Payson Blackwell

1/23/19

Special Projects CSD 399

Singleton

There are a lot of different scenarios where you may want only one instance of a given object to exist. Either you want it that way because limited memory resources, or maybe you want to simplify access to something like a database handler (or even something like a Math class object) and it wouldn’t make sense to have multiple instances of a helper object. One way to tackle this problem that people use, is to create a global variable for that object, and then refer to that variable when you need to access the methods inside of it. That way can work, but in bigger projects, it can be easy to lose track of global variables between files and it doesn’t prevent you from creating a new instantiation of that class. The right way to solve this problem is to implement the Singleton Pattern, which will prevent multiple instantiations of the class.

To implement singleton, it only requires a couple of small things. First, you need to set the default constructor to private, so it can’t be called by anyone wanting to make a new instance of the object. Second, you have a static variable of the same type as the class (This is usually called instance). And last but not least, you have a public method that is usually called getInstance() that returns the instance variable. In that method, if instance is null, you would call the default constructor to fill it up before you hand it back.

Now that you have singleton implemented, you may ask yourself, “*why did I go through that trouble instead of creating a global variable?*”. Great question, and I would reply “*because it was the smart thing to do, also stop questioning what I am telling you to do*”. What you may not realize, is that singleton is a lot more flexible than you might think. For bigger projects, you can now easily make sure that the instance of the class is only being initialized once and when you need to use the class in a new place, you now won’t feel guilty (excluding person life guilt, you’re stuck with that) for creating another global variable since now you can call “ClassNameHere”.getInstance() to get the same reference as you made 100 lines of code earlier.

As a refresher (since I know you may have skipped to the last paragraph), using singleton allows you to easily keep track of a sole instance of a class, and implemented in the way I stated (and in the way I made an example for), kind of hides the fact that pointers/references to a single object are being used. Also for some weird reason, if you want to only allow a certain number of instances of an object (I’m thinking probably most applicable to game development), you can easily create a static number to help keep track of how many instances of the object exists.