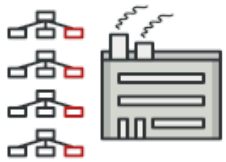

Абстрактная фабрика на Java

Абстрактная фабрика — это порождающий паттерн проектирования, который решает проблему создания целых семейств связанных продуктов, без указания конкретных классов продуктов.

Абстрактная фабрика задаёт интерфейс создания всех доступных типов продуктов, а каждая конкретная реализация фабрики порождает продукты одной из вариаций. Клиентский код вызывает методы фабрики для получения продуктов, вместо самостоятельного создания с помощью оператора `new`. При этом фабрика сама следит за тем, чтобы создать продукт нужной вариации.

Подробней о паттерне Абстрактная фабрика →

Навигация

MacOSFactory
WindowsFactory
app
Application
Demo
OutputDemo

Сложность: ★★☆

Популярность: ★★★

Применимость: Паттерн можно часто встретить в Java-коде, особенно там, где требуется создание семейств продуктов (например, внутри фреймворков).

Примеры Абстрактной фабрики в стандартных библиотеках Java:

- `javax.xml.parsers.DocumentBuilderFactory#newInstance()`
- `javax.xml.transform.TransformerFactory#newInstance()`
- `javax.xml.xpath.XPathFactory#newInstance()`

Признаки применения паттерна: Паттерн можно определить по методам, возвращающим фабрику, которая, в свою очередь, используется для создания конкретных продуктов, возвращая их через абстрактные типы или интерфейсы.

Производство семейств кросс-платформенных элементов GUI

В этом примере в роли двух семейств продуктов выступают кнопки и чекбоксы. Оба семейства продуктов имеют одинаковые вариации: для работы под MacOS и Windows.

Абстрактная фабрика задаёт интерфейс создания продуктов всех семейств. Конкретные фабрики создают различные продукты одной вариации (MacOS или Windows).

Клиенты фабрики работают как с фабрикой, так и с продуктами только через абстрактные интерфейсы. Благодаря этому, один и тот же клиентский код может работать с различными фабриками и продуктами.

buttons: Первая иерархия продуктов (кнопки)

buttons/Button.java

```
package refactoring_guru.abstract_factory.example.buttons;

/**
 * Паттерн предполагает, что у вас есть несколько семейств продуктов,
 * находящихся в отдельных иерархиях классов (Button/Checkbox). Продукты одного
 * семейства должны иметь общий интерфейс.
 *
 * Это — общий интерфейс для семейства продуктов кнопок.
 */
public interface Button {
    void paint();
}
```

buttons/MacOSButton.java

```
package refactoring_guru.abstract_factory.example.buttons;

/**
 * Все семейства продуктов имеют одни и те же вариации (MacOS/Windows).
 *
 * Это вариант кнопки под MacOS.
 */
public class MacOSButton implements Button {

    @Override
    public void paint() {
        System.out.println("You have created MacOSButton.");
    }
}
```

buttons/WindowsButton.java

```
package refactoring_guru.abstract_factory.example.buttons;

/**
 * Все семейства продуктов имеют одни и те же вариации (MacOS/Windows).
 *
 * Это вариант кнопки под Windows.
 */
public class WindowsButton implements Button {

    @Override
```

```
        public void paint() {
            System.out.println("You have created WindowsButton.");
        }
    }
```

checkboxes: Вторая иерархия продуктов (чекбоксы)

checkboxes/Checkbox.java

```
package refactoring_guru.abstract_factory.example.checkboxes;

/**
 * Чекбоксы — это второе семейство продуктов. Оно имеет те же вариации, что и
 * кнопки.
 */
public interface Checkbox {
    void paint();
}
```

checkboxes/MacOSCheckbox.java

```
package refactoring_guru.abstract_factory.example.checkboxes;

/**
 * Все семейства продуктов имеют одинаковые вариации (MacOS/Windows).
 *
 * Вариация чекбокса под MacOS.
 */
public class MacOSCheckbox implements Checkbox {

    @Override
    public void paint() {
        System.out.println("You have created MacOSCheckbox.");
    }
}
```

