

## 2. Factory

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

- move object creation logic out of client code
- let creation be decided at runtime

Why Factory?

→ Direct object creation causes problem:

Car car = new ~~Sedan~~ Sedan();

- client is tightly coupled to concrete class.
- adding new types requires modifying client code.
- violates OCP and DIP.

Factory Method solves

- tight coupling.
- if-else / switch object creation.
- hard to extend systems.

Factory Method decouples obj creation from usage.

## Bad Design

Class NotificationService {

Notification create (String type) {

if (type.equals("EMAIL")) return new EmailNotification();  
else if (type.equals("SMS")) return new SMSNotification();

return null;

• Every new notification → modify class.

• violates OCP.

## Factory method solution

Separate

What is created → Interface.

How is created → Factory.

Interface Notification

void notifyUser();

## Structure (V. Imp.)

- Product Interface.
- Concrete products
- Creator (Factory)
- Concrete creators.

class EmailNotification implements Notification {

public void notifyUser() {

System.out.println("sending Email");

class SMSNotification implements Notification {

public void notifyUser() {

System.out.println("sending SMS");

interface NotificationFactory {

Notification createNotification();

class EmailFactory implements NotificationFactory {

```
    public Notification createNotification() {
```

```
        return new EmailNotification();
```

class SmsFactory implements NotificationFactory {

```
    public Notification createNotification() {
```

```
        return new SMSNotification();
```

Client code :-

```
NotificationFactory factory = new EmailFactory();
```

```
Notification notification = factory.createNotification();
```

```
notification.notifyUser();
```

Factory analogy :-

- Client orders a product.
- Factor decides how to build it.
- Client doesn't care about construction steps.

When to use :-

- Object creation logic is complex.
- System needs to be extensible.
- You see large if-else creation logic.

Factory Method defines an interface for object creation and allows subclasses to decide which concrete class to instantiate, helping us follow OCP and DIP.