A Design Pattern: The Factory Method

One of the goals in object-oriented design is to delegate responsibility among different objects. This kind of distribution is good as it reflects two important features in object oriented methodology, encapsulation and delegation.

# 1. Problem

Often, we find ourselves in one of the following situations that needs a solution.

* An application (or framework) at runtime, is incapable of anticipating the class of object/instance. The application may know that it has to instantiate classes, but it only knows about abstract classes (or interfaces) that cannot instantiate.

PROBLEM: **The Application class only knows when it has to instantiate a new Object of an abstract class, but not the specific type of subclass.**

* A superclass may require its subclasses to specify the Objects to be created at the creation time.
* A class may delegate responsibility to one of several assisting subclasses so that services can be distributed locally to specific subclasses.

# 2. Solution: The Factory Method

The solution to the previous problem is a creational pattern, namely Factory Method.

A design pattern is a description or template for how to solve a problem that can be used in many different situations.

The Factory Method contains an interface which lets the subclasses decide at creation time which subclass to instantiate.

This pattern is named as a factory as it is responsible for *manufacturing an Object*. A factory does not make an Object. To make a new Object, we make a simple call to new. (Read “Java Review: Instantiation w/ the Keyword New”.)

However, the Factory pattern promotes loose coupling by eliminating the need to bind application-specific classes into the code. The use of factories in the code facilitates delegation of specific attributes of an object into specific subclasses. Therefore, the Factory pattern helps make a system independent of how objects are created.

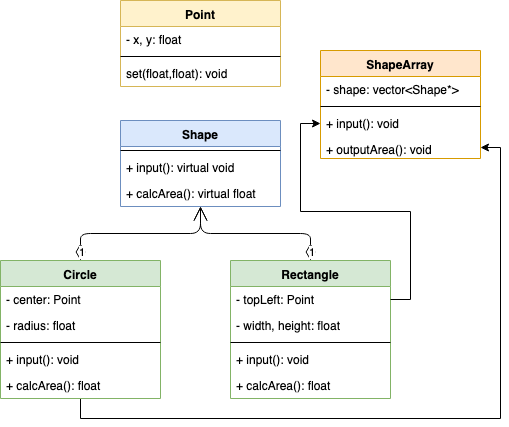
“The Factory Method lets a class defer instantiation to subclasses.”

# 3. Structure Diagram

*“Diagram credit: Object-oriented systems analysis and design by Noushin Ashrafi”*

# 4. Example with C++

In this example, upon receiving the user choice of geometric shape, either Circle or Rectangle, the factory interface will defer the instantiation to the associated subclass.



Sample code:

\*SampleCode\_FactoryMethod