**Texas Tech University**

**Computer Science Department - Software Engineering**

**Team-based Project: Release # 3**

Team Members: Dipen Pratap, Vizion Poudel, Pritish Ayer

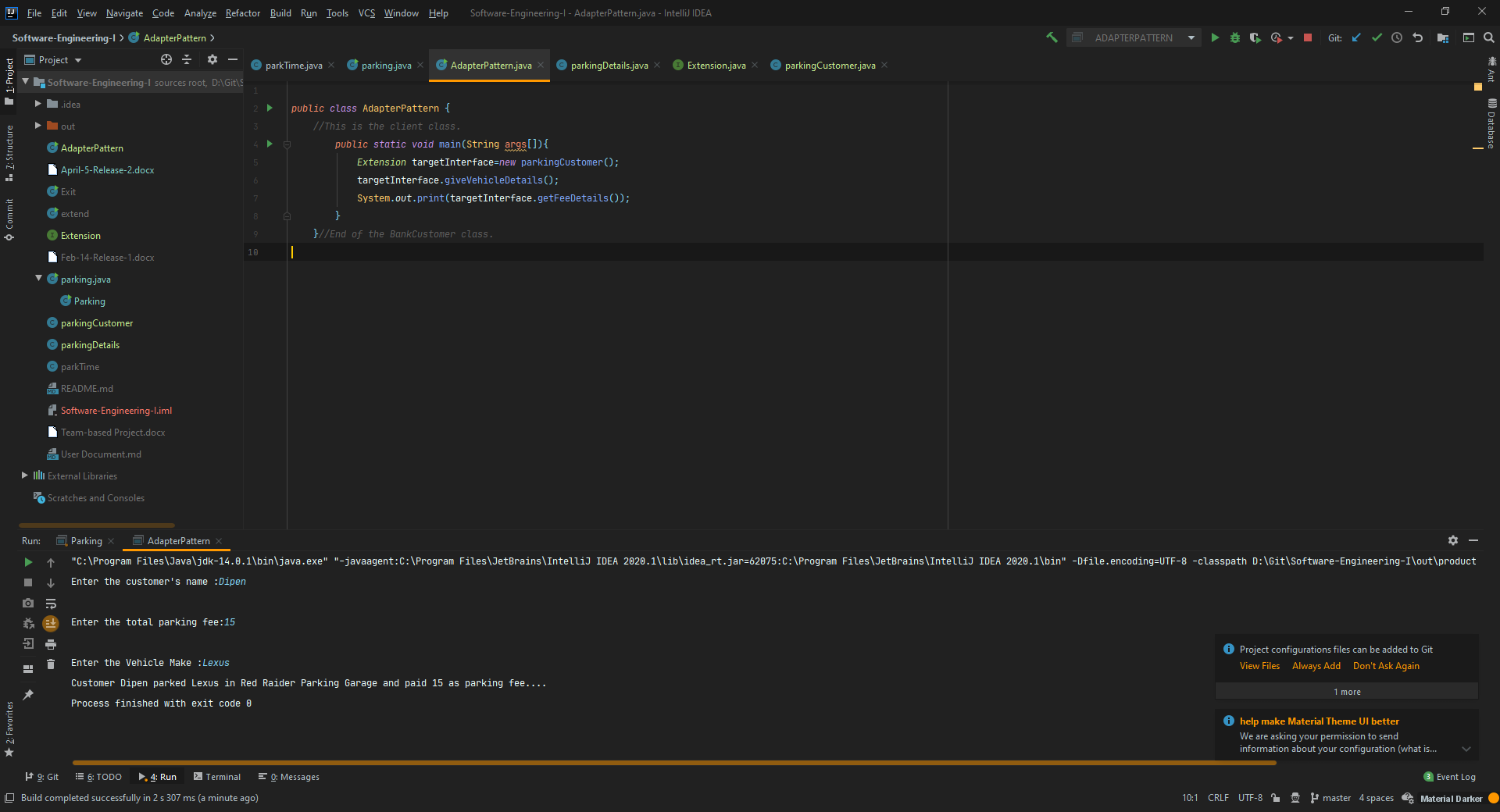
A short description of what kind of design patterns (two of them) you have implemented along with the location in the code.

* **Adapter Pattern**

An Adapter Pattern says that just "converts the interface of a class into another interface that a client wants". In other words, to provide the interface according to client requirement while using the services of a class with a different interface. The Adapter Pattern is also known as Wrapper.

The advantages of the adapter Pattern are that it allows two or more previously incompatible objects to interact and also allows reusability of existing functionality. There are the following specifications for the adapter pattern:

* **Target Interface:** This is the desired interface class which will be used by the clients.
* **Adapter class:** This class is a wrapper class which implements the desired target interface and modifies the specific request available from the Adaptee class.
* **Adaptee class:** This is the class which is used by the Adapter class to reuse the existing functionality and modify them for desired use.
* **Client:** This class will interact with the Adapter class.



* **Observer Pattern**

An Observer Pattern says that "just define a one-to-one dependency so that when one object changes state, all its dependents are notified and updated automatically".

The major advantages of the adapter Pattern are that it describes the coupling between the objects and the observer and provides the support for broadcast-type communication.

* **Singleton Pattern**

Singleton Pattern says that just "define a class that has only one instance and provides a global point of access to it". In other words, a class must ensure that only single instance should be created, and single object can be used by all other classes.

Early Instantiation (creation of instance at load time) and Lazy Instantiation (creation of instance when required) are two types of singleton patterns. The biggest advantage of the singleton Pattern is that it saves memory because object is not created at each request.

To create the singleton class, we need to have:

**Static member:** It gets memory only once because of static, it contains the instance of the Singleton class.

**Private constructor:** It will prevent to instantiate the Singleton class from outside the class.

**Static factory method:** This provides the global point of access to the Singleton object and returns the instance to the caller.