Another reason to use a prototype would be because we cannot create a new class in the current scope of the code or because allowing a constructor on the class in the current scope violates the rules of encapsulation of our application. A situation like this could occur if the class's constructor was marked internal to a domain that is not the current domain. We could not call the constructor because of its encapsulation properties. This sometimes happens in cases where a facade is used. Since you cannot call the constructor outside of the facade's domain, the only way to construct a new instance of a class would be to provide a prototype method on the class.

The Prototype pattern has one main component: the Prototype. The prototype is really an ordinary class that has a method implemented to create a copy (or clone) of itself.

This is an interesting and useful pattern. Let's take a look at some examples of how it can be used.

![UML for Prototype pattern](image)

Figure 2-11. UML for Prototype pattern