Homework #4 - Proxy Class Pattern Design

Ryan Magdaleno - CS342 - Fall 2023

The proxy class pattern design is an abstraction pattern design that allows for a class user to interact with an abstracted class via a proxy class. In the bundled program, I have a class called *ProxyOfMainClass* which acts like a wrapper for a class called *MainClass*. I made sure to have them both implement the same interface so they would both have the same function interface. The main functionality of *MainClass* is that it does simple attribute / String returns. The *MainClass* is never interacted with by a class user, the intended way to use this class would be via *ProxyOfMainClass*'s functions of the same name. *MainClass* objects are not loaded until a function that requires a returned object is needed, allowing for a programmer to control when certain resources are needed by the class wrapper *ProxyOfMainClass*. The pros of using something like this would be adding an extra layer of protection from the class user and underlying logic. The wrapper class may also allow for some extra functionality when interacting with the underlying class. The underlying class / codebase that the proxy class is linked with is decoupled between the two, allowing for quick and easier changes to the wrapper class without affecting the underlying class. The proxy class allows for lazy loading optimization, where the underlying class object isn't needed until requested by the proxy class. This type of design pattern adds an extra layer of security from a user, where they only get data from the underlying object when requested by the proxy class. Some cons that come with proxy classes would be that it adds another layer of abstraction complexity which for a codebase can make things all the more complex. The extra function calls and class object instantiations can reduce overall program performance depending on the types of operations the underlying class does and how often. My bundled program's GUI prints to a center console the *MainClass* class attributes via the *ProxyOfMainClass* class. The GUI class requests for the strings to print to the console only via the *ProxyOfMainClass* class effectively decoupling the GUI class and the *MainClass* object, making the codebase overall more abstracted.