ITEC 324 Fall 2021
Project 1: Vending Machine

Deadlines:

Note that this assignment has **two deadlines** as shown below:

Part1: Use Cases, CRC Cards, UML diagrams
[Due] 5:00pm on 09/20/2021

Part2: Program
[Due] 5:00pm on 09/27/2021

What to do:

Analyze, design, and implement a program that simulates a vending machine. The machine has two modes, one for customers, and one for an operator.

Customer Mode:

- Products can be purchased by inserting coins into the machine. The vending machine must take *one* quarter, one dime, one nickel, or one penny *each time*
- After having inserted coins into the machine, the user can select a product from a list of available products to purchase
  - If all goes well, the machine will deposit the selected product. If the user put more than enough money into the machine, the machine dispenses change as well
  - However, if any of the following cases occur, the product is not dispensed, the user should be notified of the issue, and the machine should handle the situation in some reasonable fashion. The exact details on how you want to handle this is up to you, but try and put yourself into the shoes of a customer
    - The item is no longer in stock
    - The user didn't put enough money into the machine to purchase the selected item
    - The machine doesn't contain enough coins to make valid change
- A user should be able to get back the unused money they have inserted into the machine
- There should be some way for the user to exit the program

Operator Mode:

- In order to enter operator mode, the user must supply a password
- An operator can remove coins from the machine
- An operator can restock items
- An operator can change the price of an item
- The operator should be allowed to log out (returning to customer mode)

The Vending Machine:

- Your vending machine must have at least 5 item types, where each item has a name,price, quantity in stock, etc.
    - You can hardcode the initial values for each item however you wish
    - You can assume that the items (except for the quantity and price of each item in stock) do not change throughout the program. There is no way to add or remove item *types*, although it is possible to change the *number* of each item in stock by purchasing items (user mode) / restocking items (operator mode). It is also possible for the operator to modify prices.
- The vending machine should start with exactly 10 pennies, 10 nickels, 10 dimes, and 10 quarters.
- The vending machine only has room for 10 counts of each item type.

Requirements for Part 1:

- You must include a CRC card, or equivalent representation, for each class in your program
- You must include a UML class diagram that includes every class in your program
- You should, at an absolute minimum, include two use cases. Each of these use cases must have at least one variation
    - However, you should think about ALL variations you might encounter as you design your program.
- You must include a state diagram with at least 2 states. Depending on your design, it might be very simple.
- You only need to include a sequence diagram for one of your use cases.

Additional notes:

- You can program your interface using text-only, with user prompts. If you do this, use System.out for output and System.in for input (via a Scanner, for example). Alternatively, you can use a GUI, but it is not required.

- As mentioned in the Syllabus, you must demonstrate your code **prior** to uploading it. I will ask you to run several test cases including invalid inputs. Make sure you test your code thoroughly before demonstrating it. You should be able to answer questions on your implementation. A full implementation with correct output but without being able to explain the code will result in 0 points.

What to Submit:

Part1:
Submit Use cases, CRC cards, UML diagrams (class diagram, sequence diagram, and state diagram) in D2L by the deadline (no demonstration required).

Note: To draw the UML diagrams, you may use a Violet UML editor, which can be downloaded from https://sourceforge.net/projects/violet/.

Note: For the CRC cards, you don't need to use an actual index card this time. Instead, you can draw a rectangle in your editor.

Alternatively, you can handwrite your use cases, CRC cards, and UML diagrams, and scan them into a PDF. However, if you do this, **your handwriting must be legible.** If the handwriting is not legible, I might assign no credit for the illegible portions. If you are concerned about this, the safe bet is to use editors via the computer.

Regardless, your final submission should include only PDF file(s).

Rubrics:
1. **[30]** Use Cases
2. **[20]** CRC Cards
3. **[30+10+10]** UML Diagrams (class diagram, sequence diagram, and state diagram)

Part2:
Submit your .java source code files in a zip file only after demonstration.

Rubrics (Only for those students who could explain their implementation):
1. **[25]** Customer purchase use cases, including receiving change
2. **[5]** Allowing user to retrieve unused change
3. **[5]** Allowing user to enter operator mode, that requires password
4. **[5]** Allowing operator to remove coins
5. **[10]** Allowing operator to restock items, and preventing items from going above 10
6. **[5]** Allowing operator to change item prices
7. **[5]** Correctly maintaining model (i.e., number of coins and items) when switching between user and operator modes
8. **[5]** Allowing user to exit, no runtime errors
9. **[10]** Use friendly application
10. **[10]** Javadoc comments, self-documentation, and style
11. **[15]** Object oriented design principles
12. **[10]** Extra credit (10 random conceptual questions from Object-Oriented Design Process)