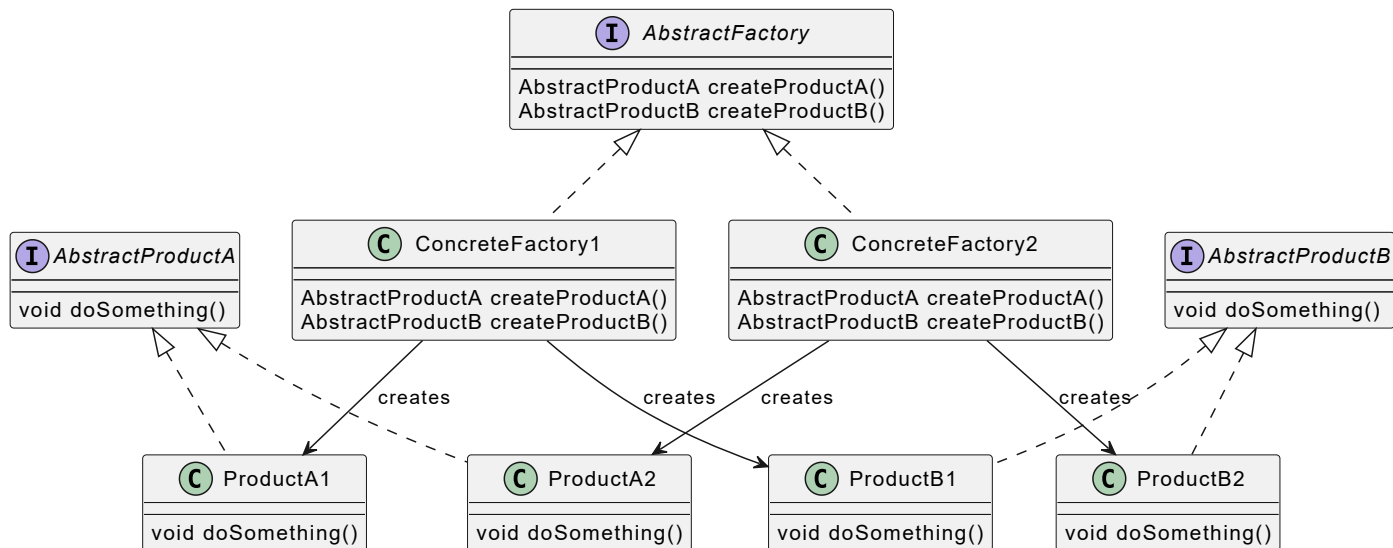# Creation Patterns

https://en.wikipedia.org/wiki/Software_design_pattern#Classification_and_list

Abstract Factory

Abstract factories are used to create a set of related objects without specifying their type.
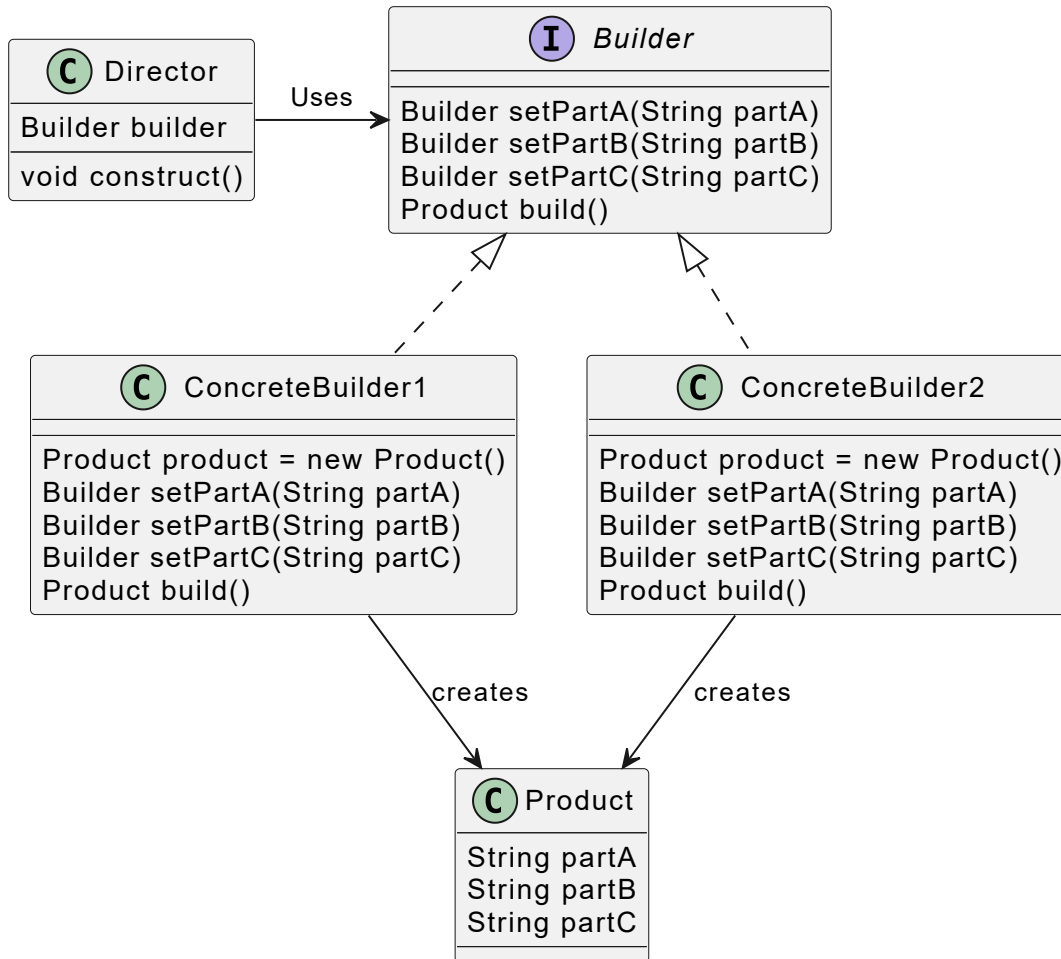
Create Pattern - Abstract Factory

Builder

Builder is used to construct an object instance with control flow

Create Pattern - Builder

```
┌──────────────────────┐         ┌───────────────────────────────────────┐
│ Ⓒ Director           │         │        Ⓘ  Builder                     │
├──────────────────────┤  Uses   ├───────────────────────────────────────┤
│ Builder builder      │────────▷│ Builder setPartA(String partA)        │
├──────────────────────┤         │ Builder setPartB(String partB)        │
│ void construct()     │         │ Builder setPartC(String partC)        │
└──────────────────────┘         │ Product build()                       │
                                 └───────────────────────────────────────┘
                                      △                        △
                                       ╲                      ╱
┌──────────────────────────────┐   ┌──────────────────────────────┐
│ Ⓒ  ConcreteBuilder1          │   │ Ⓒ  ConcreteBuilder2          │
├──────────────────────────────┤   ├──────────────────────────────┤
│ Product product = new Product()│  │ Product product = new Product()│
│ Builder setPartA(String partA)│   │ Builder setPartA(String partA)│
│ Builder setPartB(String partB)│   │ Builder setPartB(String partB)│
│ Builder setPartC(String partC)│   │ Builder setPartC(String partC)│
│ Product build()              │   │ Product build()              │
└──────────────────────────────┘   └──────────────────────────────┘
                   ╲ creates            creates ╱
                    ▼                          ▼
                  ┌──────────────────────┐
                  │ Ⓒ Product            │
                  ├──────────────────────┤
                  │ String partA         │
                  │ String partB         │
                  │ String partC         │
                  └──────────────────────┘
```

