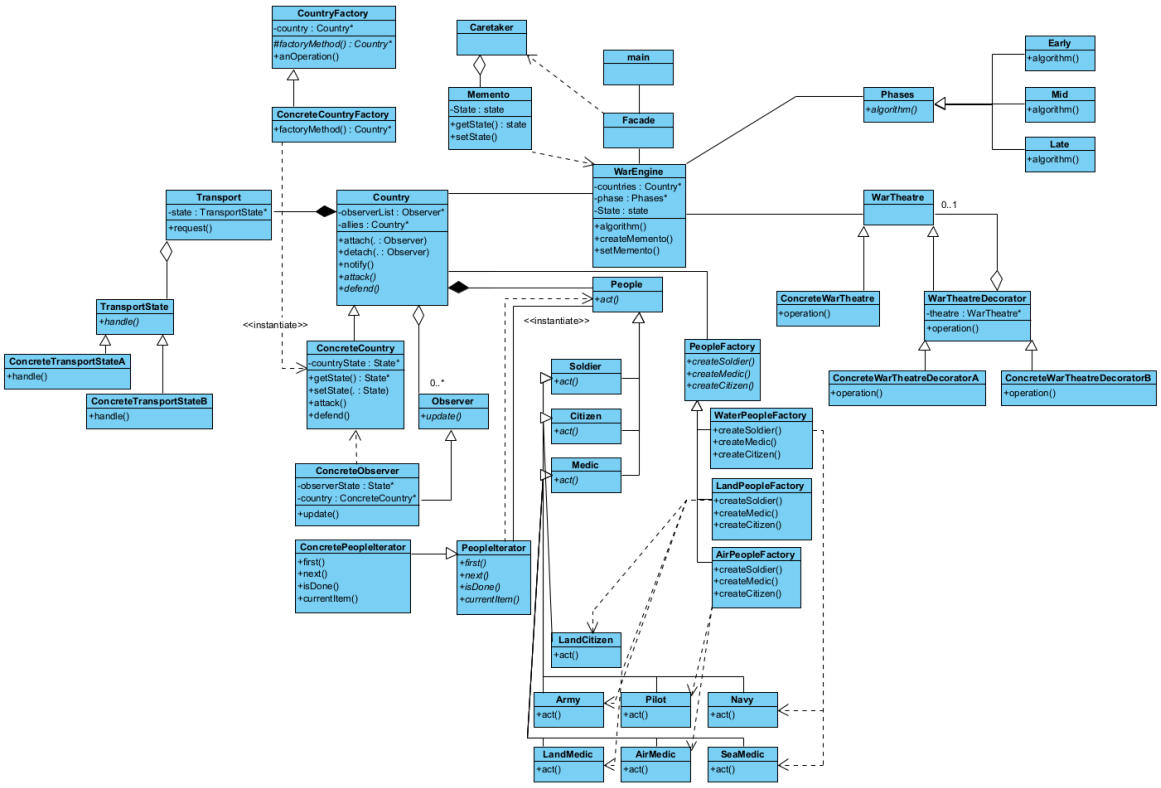

Figure 1: Full UML Diagram

Country

This will be the abstract class for the Countries.

Will include:

- A Name
- An array of allied countries and an array of enemy countries
- A pure virtual attack and defend function
- An array of people that will represent the people living in the country at the start of the war

ConcreteCountry

Concrete class for the countries involved in the war

*Implement the attack and defend methods by iterating through the people array and calling each of their respective “act” methods. (The “act” method in the people class will handle what actually happens during each turn)

CountryFactory

This will the abstract class for the CountryFactories

Includes:

- The pure virtual factory method that will be defined in the concrete factory

ConcreteCountryFactory

Concrete class for the CountryFactories

Implement the factory method. (Ensure it returns a country. How to initialize some of the variables?)

Transport

Class that will hold the state of the transport lines of a specific country

Includes:

- A TransportState*
- A method ‘request’
    - Request(){
        TransportState.handle()
      }

TransportState

Abstract class that represents the state of the transport lines

Includes a single pure virtual function ‘handle’

ConcreteTransportStateA….Z?

Concrete classes that will represent as many states for the transport lines that we will need

Implement the ‘handle’ method in such a way that it will return a value that can be used to manipulate the Country class’s ‘attack’ and ‘defend’ methods. (The attack/defend methods will do something different depending on whether the transport state is in a good condition or not)

Observer

Abstract class for the country observer

Includes a pure virtual function called ‘update’

ConcreteObserver

Includes:

- Implementation of the ‘update’ method -> Update() will update the observerState to the current state of the country.
- ObserverState* state
- ConcreteCountry* subject

Country will need to have the appropriate methods added to it so that it conforms to the Subject participant in the observer pattern.

People/PeopleFactories/PeopleIterator

This is self-explanatory if you have any questions just ask me.