Factories

Design Patterns

The Factory Tour

1. A very simple factory
2. Factory Method
3. Abstract Factory

Motivating Example

Consider factories when:

- Unsure which concrete implementation of an interface I want to return
- Creation should be separated from representation of an object
- Lots of if/else blocks when deciding which concrete class to create
- Switch statements when deciding which concrete class to create

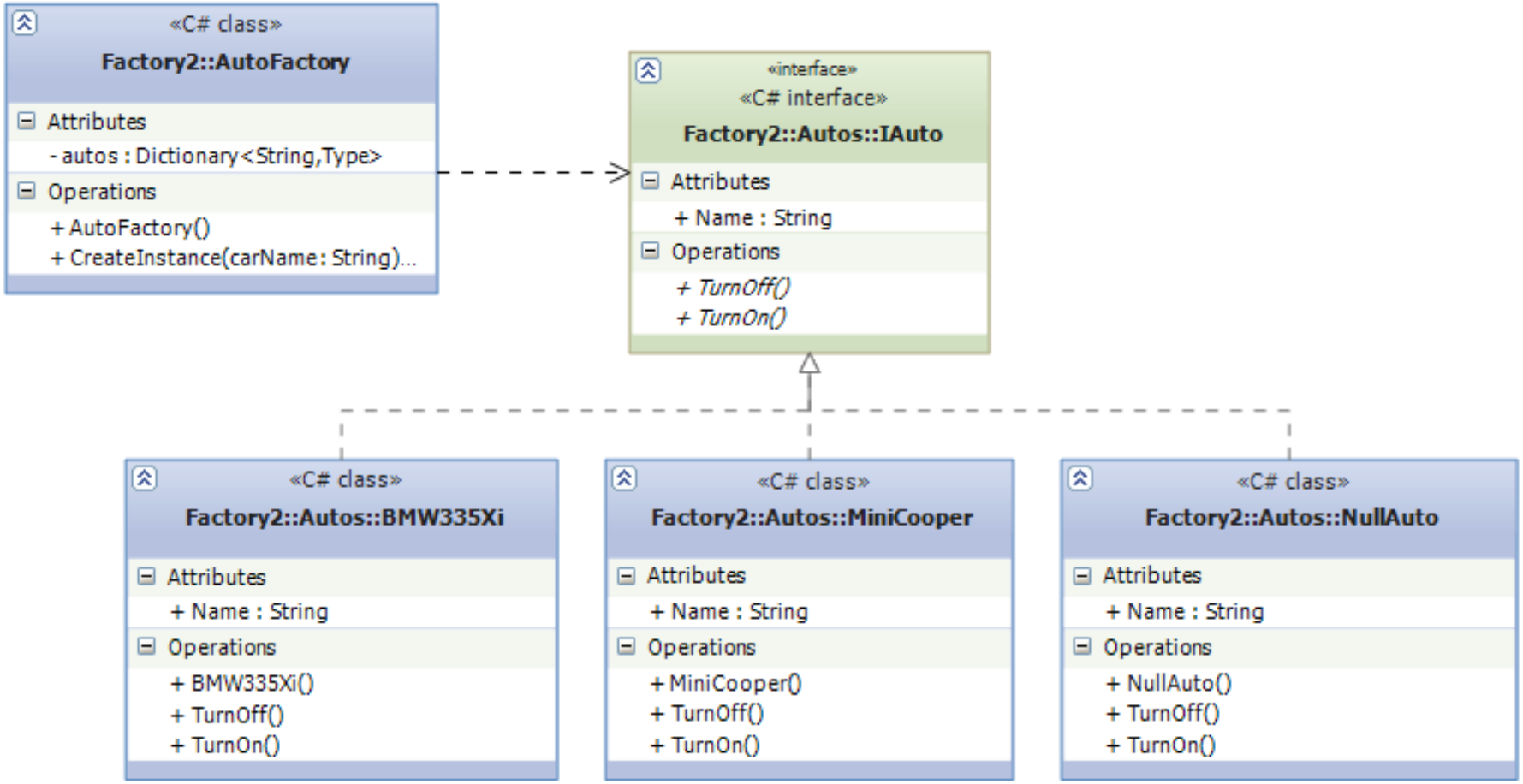
Intent

- **Separate object creation from the decision of which object to create**
- **Add new classes and functionality without breaking OCP**
  - Factory-produced objects
  - Factories themselves
- **Store which object to create outside of the program**
  - In a database
  - In configuration

Simple Factory

- **Encapsulate object creation**
- **Allows for late-bound decisions regarding instantiation**
  - configuration-based
  - other persistent storage
  - input or other dynamic data
- **Caller class knows what concrete factory it needs**