Dependency Injection

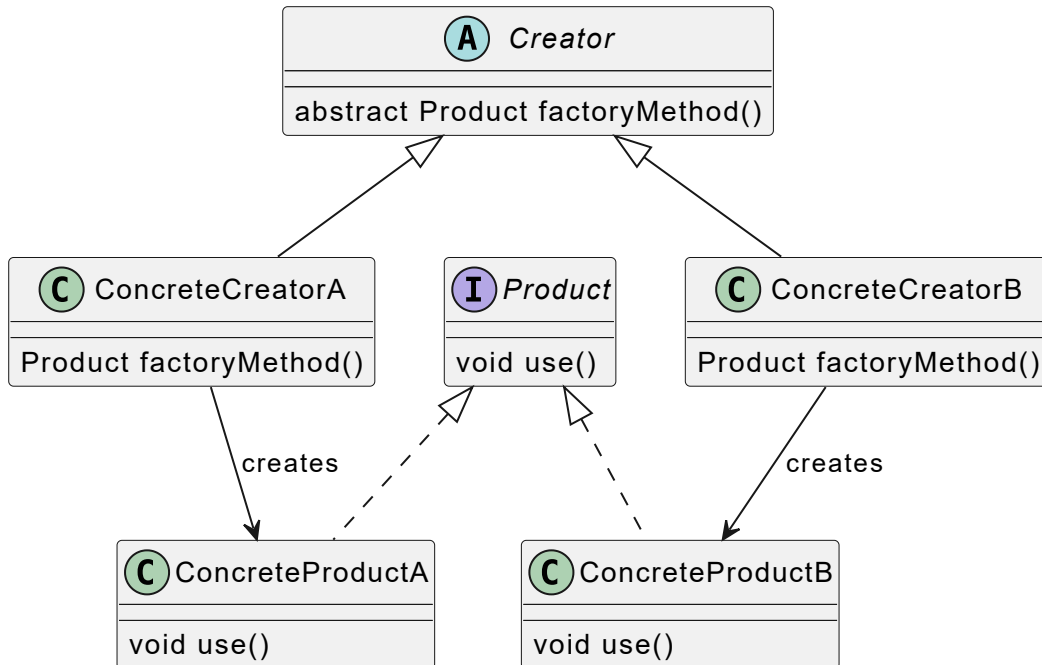Dependency Injection is a means to control/provide dependent functionality to a class

Create Pattern - Dependency Injection
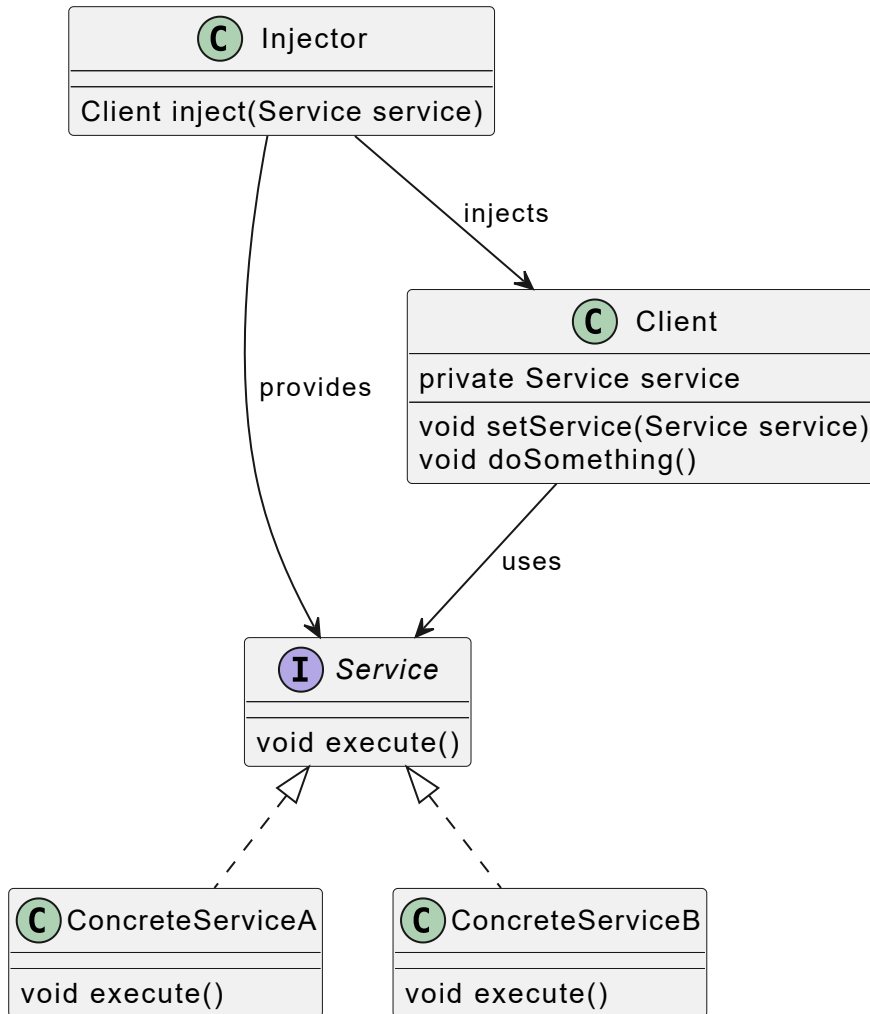
Factory

