WIA1002/WIB1002 Data Structure

Tutorial: ADTs

Question 1:

Consider the following problem:
A new candy machine is purchased for the cafeteria, but it is not working properly. The candy machine has four **dispensers** to hold and release **items** sold by the candy machine as well as a **cash register**. The machine sells four products—**candies**, **chips**, **gum**, and **cookies**—each stored in a separate dispenser. You have been asked to write a program for this candy machine so that it can be put into operation.

The program should do the following:

- *Show* the **customer** the different products sold by the **candy machine**.
- Let the **customer** *make* the selection.
- *Show* the **customer** the **cost of the item** selected.
- *Accept* the **money** from the **customer**.
- *Return* the **change**.
- *Release* the **item**, that is, *make* the sale.

You can see that the program you are about to write is supposed to deal with dispensers and cash registers. That is, the main objects are four dispensers and a cash register.

Because all the dispensers are of the same type, you need to create a class, say, Dispenser, to create the dispensers. Similarly, you need to create a class, say, CashRegister, to create a cash register. You will create the class CandyMachine containing the four dispensers, a cash register, and the application program.

Your tasks are to design ADTs to represent the three classes:

a. Identify the instance variables for each of the class (i.e. Dispenser, Cash Register, Candy Machine)
b. Identify the methods/operations for each of the class (i.e. Dispenser, Cash Register, Candy Machine)
c. Produce a UML class diagram to represent the three classes

Question 2:

A bid for installing an air conditioner consists of the name of the company, a description of the unit, the performance of the unit, the cost of the unit, and the cost of installation. Design an ADT that represents a single bid for installing an air conditioning unit. Write a Java interface named `BidInterface` to specify the following ADT operations by stating its purpose, precondition, postcondition, parameters using javadoc-style comments:

- Returns the name of the company making this bid.
- Returns the description of the air conditioner that this bid is for.
- Returns the capacity of this bid's AC in tons (1 ton = 12,000 BTU).
- Returns the seasonal efficiency of this bid's AC (SEER).
- Returns the cost of this bid's AC.
- Returns the cost of installing this bid's AC.
- Returns the yearly cost of operating this bid's AC.

Then design another ADT to represent a collection of bids. The second ADT should include methods to search for bids based on price and performance. Also note that a single company could make multiple bids, each with a different unit. Write a Java interface named `BidCollectionInterface` to specify the following ADT operations by stating its purpose, precondition, postcondition, parameters using javadoc-style comments:

- Adds a bid to this collection.
- Returns the bid in this collection with the best yearly cost.
- Returns the bid in this collection with the best initial cost. The initial cost will be defined as the unit cost plus the installation cost.
- Clears all of the items from this collection.
- Gets the number of items in this collection.
- Sees whether this collection is empty.