Vy Duong

CSC 470

**Object Pool Design Pattern**

**What is it and what is the problem that it solves?**

It is known as a resource pool. It is used to manage the object caching. This method will reuse the object that has been instantiated so it can cut down the cost of object instantiation. It is a good idea to keep all reusable objects that are not currently in the same object pool so they can be reused if needed. It is designed to be a singleton class.

However, if an object is checked out from the pool, the object pool can also create a new object if they need to, but they must clean up unused object periodically

The reusable pool has its constructor private, which forces other classes to call its getIntances method to get the instance of the class

**Example of when you might use it - Use Case**

An object pool is similar to an office warehouse. When a new employee comes into the office, the manager will take the tool that is needed for the job for the employee to use. If there is a tool that is not in the warehouse, the manager can order a tool for the employee to use. After the employee leaves the company, the tools are done using. Those tools will go back to the warehouse so that the other employee can use them.

**Facade Design Pattern**

**What is it and what is the problem that it solves?**

It is a commonly used design pattern. A façade is an object that provides a simplified interface to a larger body of code. It makes the code easy and more understandable. It is often used when the system code is hard to understand. The patterns hide the complexity of the code under a wrapper. When the user uses the wrapper, they just need to plug in the value that they want and the wrapper will generate the code behind the scene to implement that wrapper

It provides a unified interface so that the subsystem is easy to use. It also reduces a learning curve for new programmers

**Example of when you might use it - Use Case**

For example, in web design, you need to create an alert banner at a certain length, with a certain color, a certain symbol at a certain position. You will use that similar in a lot of places for a company website. For example, a success banner when the user logs in or apply for something successful. It is a good idea to write all of the code for that banner and create a wrapper for it. For example, @Hthml.Banner(“Alert”).Create(). When this code is used, the banner will automatically create with the right length, symbol, color, text so that the programmer can save time to write other code and keep the consistency for the website

**Factory Design Pattern**

**What is it and what is the problem that it solves?**

This method lets a class instantiation an object using a subclass

There might be a lot of applications out there in a company product. This framework will allow individual applications to define their own domain objects and provide their instantiation. The user will give the creation details to a subclass. From there, the subclass can instantiate the object using the user given input. The Factory pattern only requires a new operation

**Example of when you might use it - Use Case**

For example, the user wants to create a car. There are 5 different colors of the car which are red, blue, white, yellow, and black. As the user put in the input “Black”, the factory will match that input with the BlackCarCreation() Method to create a black car with its part. If the user chooses the white car, the factory will match that input with the WhiteCarCreation() method to create a white car with its part. For the factory design, you need to create an interface, create classes implementing the same interface, create a factory to generate an object of the class based on given information. The factory will pass the information to the class such as type to get the object and then give the output