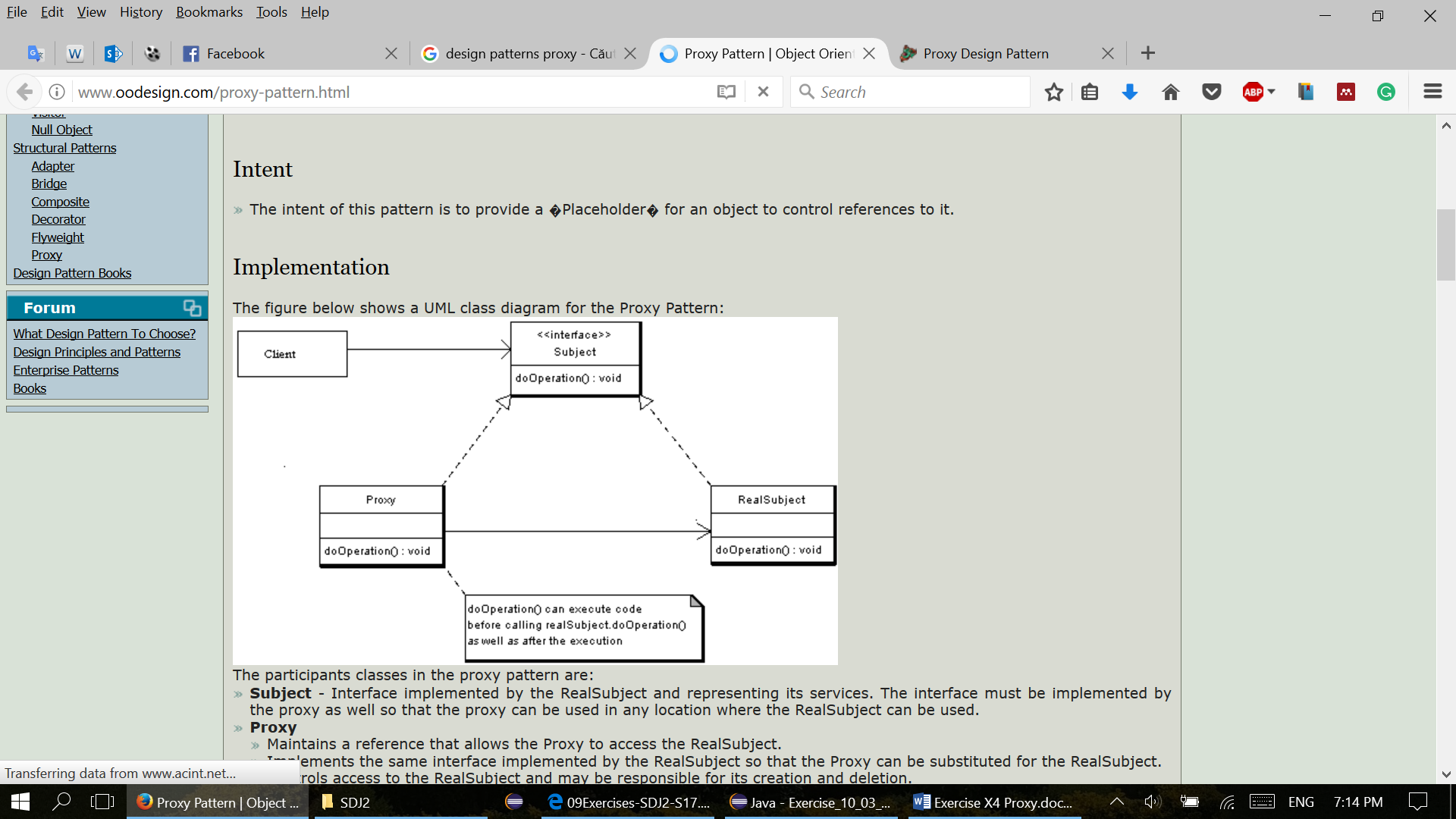
**Proxy**

**A general UML class diagram of a Proxy design pattern.**



**The overall purpose for the Proxy design pattern.**

Provide a placeholder for another object to control access to it. A wrapper to delegate work to a real subject.