Simple Factory



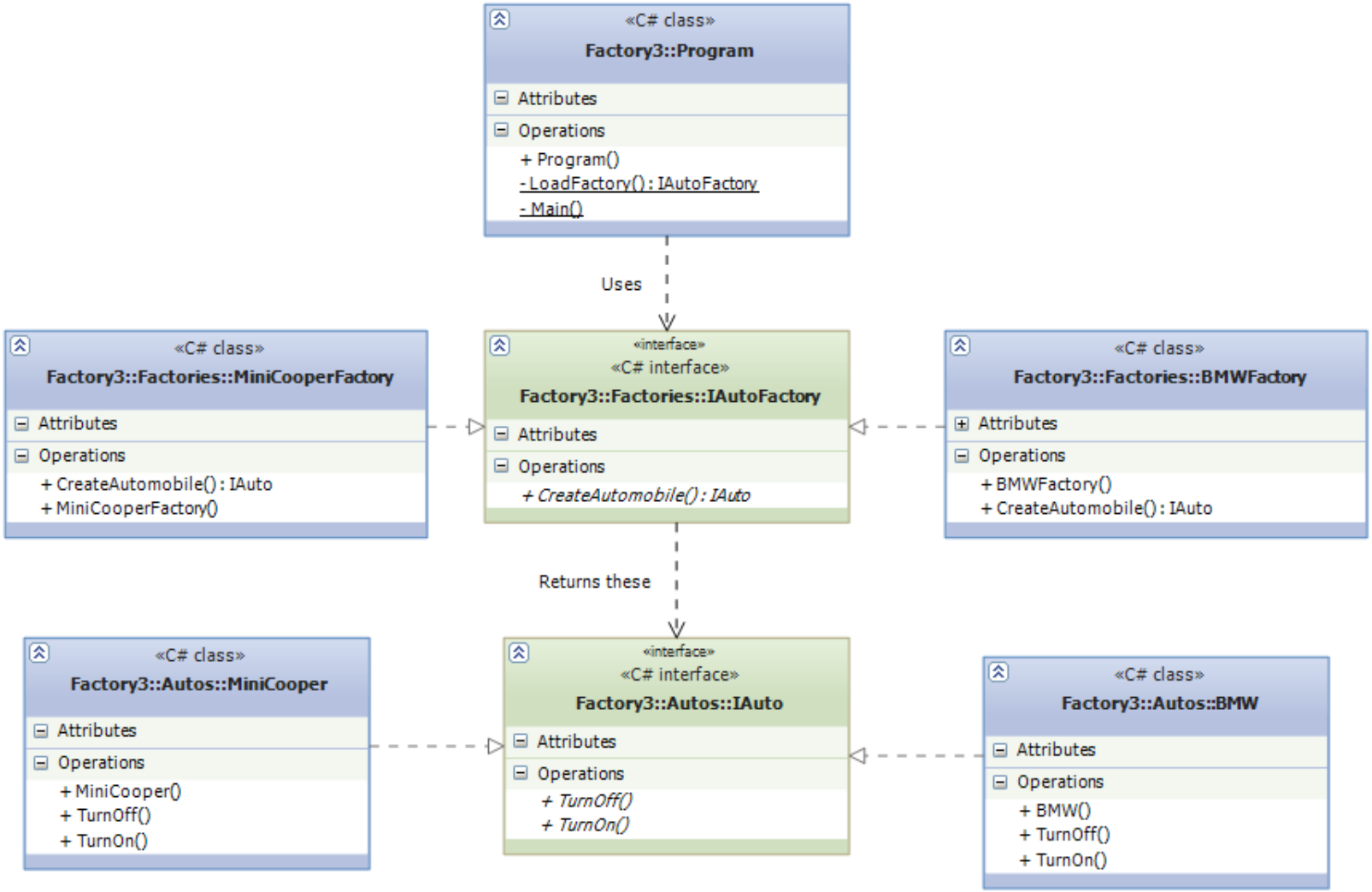
Factory Method

"Define an interface for creating an object, but let the subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses."

- Design Patterns

- **Adds an interface to the factory itself from Simple Factory**
- **Defers object creation to multiple factories that share an interface**
- **Derived classes implement or override the *factory method* of the base**

Factory Method



Factory Method

Advantage	Disadvantage

• Eliminate references to concrete classes
  • Factories
  • Objects created by factories
• Factories can be inherited to provide even more specialized object creation
• Rules for object initialization is centralized
	• May need to create a factory just to get a concrete class delivered
	• The inheritance hierarchy gets deeper with coupling between concrete factories and created classes

