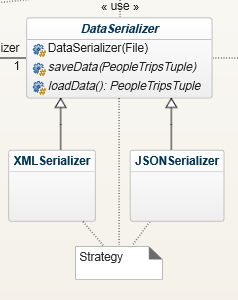
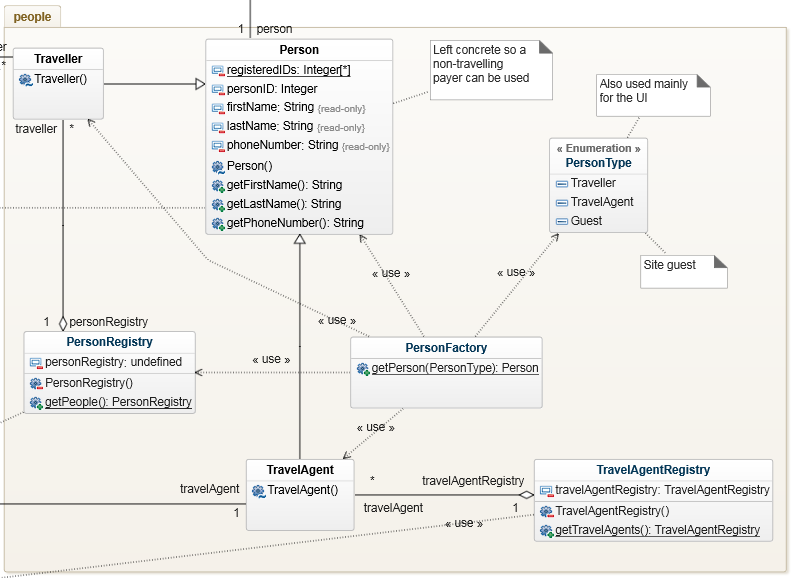
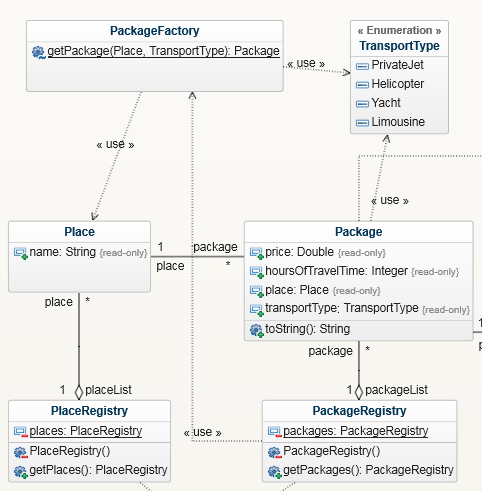
## Strategy

The strategy pattern will use an abstract class called *DataSerializer* to define the operations to be used. XMLSerializer and JSONSerializer will be swapped out inside the Cache object depending on which scheme is set using an associated enum. The *saveData* and *loadData* methods will be implemented in the children classes according to their respective encoding scheme.

## 2 Factories

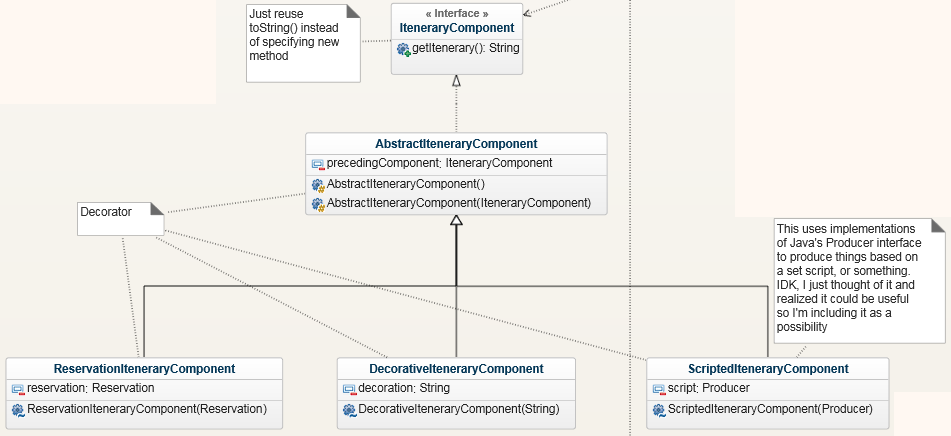


 The factory functionality as defined in the Iteration 2 instructions will be handled directly by the serializer. Instead, 2 different factories will be implemented.

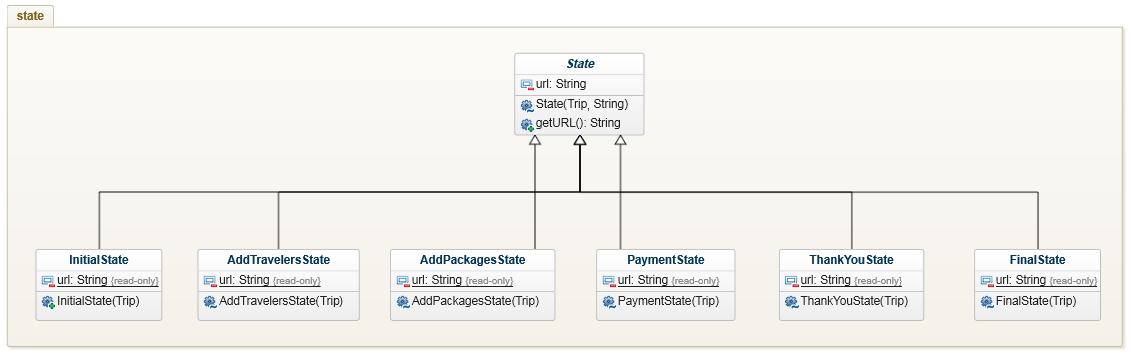
The first will create new Person classes based on the PersonType enum. The signature has been shortened in the UML diagram, but it will accept the first and last names as well as the phone number. It will then register the person with the appropriate registry.

The second factory will be used to construct the available packages for the package registry based on the different places to visit and transportation types to get there.

## Decorator



## State

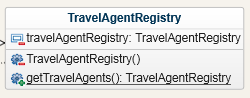


The state machine will have a very minimal implementation on the backend. It will mostly be limited to a state flag role. The state machine’s functionality will be implemented on the frontend. When a Trip is loaded, its state will be retrieved, from which will be determined the Trip’s current state’s URL. Once the frontend has completed its current state, it will set the next state in the Trip and load that page.

## Package Singleton



## Travel Agent Singleton



### Person Singleton