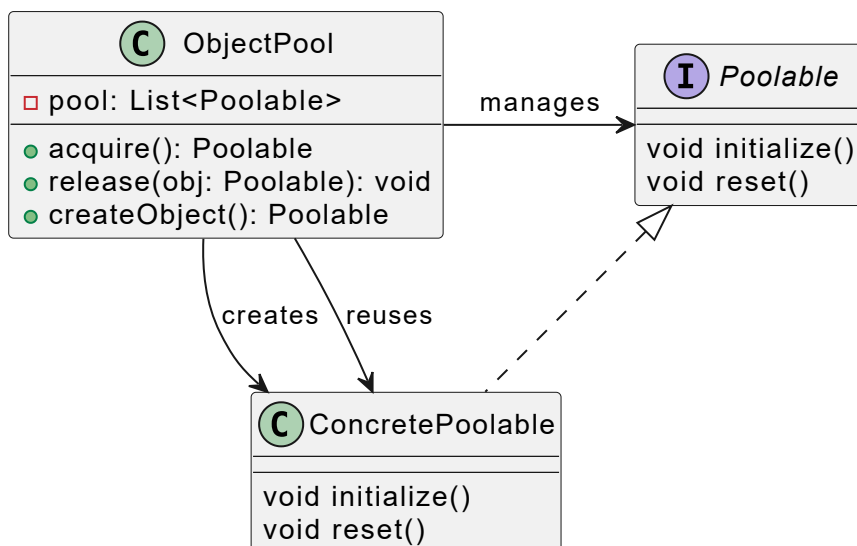Factories are used to create the particular type of class instance

Create Pattern - Factory

Pool

Object pools provide a means to reuse object instances that are expensive to create. Examples may include data service connection or threads.

