7. Making several exact copies of a complex object

Imagine that you have an object that you need to copy multiple times. Because it's not a primitive type, you cannot just copy the value of the whole thing into a new variable of the same type. Instead, you would need to instantiate a new object of this type and then copy the value of every field from the original object instance.

That would be fine for simple objects with a relatively small number of fields that contain primitive types, such as integers and Booleans. But what if you are dealing with complex objects with many fields, some of which contain other objects? And what if such an object contains private fields that you also want to be copied, but can't access?

In this case, you would need to write a complex code to make a copy of an object. For private fields in particular, you may also need to run additional logic to instantiate them. This way of doing things adds complexity; therefore it makes the code less readable and vulnerable to errors. Plus, as you will have no direct access to private members of the object, you may not necessarily end up with an exact copy and suffer some undesirable side-effects as a result.

Suitable design patterns

Prototype

Prototype is a design pattern that was created specifically for this problem.

If you have a complex object that is meant to be copied often, you can make it cloneable. And this is exactly what this design pattern enables you to do.

You would have an interface with a method that produces an instance of an object that implements this interface. Usually, such a method will be called `Clone`. And, if you want to make your object cloneable, you just implement this interface when you define the object type.

Inside this method, you will still need to copy the value of every field into the output object. However, this time, you will only have a single place in the entire code base to do it in – the object itself. So, it will be easier to have a close look at it and consider all edge cases. Likewise, you will have full access to the private members of the object, so your copy will be exact.

Why would you want to use Prototype

- All code to copy complex objects is located in one place.
- You can copy private members of the object.
- Any code that needs to generate a copy of the object will only need to call the `Clone` method without having to worry about the details of the cloning process.