Factories

Sergej Kurakin

shopify | Oberlo

Most of us write code.

Most of us use
Frameworks.

Most of us write
Object-oriented Code.

Most of us want to create extensible and maintainable code.

One of the reasons we fail is...

```
new SomeClass();
```

But we have design patterns to solve that.

Software design pattern is a general, reusable solution to a commonly occurring problem.

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

Factory

- Factory
- Factory method pattern
- Abstract factory pattern

Factory patterns are creational patterns.

Other Creational Patterns

- Builder pattern
- Prototype pattern
- Singleton pattern*

Factory

“Factory is a function or method that returns objects of a varying prototype or class.”

https://en.wikipedia.org/wiki/Factory_(object-oriented_programming)

Method or function that returns new object.

```
class GreeterFactory {
  public function make(): Greeter
  {
    return new Greeter();
  }
}
```

```
function makeGreeter(): Greeter
{
  return new Greeter();
}
```

Parameterized factory.

This is up to developer
how to use such factory.

Factory method pattern

a.k.a. Virtual Constructor

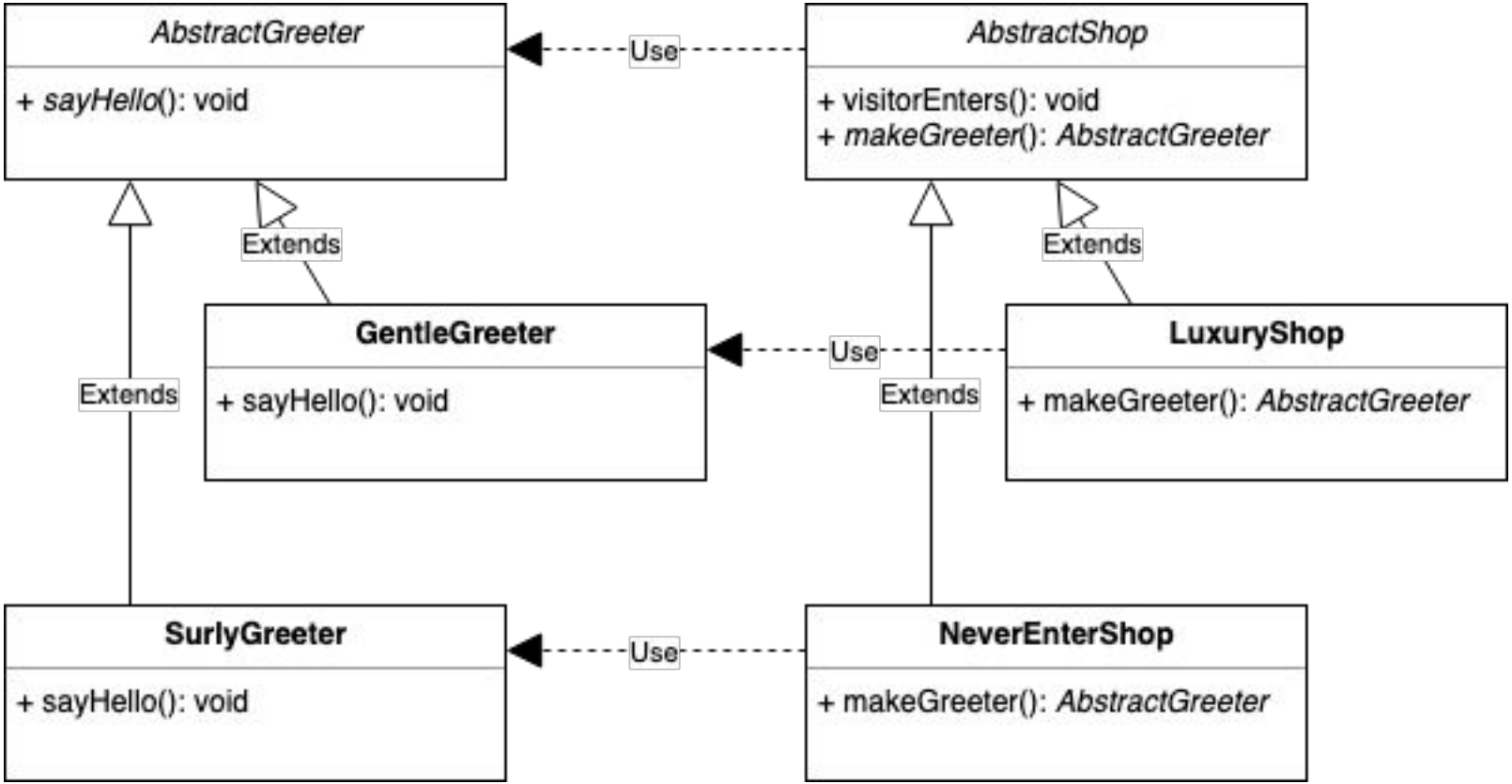
“Define an interface for creating an object, but let subclasses decide which class to instantiate.”

“Design Patterns: Elements of Reusable Object-Oriented Software” 1994

You define a class with “factory method”.

Also you define a subclass that implements “factory method”.

Every subclass returns new object type.


AbstractGreeter
+ sayHello(): void
Use
AbstractShop
+ visitorEnters(): void
+ makeGreeter(): AbstractGreeter
Extends
Extends
GentleGreeter
+ sayHello(): void
Use
LuxuryShop
+ makeGreeter(): AbstractGreeter
Extends
Extends
SurlyGreeter
+ sayHello(): void
Use
NeverEnterShop
+ makeGreeter(): AbstractGreeter

Abstract Factory Method in Abstract Class.