It can be a thread safe collection delegating work to another collection in synchronized methods, or a client object with access to a model on another computer.

**The description of the general UML class diagram (each interface and class). Purpose and relation.**

**The Subject** Interface implemented by the RealSubject and representing its services. The interface must be implemented by the proxy as well, so that the proxy can be used in any location where the RealSubject can be used.

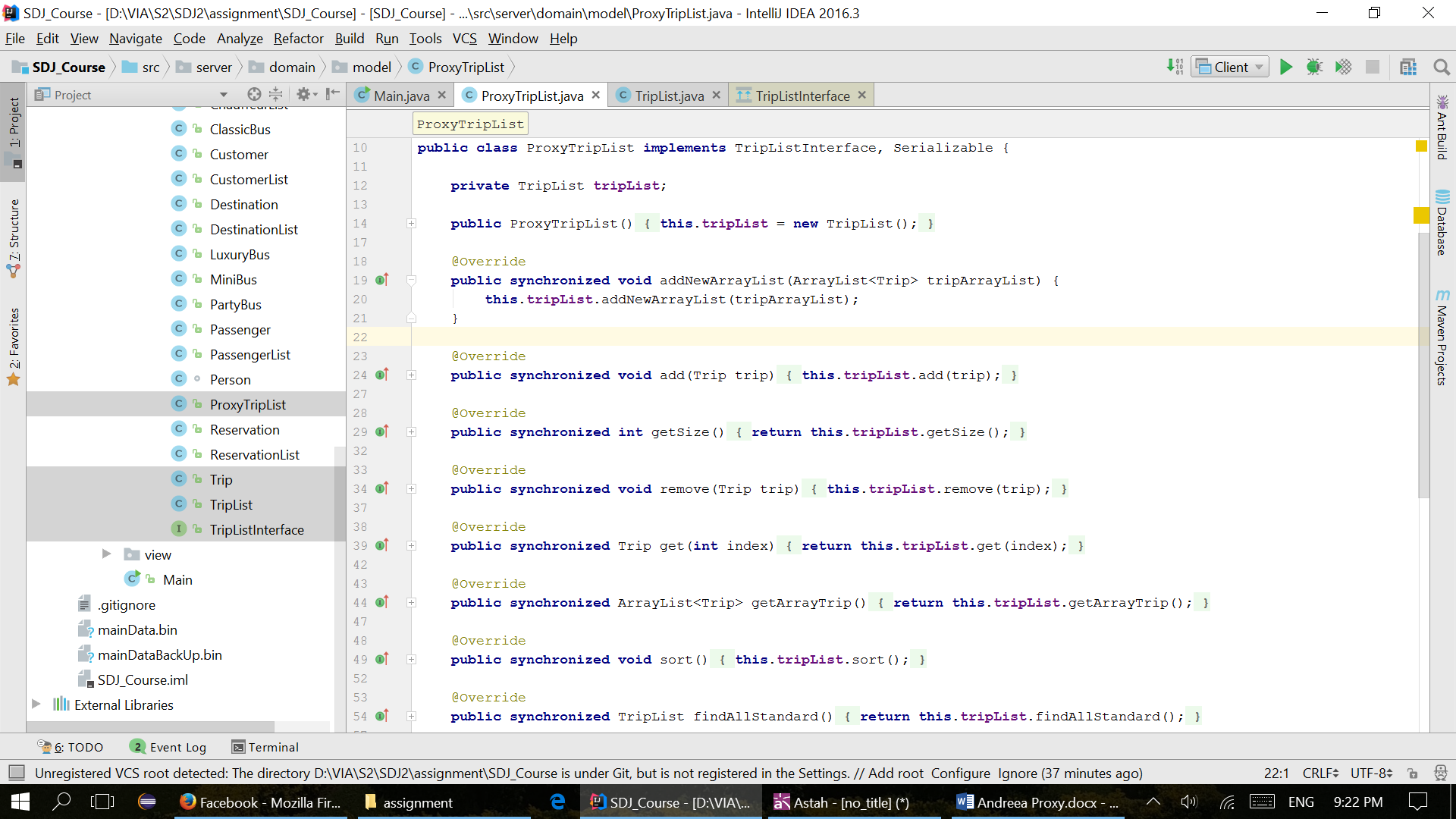
**The Proxy** has an instance variable from RealSubject that allows the Proxy to access it. Implements the same interface implemented by the RealSubject, so that the Proxy can be substituted for the RealSubject. Controls access to the RealSubject and may be responsible for its creation and deletion.

