Factory Design Pattern and Sequence Diagrams

UML SEQUENCE DIAGRAMS, FACTORY DESIGN PATTERNS

Perspective

Creating an abstraction of a target file reader allows you to separate your concrete implementation from the application.

We already know how to model this, and now we want to know an easy way to create a file reader.

To do so we will explore the Factory Design Pattern, a creational design pattern.

Creational

There is a set of design patterns that are specific to the construction of objects.

These patterns define the structure and relationships between:

- Abstract Types
- Concrete Types

The factory design pattern is to instantiate concrete types without any clients really caring about what they created.

A car factory builds a passenger car. We don’t care if it’s a Ford Fusion or Toyota Prius. It just drives us from point A to point B

A Target File factory builds a target file reader. We don’t care if it’s an XML file we are reading from or an INI file. We just care that we can read the data from the file.

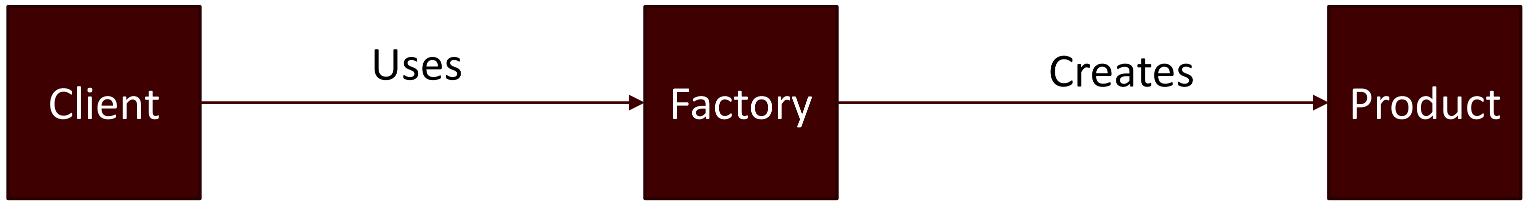
Factory Players

Product
- What is being built by the factory

Factory
- What is building products

Client
- Object that requests a product

Factory Visually



Your application

Creates concrete classes

Returns abstraction

The Factory Design Pattern…

Allows us to focus the construction of an object in one location

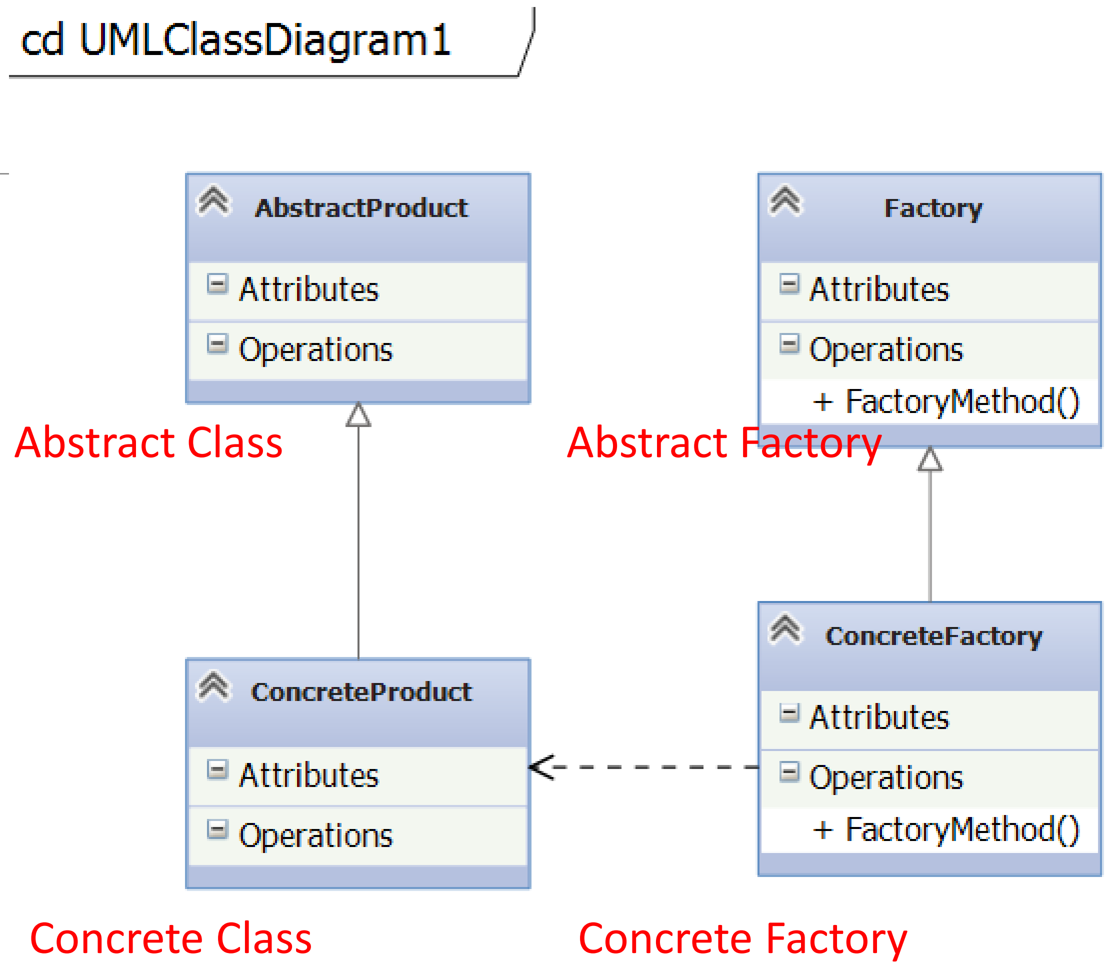
Allows us to create an object without specifying the exact object to be constructed.

