Kristopher Ferrell

Period 4

8/30/21

Mr. Beck – Independent Studies

Project Progress Report

The Singleton is a creational design pattern that’s supposed to restrict a class to a single instance, while providing global access to that instance. You should use it whenever that single instance needs to be extended with subclassing, or when a class needs to only have one single instance with one easily accessible point. There is a huge debate over whether the singleton should be used. Some developers have noticed that most times when this pattern is used, it only makes the codebase harder to understand and are lead to consider it an anti-pattern that should NEVER be implemented. I believe in moderation – only use a singleton when absolutely necessary. Don’t take this lightly, either. You have to be sure that a class really makes sense to only have one instance of it in the codebase’s entire existence, because if you decide later on that you actually needed multiple instances for a singleton, then changing it back to a non-instance object can be really tough. The best way to implement a singleton is by using a single element enum type. It looks like this:

Public enum Singleton {

INSTANCE;

//methods

}

Global access has a lot of negatives. A big one is that any object used by a global accessed object won’t be able to be deleted by the Java Runtime Environment (garbage collection), so you must be careful in considering which objects a singleton object has access to in its methods.

Why would learning this help my career? Design patterns are well known, and are great for communicating basic patterns of implementing application-specific code in different ways through them. Because they fix common problems, it is important especially for new developers to learn to get into the software developing mindset better earlier on before they’re faced with real world tasks where it may be hard to learn at the same time later on. The Singleton can be paired with other common design patterns, such as a façade object, which usually only needs one instance anyways. Knowing the singleton along with its negatives and positives helps you be more knowledgeable in deciding when to use it and when not to use it.

Quiz