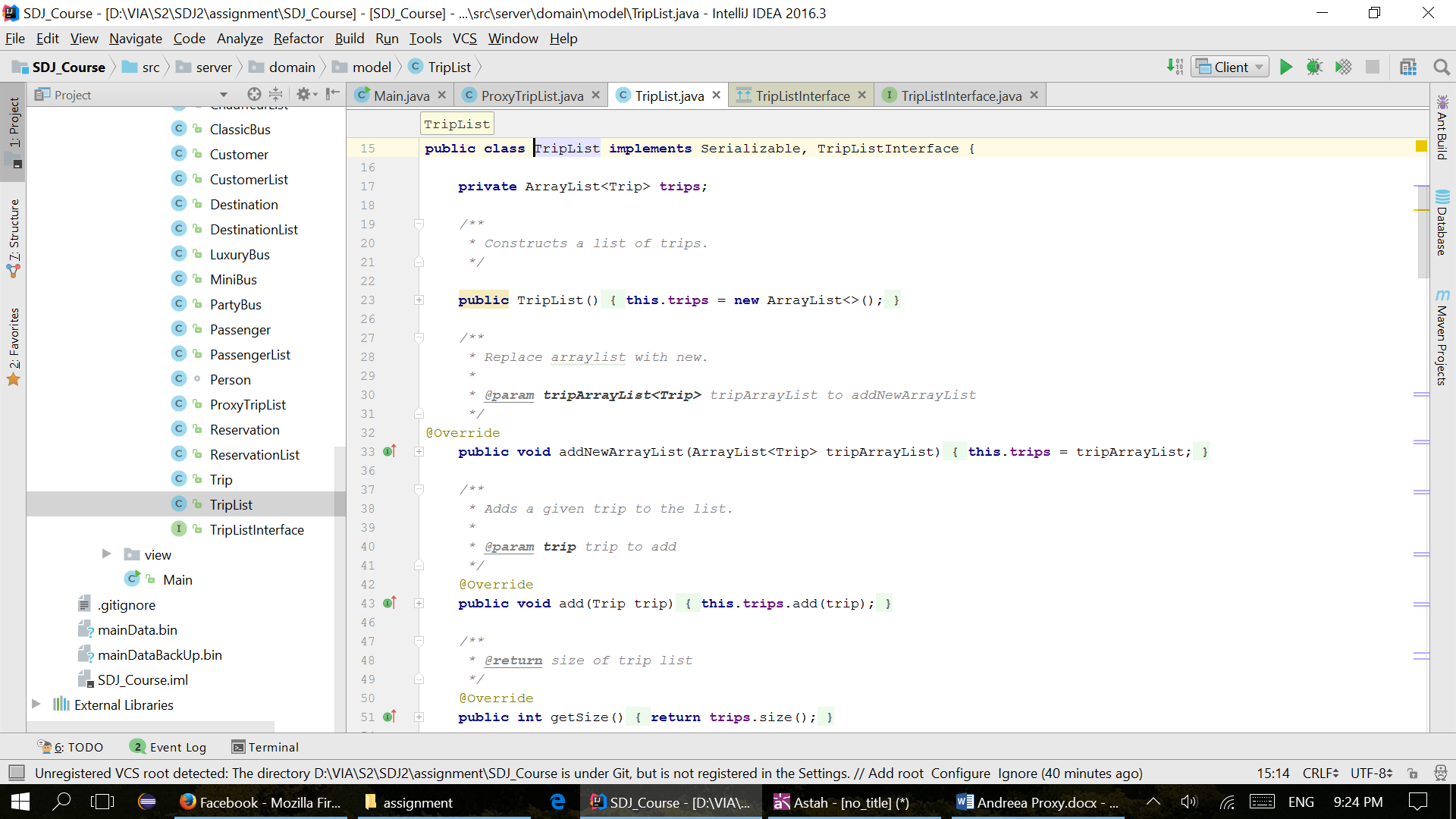
**RealSubject** - the real object that the proxy represents.