Concrete Factory Method in Abstract Class.

Parameterized factory method.

Provides hooks for subclasses.

When to use?

Abstract factory pattern

a.k.a. Kit

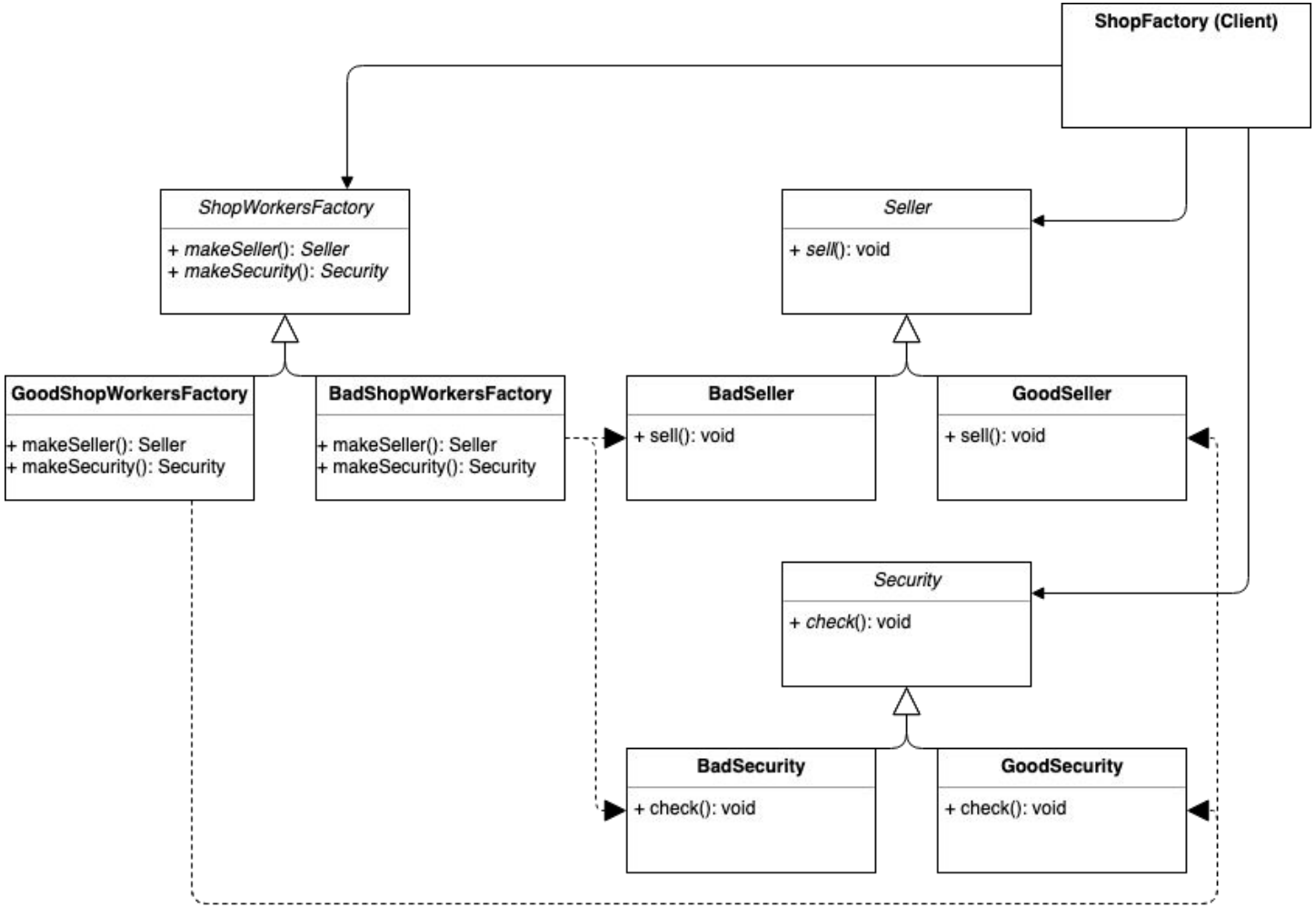
“Provide an interface for creating families of related or dependent objects without specifying their concrete classes.”

“Design Patterns: Elements of Reusable Object-Oriented Software” 1994

You define an abstraction with factory methods.

You create concrete implementations of factory methods.

You instantiate one concrete implementation by some rules.


ShopFactory (Client)
ShopWorkersFactory
+ makeSeller(): Seller
+ makeSecurity(): Security
Seller
+ sell(): void
GoodShopWorkersFactory
+ makeSeller(): Seller
+ makeSecurity(): Security
BadShopWorkersFactory
+ makeSeller(): Seller
+ makeSecurity(): Security
BadSeller
+ sell(): void
GoodSeller
+ sell(): void
Security
+ check(): void
BadSecurity
+ check(): void
GoodSecurity
+ check(): void

During execution you will need one “Concrete Factory”.

Parameterized factory methods are possible.

Isolates concrete classes.

Makes exchanging product families easy.

Promotes consistency among products.

Supporting new kinds of products is difficult.

When to use?

What about XXI century?

We widely use Factory but together with Dependency Containers.

Factory method pattern.

Abstract factory performs great together with Dependency Containers.

Discussion!

shopify | Oberlo

Factory (Wikipedia)
Factory Method Pattern (Wikipedia)
Abstract Factory Pattern (Wikipedia)

Factory Method Pattern (Refactoring Guru)
Abstract Factory Pattern (Refactoring Guru)

Design Patterns: Elements of Reusable Object-Oriented Software

Thanks!

shopify | Oberlo