Factory pattern

**Examples of swift factory pattern**

The factory pattern provides an interface for creating objects in a superclass. This article provides some details around this design pattern, and also has some examples of it! We should get building!

In programming, a factory is an object for creating other objects. In Swift that object will typically be a class, and will produce a concrete instance of an object

**Advantage of a factory pattern**

Instantiations may be complex, and encapsulating instantiation can simplify creating concrete instances by defining a single place. This means that we can follow the Dependency Inversion Principle, and even open the door to dependency injection.

**Decouple the use of an object from creating it**

We can abstract our code, so where modifications are made to a class the client of that class can continue to use it without further modification. By implementing to an interface we are provided with an abstraction rather than a concrete type, so code is protected from unwanted implementation details.

Modular expandability of the application  
good testability

Significant method names

**Disadvantages of a factory pattern**

High number of required classes,   
extension of the application is very elaborate