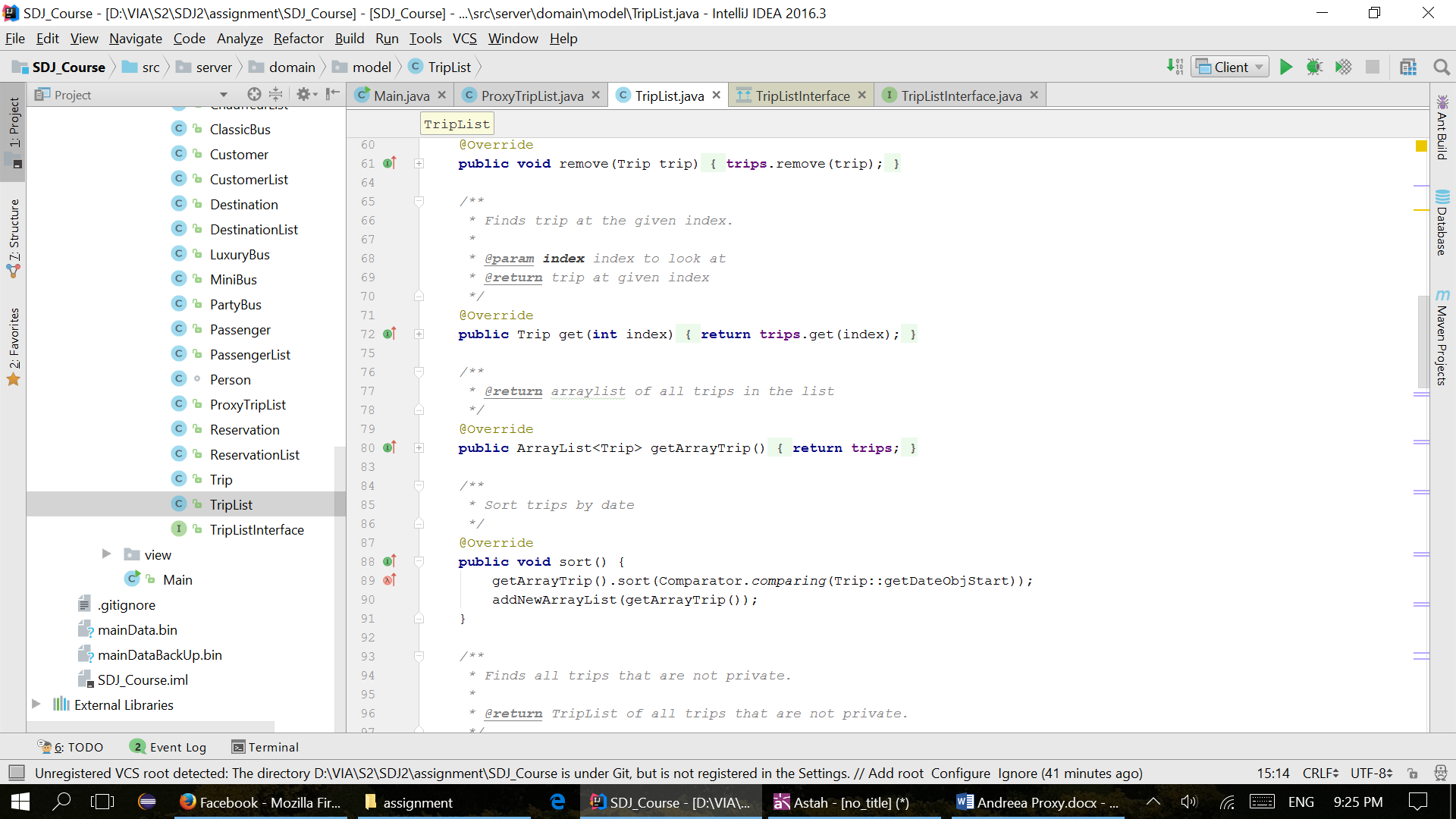
**How to use a proxy design pattern.**

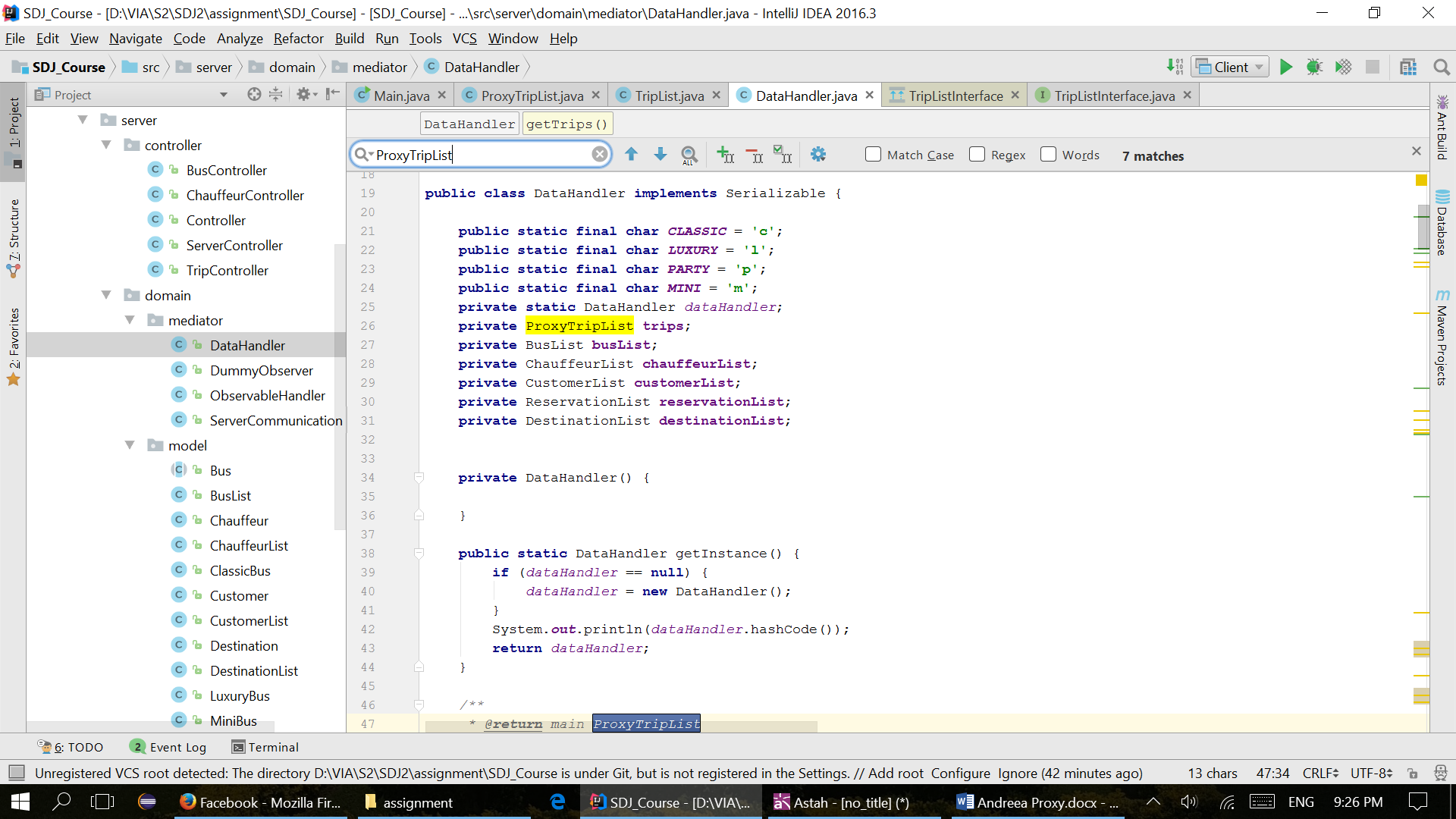
A client obtains a reference to a Proxy, then the client handles the proxy in the same way it handles RealSubject and therefor invoking the method doSomething(). At that point the proxy can do different things prior to invoking RealSubject - doSomething() method. The client might create a RealSubject object at that point, perform initialization, check permissions of the client to invoke the method, and then invoke the method on the object. The client can also do additional tasks after invoking the doSomething() method, such as incrementing the number of references to the object.