Defines the interface for creating an object, while letting the classes that implement the interface decide which class to instantiate.

Gamma, et al., *Design Patterns: Elements of Reusable Object-Oriented Software*

Modeling in UML

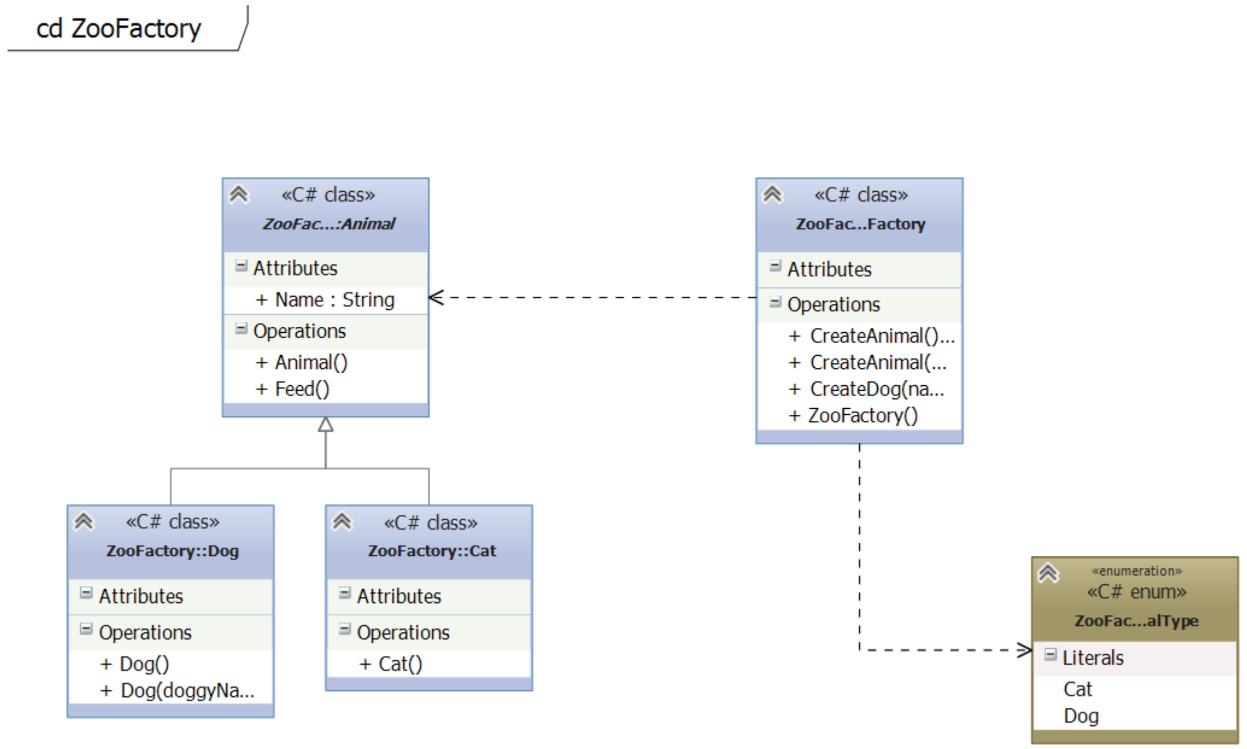
The factory design pattern looks like this:



Example: Zoo Factory Code

See example code: https://github.com/brianlamarche/Cpts323/FactoryDesignPattern

Example: Animals



See example code: https://github.com/brianlamarche/Cpts323/FactoryDesignPattern

Factory Pros and Cons

Pros

- Centralized instantiation of an object
- Decouples the application from the concrete types

Cons

- Sometimes the constructors require arguments. These may be set to defaults. If you are using an abstract class, this may be problematic
- Extending an object may require specialized initialization that was not foreseen

Considerations

- Don’t use for classes that do not extend a common base class
- Create factories for specific base classes
- Consider the possible extension of an object type
- Don’t overcomplicate things

Pairs great with other design patterns:

- Strategy

My $1.05

Factories can be useful for managing the construction of an object type when the constructors don’t vary much.

They are extremely useful for creating product families

- Algorithms
- File Readers