The main vending machine design centers around the VendingMachine and Simulator class, which reads in the machine's accepted currency and products. Both of these options are read in from a text file that defines all the information required to construct their objects. The advantage for the currency is that changing between national currencies is as simple as overwriting the text file with the values and names of the different currencies. The only limitation of this method is that all currencies that might be used must be included in this file, even if their quantity is zero, because this list also defines all the currency items that are accepted from users. A similar method is used for the items to be sold, where the items and all the requisite information to construct the inventory. Again the limitation is that all possible products to be sold, even if they have zero quantity, must be in this file since there is not a mechanism currently built in to add products after the fact. Technically there are set methods in the VendingMachine class, but they are never called by this implementation.

The other major design choice revolves around developing the Currency and Product classes to define a class that represents the actual real-world objects. Currency is meant to represent a dollar bill or quarter similarly to the Product being meant to represent something like a Coke or banana. The Change object contains a Currency object and also contains the amount and is used to define money contained within the vending machine. Similarly an InventoryItem contains a Product and a quantity which is used to define what is sold within the vending machine.

One initial implementation I tried was making the Product and Currency classes abstract and having InventoryItem and Change be the concrete classes. I decided against this because it made more sense to be able to use the two as their own concrete classes rather than only relying solely on classes used by the vending machine internally.

If I were redoing this project I think I would prefer not to make Packaging and Material their own classes. In my initial design plan it made sense to make them classes because they felt like they could be more complex than only holding a simple string. In an expanded program it might make sense to expand these classes to only allow specific values, or contain more detailed information about the property that could expand the functionality of the program. Since that was not included in my current implementation it may have made more sense to just use a string field in the class.