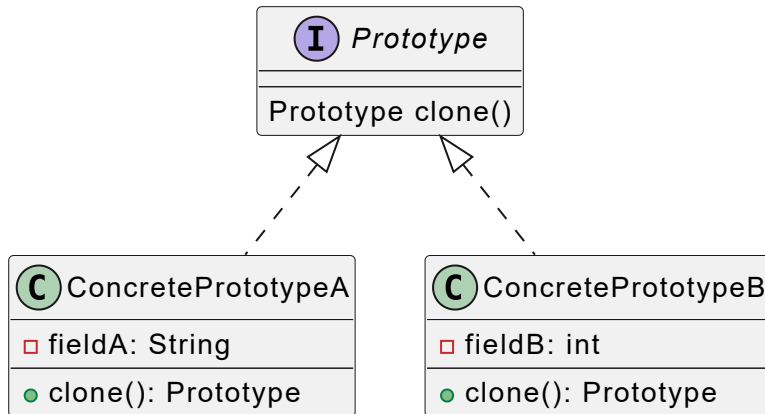**Create Pattern - Pool**

Prototype

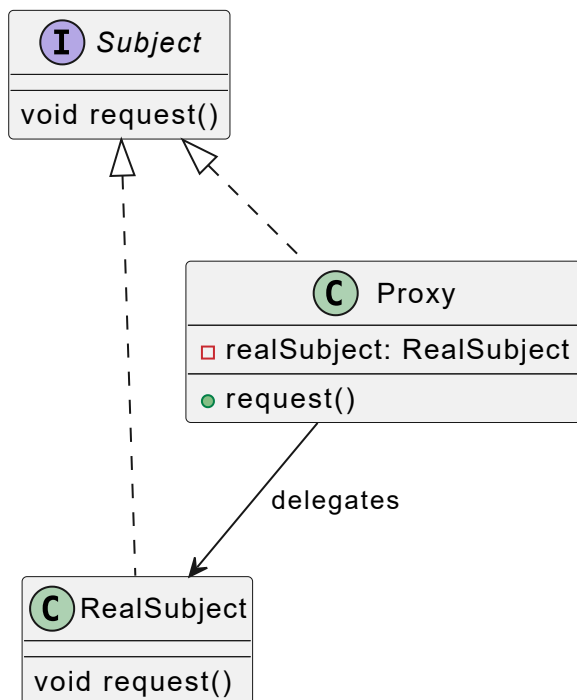Prototypes create instances by copying an existing object

Create Pattern - Prototype

```
┌──────────────────────┐
│   (I)  Prototype      │
├──────────────────────┤
│  Prototype clone()    │
└──────────────────────┘
```

```
┌──────────────────────────┐        ┌──────────────────────────┐
│ (C) ConcretePrototypeA   │        │ (C) ConcretePrototypeB   │
├──────────────────────────┤        ├──────────────────────────┤
│ □ fieldA: String         │        │ □ fieldB: int            │
├──────────────────────────┤        ├──────────────────────────┤
│ ● clone(): Prototype     │        │ ● clone(): Prototype     │
└──────────────────────────┘        └──────────────────────────┘
```

Proxy (Lazy Initialization)

Proxies or Lazy Initialized objects may be used for objects that are expensive to create. The actual creation of the object is not called until the first time the object is required.
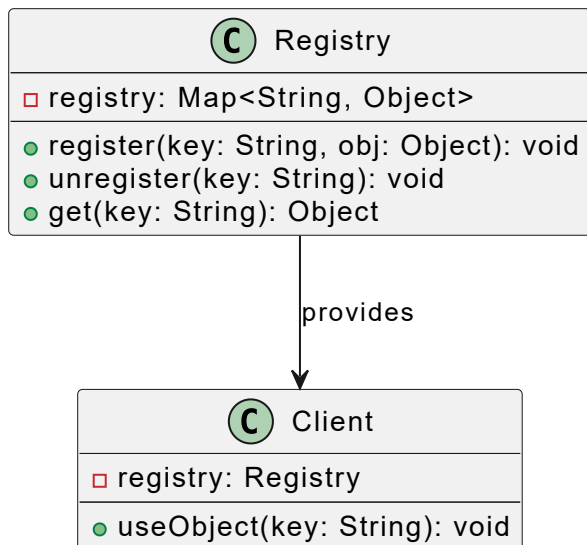
Create Pattern - Proxy

```
┌────────────────────┐
│  (I)  Subject      │
├────────────────────┤
│  void request()    │
└────────────────────┘
```

```
┌──────────────────────────────┐
│      (C)   Proxy             │
├──────────────────────────────┤
│ □ realSubject: RealSubject   │
├──────────────────────────────┤
│ ● request()                  │
└──────────────────────────────┘
```

delegates

```
┌────────────────────┐
│ (C) RealSubject    │
├────────────────────┤
│ void request()     │
└────────────────────┘
```

Registry (Multiton)

A registry allowed for storing/re-using a collect of named instances of an object.

Create Pattern - Registry

C Registry
□ registry: Map<String, Object>
● register(key: String, obj: Object): void ● unregister(key: String): void ● get(key: String): Object

provides

C Client
□ registry: Registry
● useObject(key: String): void

Singleton

Singletons are used to ensure only a single instance of an object is created in the lifetime of the application.

Create Pattern - Singleton

C Singleton
□ instance: Singleton
■ Singleton() // private constructor ● getInstance(): Singleton ● someMethod()