# 2.1 Task 1: Implementation Description

* High Level Description and Approach
  + The high-level concept of this implementation is that VendBev is a vending machine that has the following components to achieve a vend
    - Various Beverages (currently only Coffees and Teas)
    - Order
      * Has the details of the Beverage being ordered such as
        + Beverage type
        + Condiments and sweeteners
    - Barista
      * Brews Beverage
      * Calculates Order Price
    - Menu
      * Has information about beverages, condiments and prices
    - Customer
      * Chooses from menu options from which an Order can be created
* How flexible is your implementation, e.g., how you add or remove in future new drink types?
  + This implementation is highly flexible
  + The Beverage abstract class is extended by a Concrete class HotBeverage to support the current use case
    - To this end HotBeverage is extended by both Coffee and Tea subclasses
  + Using inheritance in this way one might imagine adding beverage types or new drinks by creating a concrete class extending Beverage, ColdBeverages
    - ColdBeverages might be extended by Juice and SoftDrink etc.
  + Inheritance in this way allows us to utilize polymorphism with our set and getType() methods from Beverage
    - These types in this current implementation are determined by the creation of two intermediary classes
      * The Menu which has String array attributes for coffee and tea choices, in addition to condiments, sizes and prices respectively.
      * The Customer class has methods to choose from these array values (at random in this implementation for simulation) and thus the appropriate Beverage sub type is determined.
    - If we wished to add or remove new drink types in the future it would simply be a matter of either updating the above referenced String arrays or extending the Beverage class appropriately and creating new appropriate String array attributes for said Beverage subclasses.
* How is the simplicity and understandability of your implementation?
  + With a pre-requisite basic knowledge of inheritance and polymorphism, this implementation is quite simple.
  + It would further be quite intuitive to explain to a superior or supervisor who did not understand how to write software because the principles of inheritance utilized in this implementation are inspired by and mirror a Starbucks (or any Coffee shop) experience, only in code.
    - To this end, once we understand that a Beverage is an abstraction (not a type to actually be ordered) of a HotBeverage (still not to be actually ordered) which is an abstraction of both Tea and Coffee (types the customer will actually order) basically we can easily see the following: (*see how you avoided duplicated code*)
* How you avoided duplicated code?
  + Through inheritance and designing classes modularity based on unique functional characteristics
    - A customer chooses their beverage, size and condiments from a menu
    - A menu has information on prices of available beverage types according to size and type
    - An order is created from the customers choice
    - A barista brews the order, charges the customer and delivers the beverage when ready
      * This implementation uses threading to simulate the concept of brewing multiple beverages at once and sleeps the threads to elongate the customer experience to a comparable real world brew time per order
  + Effort was made to determine the necessary processes and methods to meet the requirements laid out in the Implementation Description and to utilize the single responsibility principle learned in CS622 Advanced Programming Techniques
    - Each class should have a single job, or only one reason for its data to change
      * Each class method should have a single job, changing one single attribute of its class
  + Inheritance in Java gives us the Open Closed principle as well to facilitate avoiding duplication of code where objects are open for extension but closed for modification.

# 2.2 Task 2 UML VendBev Class Diagram

. VendBev Class Diagram also available [here](https://lucid.app/lucidchart/2484a2af-cd51-435b-a4b1-ae84ba4731b4/edit?beaconFlowId=7CAF6B5C9AB6C9EE&invitationId=inv_7b24c6f0-bc6f-43cf-8635-d22b8b35fd71&page=0_0%23)Diagram

Description automatically generated

# 2.2 Task 2.2 VendBev Sequence Diagram

. VendBev Sequence Diagram also available  [here](%20https:/lucid.app/lucidchart/8d158900-fab4-4af9-9b1c-abb8c04e7eb0/edit?beaconFlowId=3068D87DE955C7FF&invitationId=inv_bda9f4e1-11ee-497a-ba49-74b1d2093c9d&page=0_0%23)

A picture containing diagram

Description automatically generated

# 2.3 Task 3 Code:

<https://github.com/metcs/met-cs665-assignment-1-MichaelKramerGuitar>