive the same mark. You should indicate your partner (or if you have none) in your README.txt file. Once you have found your partner, enroll yourself into a group together on d2l. Do not enroll yourself into the same group as another student without their permission.

Refer to chapters 8 & 12 of the text, Appendix H of the text, and the lecture notes to guide you.

You are to design a module that simulates a Vending Machine. You are not to write the implementation, only do the design.

- The vending machine sells some products. Each product has a name and a price.
- The vending machine has a fixed number of products, and a fixed quantity of those products.
- The vending machine also has a cash box that stores coins.
    - The machine can hold at most 500 coins, though it has a little extra room to complete a final sale.
- A customer may purchase an item from the vending machine.
    - A customer selects a product for purchase
    - If the product is sold out, the machine displays a message on its display, and the purchase is aborted.
    - If it is not sold out, the machine displays the amount due to complete the purchase
    - The customer then adds (individual) coins
    - The machine holds onto the coins until the transaction is completed, and it updates the amount due on the display for each coin entered
    - When sufficient money is entered:
        - The machine vends the appropriate product (the machine does not give change), and displays "*** VEND ***" on its display
        - The coins are then placed in the machine's cash box.
        - The machine always successfully vends. You do not have to model any failure conditions.
    - OR
        - The user may abort the transaction and get all of their coins back. They must be the exact same coins that the user inserted.
- If / when the capacity of the cash box is reached,
    - the machine goes out-of-order and will not accept any more purchases.
    - It also sends a message to its operator that it needs its cash box emptied.
- If the machine is in a non-purchasing state, any coins inserted are immediately returned. Again, these must be the exact same coins that were entered.
- If the machine ever runs out of stock for any item, it sends a message to its operator that it needs that item restocked.
- If the machine completely runs out of all stock, it enters and out-of-order state and will not accept any more purchases.
- Whenever the machine is out of order, it displays an appropriate message on its display.
- At any time, an operator may come to empty the cash box and/or restock any items. The operator has a secret key that allows them to do this. Anyone with the key may perform these operations, but no one else can.
- Coins come in the following denominations: 5, 10, 25, 100, 200. The machines does not accept bills.

Do not use the console for any output (or input). Instead, create classes to model the different components of the Vending Machine, and give them appropriate methods to call.

Note that the machine has several distinct states and transitions between those states. Be sure to model them appropriately.

For this assignment, do the analysis and design phases. Do not do any implementation.

- Understand what you need to build.
- Create user stories for each use case. As a ______, I want to ______, so that _______. ("UserStories.txt")
- Create a UML use case diagram. ("UMLUseCases.png")
- Create a list of test cases for this simulation. ("TestCases.txt")
- Create CRC cards for each class that you are planning to design. ("CRCcards.txt") You may list the sections vertically, but be sure to clearly indicate them and separate CRC cards clearly.
- Create a UML class diagram showing the classes and their relationships in your design. This should include method names. ("UMLClassDiagram.png")
- Create a UML sequence diagram for the "successful purchase" use case. In a file called ("UMLSequencePurchase.png")
- Create a UML State diagram to model the state and transition between those states. ("UMLStateDiagram.png")
- Include a file (README.txt) which contains your name, student ID, and any notes that the marker may need to be aware of. You should not describe your design here because it should be self-evident from the rest of your submission.

You may use a tool such as yuml, gliffy, creately, or umlet to make your UML diagrams.

Archive all of your files into a zip file named "a06.zip" and submit it to your d2l dropbox.