**WIA1002/WIB1002/WXES1117 Data Structures**

**Tutorial : ADTs**

**Question 1**

1. A new candy machine is bought for the cafeteria and a program is needed to make the machine function properly. The machine sells candies, chips, gum, and cookies. A Java application program is needed for the candy machine so that it can be put into operation.
2. The program should do the following :
   1. Show the customer the different products sold by the candy machine.
   2. Let the customer make the selection.
   3. Show the customer the cost of the item selected.
   4. Accept the money from the customer.
   5. Release the item.

Input: The item selection and the cost of the item

Output: The selected item

Your task is to design the components of the candy machine. Identify the necessary requirements (data and operations) and produce a UML diagram for this and specify the pseudocodes for the process of buying a candy.

**Answer:**

A candy machine has three main components:

* a built-in cash register,
* several dispensers to hold and release the products, and
* the candy machine itself.

Therefore, we need to define a class to implement:

* the cash register,
* the dispenser, and
* the candy machine

|  |
| --- |
| Cash Register |
| -cashOnHand: int |
| +CashRegister()  +CashRegister(int)  +currentBalance(): int  +acceptAmount(int): void |

|  |
| --- |
| Dispenser |
| -numberOfItems: int  -cost: int |
| +Dispenser()  +Dispenser(int, int)  +getCount(): int  +getProductCost(): int  +makeSale(): void |

When the program executes, it must do the following:

1. Show the different products sold by the candy machine.
2. Show how to select a particular product.
3. Show how to terminate the program.

This is translated into the following algorithm:

1. Show the selection to the customer.
2. Get the selection.
3. If the selection is valid and the dispenser corresponding to the selection is not empty, sell the product.

Divide this program into three methods—showSelection, sellProduct, and main.

**//Method 1: Show Selection**

In Pseudocode, the algorithm is:

1. Show the list of options for user to select
2. Let user enter the selection
3. Receive user input

**//Method 2: Sell Product**

In pseudocode, the algorithm for this method is:

1. If the dispenser is not empty
   1. Get the product cost.
   2. Set the variable coinsRequired to the price of the product.
   3. Set the variable coinsInserted to 0.
   4. While coinsRequired is greater than 0:
      1. Show and prompt the customer to enter the additional amount.
      2. Calculate the total amount entered by the customer.
      3. Determine the amount needed.
   5. Update the amount in the cash register.
   6. Sell the product—that is, decrement the number of items in the dispenser by 1.
   7. Display an appropriate message.
2. If the dispenser is empty, tell the user that this product is sold out.

**//Method 3: Main**

The algorithm for the method main follows:

1. Create the cash register—that is, create and initialize a CashRegister object.
2. Create four dispensers—that is, create and initialize four objects of type Dispenser. For example, the statement:

Dispenser candy = new Dispenser(100, 50);

creates a dispenser object, candy, to hold the candies. The number of items in the dispenser is 100, and the cost of an item is 50 cents.

1. Declare additional variables as necessary.
2. Show the selection; call the method showSelection.
3. Get the selection.
4. While not done (a selection of a value to exit the program):
   1. Sell the product; call the method sellProduct.
   2. Show the selection; call the method showSelection.
   3. Get the selection.

**Question 2**

Write the definition of a class, swimmingPool, to implement the properties of a swimming pool. Your class should have the instance variables to store the length (in feet), width (in feet), depth (in feet), the rate (in gallons per minute) at which the water is filling the pool, and the rate (in gallons per minute) at which the water is draining from the pool. Add appropriate constructors to initialize the instance variables. Also add member functions, to do the following: Determine the amount of water needed to fill an empty or partially filled pool; the time needed to completely or partially fill the pool, or empty the pool; add water or drain for a specific amount of time.

**Answer:**

|  |
| --- |
| Class SwimmingPool |
| -length(lfeet: int): int  -width(wfeet: int): int  -depth(dfeet: int): int  -waterFillingRate(gallons: int): int  -waterDrainingRate(gallons: int): int |
| +SwimmingPool()  +SwimmingPool(l, w, d : int): int  + amountPartialFilledPool(): int  + amountFullFilledPool(): int  + duration PartialDrainPool(): double  + duration fullDrainPool(): double  + duration addWaterTimer(time: double) : double  + duration drainWaterTimer(time: double): double |