3. Abstract Factory.

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

Factory Method → creates one product.
Abstract Factory → creates a group (family) of related products.

Problem without Abstract Factory

Inconsistent Object Creation.

```
Button button = new WindowsButton();
Checkbox checkbox = new MacCheckbox();   X mismatch.
```

- No UI consistency.
- No guarantee of compatibility.
- Client controls too much creation logic.

Step 1:

```
interface Button {
    void render();
}
interface CheckBox {
    void render();
}
```

Step 2: Concrete Products (Windows).

```
class WindowsButton implements Button {
    public void render() {
        S.o.Pln ("Windows Button");
    }
}
class WindowsCheckBox implements CheckBox {
    public void render() {
        S.o.Pln ("Windows CheckBox");
    }
}
```

Step3:- Concrete Products (Mac).

```
class MacButton implements Button {
    public void render() {
        System.out.println("Mac Button");
    }
}

class MacCheckBox implements Checkbox {
    public void render() {
        System.out.println("Mac CheckBox");
    }
}
```

Abstract Factory:

```
interface UIFactory {
    Button CreateButton();
    CheckBox CreateCheckbox();
}

class WindowsFactory implements UIFactory {
    public Button createButton() {
        return new WindowsButton();
    }
    public CheckBox createCheckBox() {
        return new WindowsCheckBox();
    }
}

class MacFactory implements UIFactory {
    public Button createButton() {
        return new MacButton();
    }
    public CheckBox createCheckBox() {
        return new MacCheckBox();
    }
}
```

Client code:

```
UIFactory factory = new WindowsFactory();
Button button = factory.CreateButton();
Checkbox checkbox = factory.CreateCheckBox();
```