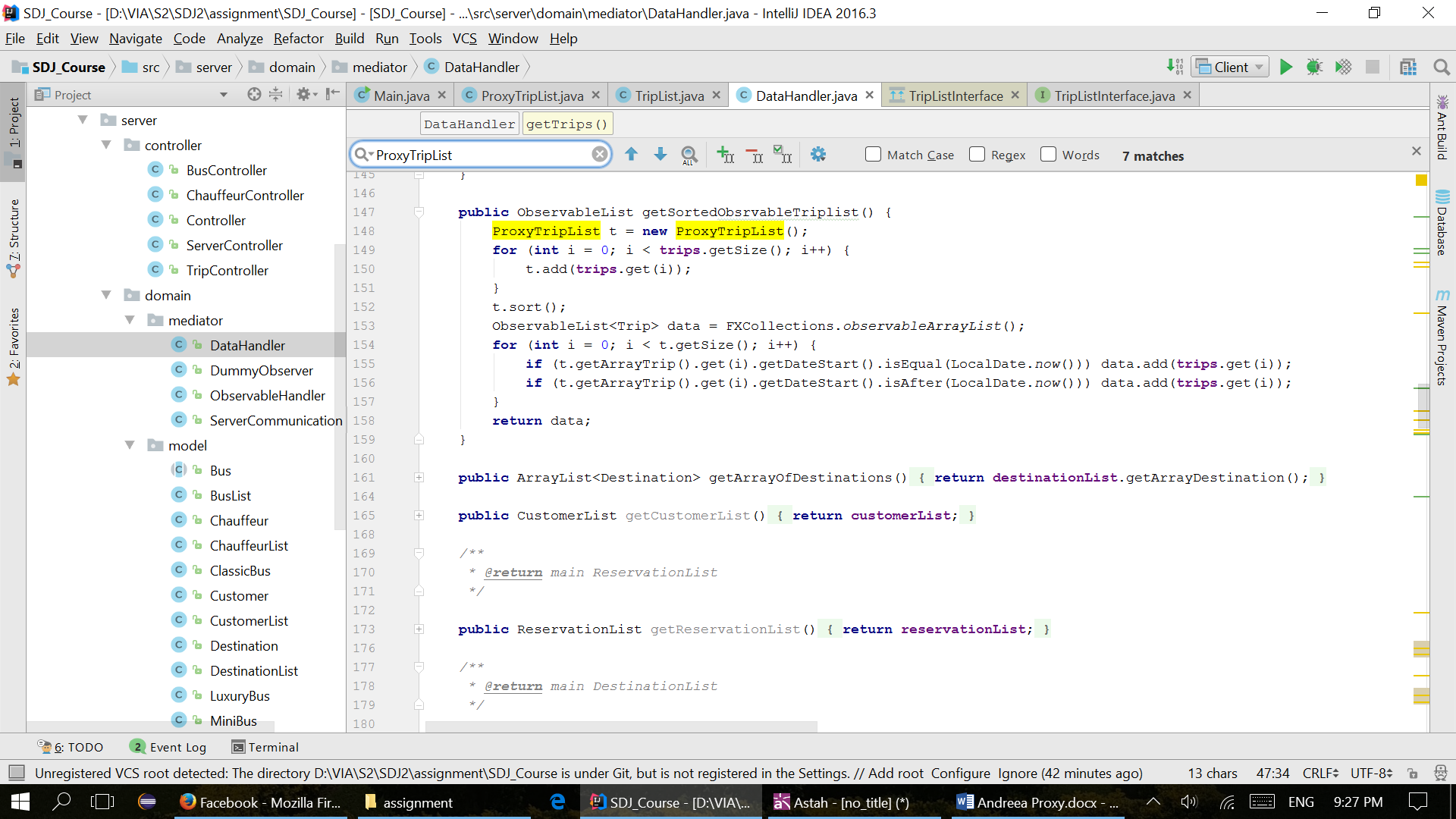
**Code examples for each part of the Proxy pattern.**

