Vending Machine

**How was it developed?**

It started with creating the vending machine class with products and coins.coins. This was later split into their own classes:

* Vending machine (holds the main logic)
* Product
* Coins

The vending machine is initialized with products such as (Tango, Pepsi and KitKat) with their prices and quantities in the machine. Also, the machine has available coins in the UK denomination of (1p, 2p, 5p, 10p, 20p, 50p, £1 and £2).

The vending machine keeps track of when a product is selected and removed and returns the appropriate change to the user.

1. Initially started with defining a class for the vending machine to include change when created, contains products, and products selected have a price and customer is able to select a product.
2. Next step was to assign the quantity of coins to the demonination like for the change 10p – there are 10 quantity. And the ability to covert the demoniation to pence.

Improve the products to contain the product name, price and quantity

Also, create a method for converting the price into pounds or pence

Also provides change for a when a product is purchased.

1. Each product has a defined quantity
2. Separated product into its own class
3. **Separated coins into its own class**

**incorporated methods for receiving cash, converting cash and giving out change**

1. **Incorporating Coins element into the vending machine**
2. **When a product is purchased and the quantity is correct, the coins will be added to the vending machine**

# Using **Class-responsibility-collaboration card**

###### Product

- knows its price

- knows its name

###### Coin

- knows its value

###### Container

- knows how many products of each kind it holds

- is initialised with an equal amount of each type of product

- releases product

- receives products

- knows when it is full

- knows when it is empty

###### Till

- knows how many coins of each kind it holds

- accepts coins

- releases coins

- knows when it is full

- knows when it is empty

###### MiniComputer

- loads the vending machine with products and coins on initialization - by default 25 products of each type and 20 coins of each type.

- calculates and keeps track of amount inserted

- asks for more money if money inserted not enough

- decides whether to release product

- calculates change that needs to be returned and knows how many coins to return

- puts the coins earned in the till after purchase

### What is it?

Vending machine that allows to select products and receive the products. If too much money is inserted, returns the change, if the money is not enough, asks for more.

### How did I work?

I started creating the machine class with products and coins on initialization. Then I divided the code in three different classes, one for the machine (which has the main methods), one for the single product, and one for the stack of coins. The .buy method inside the machine class is the most important, in which you can find all the logic for the products and the amount of money provided. The machine is initialized with some defined products (in this case Coke, Pringles and Mars), and a stack of coins of the UK denomination (5 for each small coin, 2 coin of £2 and £1). Every time an object is released from the machine, the relative amount of coin is added to the machine stack.

### How to use it?

Install ruby with RVM (if haven't done already): $ \curl -sSL https://get.rvm.io | bash

$ rvm install 2.1.1

$ git clone https://github.com/mserino/Vending-machine

$ cd Vending-machine

$ irb

> require './lib/machine'

> machine = Machine.new

> machine.buy("Coke", "£1.5")

if the amount is correct you'll receive this message

Your product: Coke

> machine.buy("Coke", "£2")

if the amount is more than the price you'll receive this message

Your product: Coke - Change: 50p

> machine.buy("Coke", "£1")

if the amount is not enough you'll be asked to add more money

Please insert another 50p

> machine.add("50p", "Coke")

Your product: Coke

You can check anytime the quantity of products and the amount of coin in the machine

> machine.products

> machine.coins

If you want to reload products and coins:

> machine.reload