How to create and implement!

Extensible – code that can be easily changed in the face of new requirements. A contract can have as many implementations applied to a single contract.

Repository Pattern – a common design pattern that adds a layer of abstraction to the CRUD of data stores. A layer to separate our application from the data storage technology.

Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.

This means the app doesn’t need to know the details of the data store.

The Repository sits between the app and the data store.

The app cares about the contract defined by the Repository.

You’d have technology specific repository classes that implement the Repository interface. This means the app can use any and they should work. It also means new ones can be added.

AppDomain.CurrentDomain.BaseDirectory – the same location as the current executable.

String.Split(‘,’) – will divide a string by the character. This example can be used for a CSV file.

Factory Method – Add a Factory class. Public. Static. Public static method that returns the specific type as a generic interface. Basically use a switch statement. If a method is handed an invalid arg option but it’s the correct type you can throw an ArgumentException.