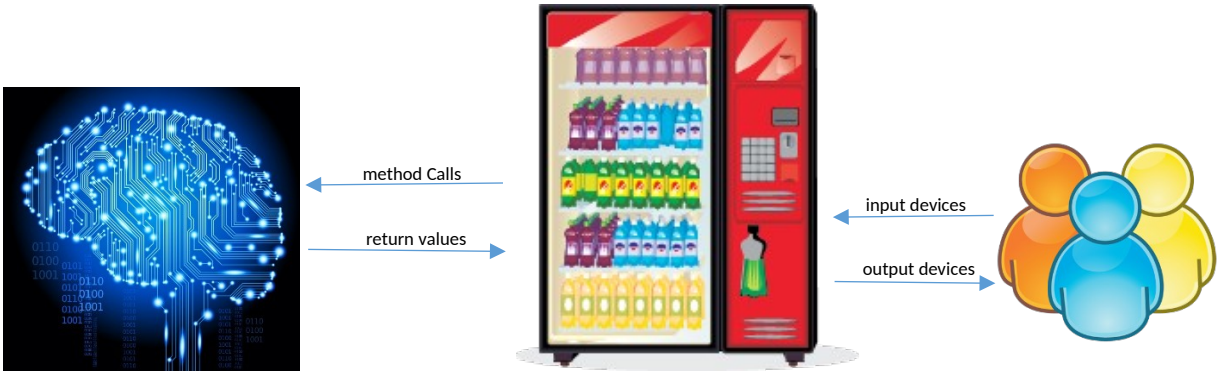
COMP10062: Assignment 2

The Assignment: Vending Machines in a Break Room

This assignment is about objects, instance variables, methods and encapsulation. You will create two classes to simulate a “break room” containing two vending machines.

In graphical applications, programmers often separate the **model** from the **view**. The model keeps track of the internal state of the program, and the view is in the middle between the user and the model. It talks to the user through a user interface, and it talks to the model by calling its methods and interpreting the return values from those methods. The model never talks directly to the user.

This is not a graphical application. In this assignment, the vending machine class is the model and the break room class implements a “view” that consists of a text-based conversation with the user. But if you implement the model well, it should be easy to re-use it later in a graphical view.



Step 1: Design and UML

Your first step should be to create a UML class diagram to represent a vending machine. You can create this diagram on paper or using UMLet or draw.io (links on Canvas). If you want to use a different piece of software, you must check with the instructor first.

We’re going to keep things very simple. A vending machine contains a single type of product. The product has a name and price and there is a limited quantity available. A vending machine tracks its current unused credit and the total amount of money it has taken in since it was switched on. The product name, price, and quantity can be set when the vending machine object is created, or the object can be created with default values for some or all these attributes.

A user can put loonies, toonies, quarters, nickels, and dimes into the machine. Coins must be entered one at a time. The machine does not accept bills or pennies. The user presses a button to vend the product and will expect a response (either something vends or it doesn’t).

Vending should be denied if there is not enough credit in the machine, or if there is no product left. It is up to you how to handle the user’s change. The machine might automatically return it when the vend button is pressed, or it might just deduct the price and leave it as credit. Either way, there should also be a coin return button that will return all the current credit to the user.

When the machine is first turned on, it is initialized with the name, price, and quantity of the product it contains. There are no restrictions on how the user can use the machine. They can insert their money before they press buttons, after they press buttons, or while pressing buttons.

There should be a **toString** function that returns a full report, something like this: "VendingMachine: 8 chocolate bars left, $1.99 each, $4.50 credit, balance $45.25".

IMPORTANT BASIC RULE #1: The vending machine is the **model** (i.e. the "brain" of an actual vending machine). The model should never talk to the user. It should do no input and produce no output. Instead, it should operate only through method calls.

Step 2: Implement the Vending Machine and Break Room Classes

Once the class diagram is finished, implement your vending machine in Java code. Then write a main method in a different class to simulate a user interacting with the machines in a break room. This method should create two vending machine objects and allow a user to interact with each one using a menu interface. It's up to you how you structure this, but one possibility is shown in the example dialog below.

```
Welcome to the Break Room!

There are two vending machines here:
1. VendingMachine: 8 chocolate, $1.99, $4.50, $45.25
2. VendingMachine: 1 iPad mini, $199.99, $0.00, $0.00

What would you like to do?
1. Enter money
2. Get change back
3. Vend an item
4. Leave the break room
Your Choice? 3
*** VEND AN ITEM ***

Which machine? 2
*** VEND FAILED: Not Enough Credit ***

Welcome to the Break Room. There are two vending machines here:
1. VendingMachine: 8 chocolate, $1.99, $4.50, $45.25
2. VendingMachine: 1 iPad mini, $199.99, $0.00, $0.00

What would you like to do?
1. Enter money
2. Get change back
3. Vend an item
4. Leave the break room
Your Choice? 4

Goodbye!
```

IMPORTANT BASIC RULE #2: The main method is the **view**. It talks to the user, calls the appropriate vending machine methods in response to the user's input, and displays the results. It does not determine the results of a user's actions or keep track of money. That's the job of the vending machine objects.

> **Extra challenge:** Implement your code in the `animate` method of the `FXAnimationTemplate` instead of in a `main` method. You will still get input from the user in the console, but you can show the vending machine display on the canvas. See `GraphicsExampleForAssignment2.java` for an example of this way of using `animate`.

Encapsulation and Documentation

Make sure you **encapsulate** your instance variables, with appropriate **get** and **set** methods and a **`toString`** method, and follow the **Documentation Standards** posted on Canvas.

Handing In

You have about 1 week to complete this assignment. See the due date and time on the Canvas assignment. Hand in by attaching a zipped file of your two **.java** (not .class) files and your class diagram to the drop box.

Evaluation

Your assignment will be evaluated for performance (20%), class diagram (20%), structure (40%), and documentation (20%) using the rubric in the drop box.