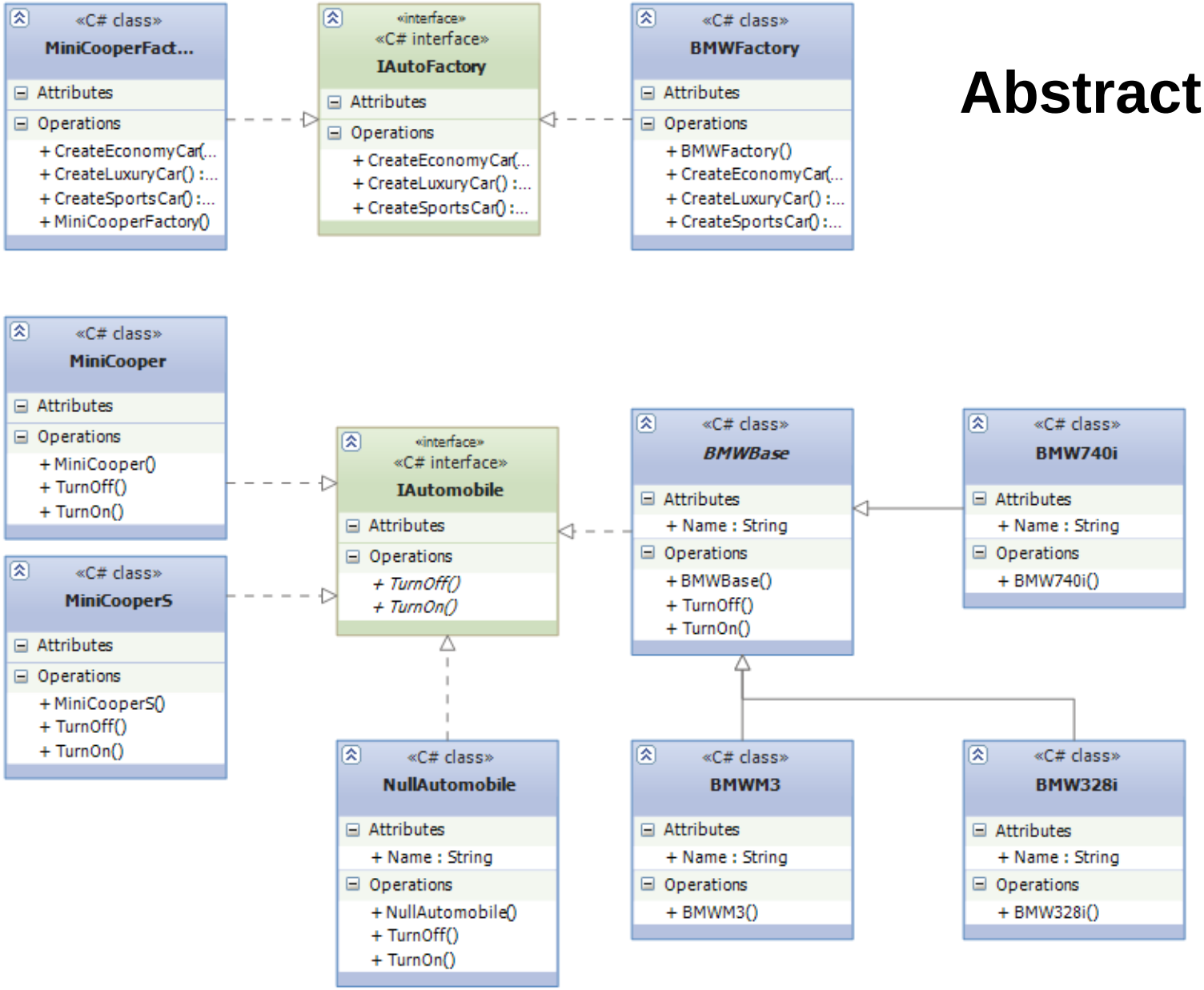
Abstract Factory

"Provide an interface for creating families of related or dependent objects without specifying their concrete classes."

– Design Patterns

- **Factories create different types of concrete objects (products)**
- **A Factory now represents a "family" of objects that it can create**
- **Factories may have more than one factory method**

Abstract Factory



Consequences

- Add new factories and classes without breaking OCP
- Defer choosing classes to classes that specialize in making that decision
- Using private or internal constructors hides direct construction with the new keyword

Known Uses of Factories

- Pluralsight.com
- .NET Framework

Related Materials

Other Creational Patterns

- **Singleton**
- **Builder**
- **Object Pool**
- **Prototype**

Related Subjects

- **NullObject**
- **Open Close Principle**

Summary

- **Varying levels of sophistication support different scenarios**
- **Factory Method**
  - A base class or interface defines the creation method
  - Subclasses implement the creation method in different ways
- **Abstract Factory**
  - All the attributes of Factory Method
  - Concrete factory classes may return various objects from a "family" of objects
- **Factories in .NET commonly make use of reflection**