# 2.1 Task 1 : Review your code and finding refactoring points (10 points )

Provide a description of your refactoring process. Describe what do you want to change and why do you want to change it.

## Refactoring Type 1, Code Smells: Dispensables

### Clean up dead attributes and methods

* There are a handful of methods and attributes that I did not have time to write full implementations for and cleaning these up will make my code easier to understand
  + Beverage class
    - caffeine and calories attributes and associated methods
  + Coffee class
    - grindSize and agitation and associated methods
  + HotBeverage
    - waterTemperature and associated methods
* Makes code easier to read and maintain, eliminating dead code.

## Refactoring Type 2, Big Refactoring: Converting Procedural Design to objects

* Added the command pattern to the Customer package.
  + This increases the number of classes, however makes the code base much more maintainable because - before in the simulations, we would have to manually call these command methods (i.e. ```customer.chooseBeverage()``` etc.) and now these commands are encapsulated within Choose classes which implement the Choose interface.
  + The choice method automates binding the Customer's choice to the HotBeverage object attributes and returns the updated beverage for easy printing.
  + Further, now with the Ticket class, part of the command pattern that remembers the commands.
  + There is also undoOrder() which is an important feature of the Command Pattern.

## Refactoring Type 3, Composing Methods: Simplifying code by factoring out methods

* As it turns out I had in fact had a HotBeverage class attribute HashMap for condiments previously
* This data structure was being used in the the same class' ```addCondiments()``` method however was not being accessed in the simulation.
* I greatly simplified the simulation by adding a ```toString()``` method for HotBeveragewhich returns a HotBeverage description based on it's type and the contents of the condiments class attribute HashMap.
* I further Changed the Coffee and Tea subclasses of HotBeverage to set and get the type attribute to their superclass to enhance the power of HotBeverages ```toString``` and removed the subclasses Coffee and Tea's class attribute ```type``` to eliminate code smell.

# 2.2 Solution Description Change and Code Change including UML Class Diagram

* Description included above and Github links below
* <https://github.com/metcs/met-cs665-assignment-6-MichaelKramerGuitar>

. VenBev Refactored Class Diagram also available [here](https://lucid.app/lucidchart/01cc292f-b065-4908-8137-e9a8649ca142/edit?invitationId=inv_0e897616-2cc7-4f14-bbf1-d7c210a52233&page=0_0%23)

A picture containing timeline

Description automatically generated