This assignment was to create examples of 3 different design patterns talked about in this module. Patterns such as factory pattern, proxy, and facade will indeed be included in my final project, but I created a simple example of the default three: factory, singleton, and facade to give me more practice of implementing the patterns.

Factory Pattern

In EmployeeFactory.py, I implement a few different classes describing employees. This includes a Manager and a LineWorker. The factory pattern abstracts object creation into a factory class, which I called EmployeeFactory. I then can create a Manager or a LineWorker depending on my factory's create method inputs and the factory will return an Employee of that type. My main method tests this by ensuring a Manager is of type Manager and my LineWorker is of type LineWorker.

Singleton Pattern

In MySingleton.py, I create a class called MySingleton. This singleton includes a uuid generated by the initialization of the class. This is used for our test where we fetch two instances of the MySingleton class. If there were two distinct initializations, the uuids would not match. However, our assert does indeed pass and we know that we receive a duplicate issue.

Facade Pattern

For the facade pattern, I decided to write a program to clean my house (I wish it would actually affect the house itself, but one can dream, right?)! Cleaning the house takes multiple steps, including washing dishes, doing laundry, sweeping, mopping, etc. I created a class called HouseCleaner which handles coordinating all of that work. Our main method can then simply call HouseCleaner.cleanHouse and the house will be all clean!