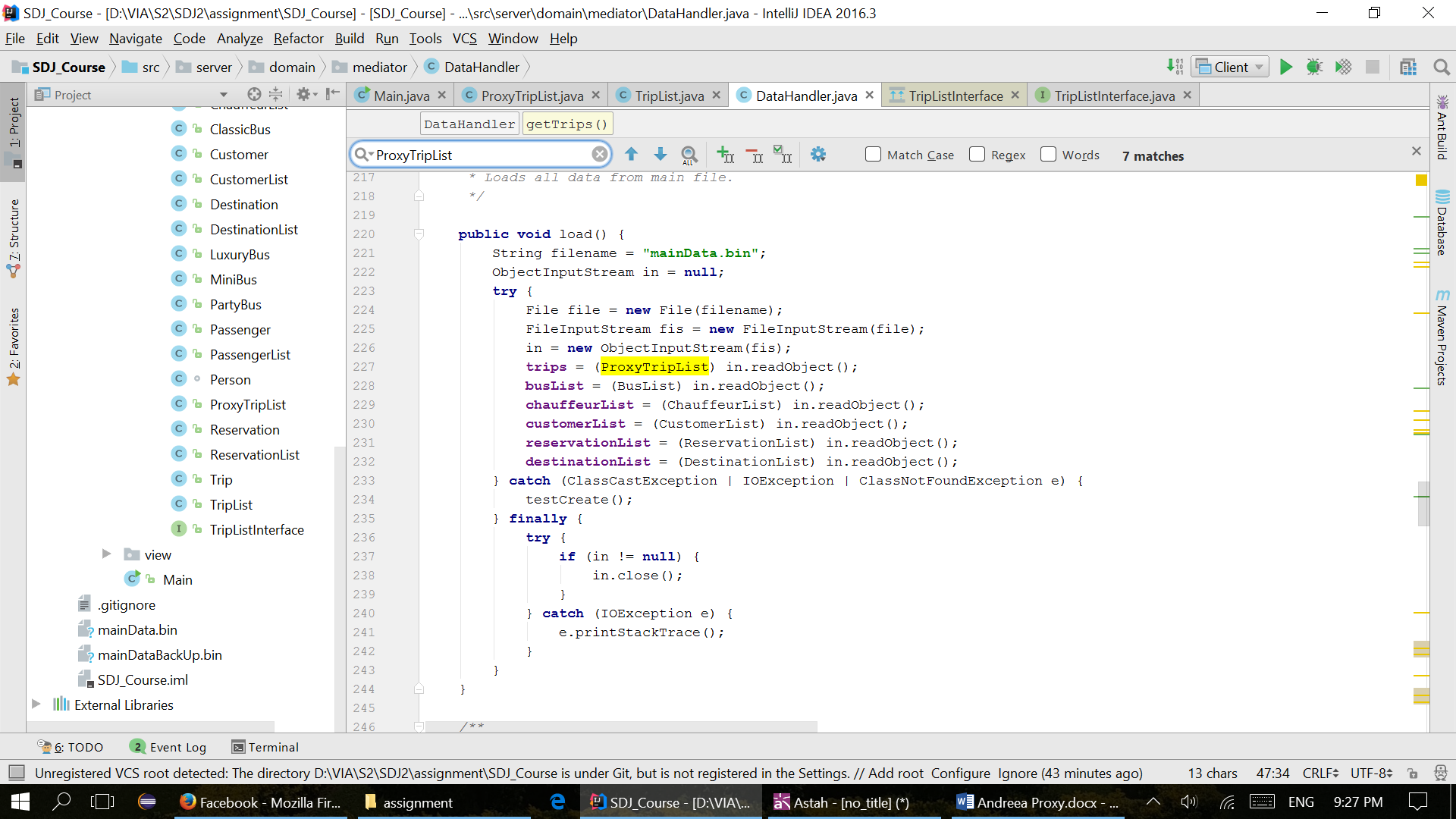












**The UML class diagram of the Proxy design pattern implemented.**