checkboxes/WindowsCheckbox.java

```
package refactoring_guru.abstract_factory.example.checkboxes;

/**
```

```
 * Все семейства продуктов имеют одинаковые вариации (MacOS/Windows).
 *
 * Вариация чекбокса под Windows.
 */
public class WindowsCheckbox implements Checkbox {

    @Override
    public void paint() {
        System.out.println("You have created WindowsCheckbox.");
    }
}
```

factories

factories/GUIFactory.java: Абстрактная фабрика

```
package refactoring_guru.abstract_factory.example.factories;

import refactoring_guru.abstract_factory.example.buttons.Button;
import refactoring_guru.abstract_factory.example.checkboxes.Checkbox;

/**
 * Абстрактная фабрика знает обо всех (абстрактных) типах продуктов.
 */
public interface GUIFactory {
    Button createButton();
    Checkbox createCheckbox();
}
```

factories/MacOSFactory.java: Конкретная фабрика (MacOS)

```
package refactoring_guru.abstract_factory.example.factories;

import refactoring_guru.abstract_factory.example.buttons.Button;
import refactoring_guru.abstract_factory.example.buttons.MacOSButton;
import refactoring_guru.abstract_factory.example.checkboxes.Checkbox;
import refactoring_guru.abstract_factory.example.checkboxes.MacOSCheckbox;

/**
 * Каждая конкретная фабрика знает и создаёт только продукты своей вариации.
 */
public class MacOSFactory implements GUIFactory {

    @Override
```

```
        public Button createButton() {
            return new MacOSButton();
        }

        @Override
        public Checkbox createCheckbox() {
            return new MacOSCheckbox();
        }
    }
```

factories/WindowsFactory.java: Конкретная фабрика (Windows)

```
    package refactoring_guru.abstract_factory.example.factories;

    import refactoring_guru.abstract_factory.example.buttons.Button;
    import refactoring_guru.abstract_factory.example.buttons.WindowsButton;
    import refactoring_guru.abstract_factory.example.checkboxes.Checkbox;
    import refactoring_guru.abstract_factory.example.checkboxes.WindowsCheckbox;

    /**
     * Каждая конкретная фабрика знает и создаёт только продукты своей вариации.
     */
    public class WindowsFactory implements GUIFactory {

        @Override
        public Button createButton() {
            return new WindowsButton();
        }

        @Override
        public Checkbox createCheckbox() {
            return new WindowsCheckbox();
        }
    }
```

app

app/Application.java: Клиентский код

```
    package refactoring_guru.abstract_factory.example.app;

    import refactoring_guru.abstract_factory.example.buttons.Button;
    import refactoring_guru.abstract_factory.example.checkboxes.Checkbox;
    import refactoring_guru.abstract_factory.example.factories.GUIFactory;
```

```
/**
 * Код, использующий фабрику, не волнует с какой конкретно фабрикой он работает.
 * Все получатели продуктов работают с продуктами через абстрактный интерфейс.
 */
public class Application {
    private Button button;
    private Checkbox checkbox;

    public Application(GUIFactory factory) {
        button = factory.createButton();
        checkbox = factory.createCheckbox();
    }

    public void paint() {
        button.paint();
        checkbox.paint();
    }
}
```

Demo.java: Конфигуратор приложения

```
package refactoring_guru.abstract_factory.example;

import refactoring_guru.abstract_factory.example.app.Application;
import refactoring_guru.abstract_factory.example.factories.GUIFactory;
import refactoring_guru.abstract_factory.example.factories.MacOSFactory;
import refactoring_guru.abstract_factory.example.factories.WindowsFactory;

/**
 * Демо-класс. Здесь всё сводится воедино.
 */
public class Demo {

    /**
     * Приложение выбирает тип и создаёт конкретные фабрики динамически исходя
     * из конфигурации или окружения.
     */
    private static Application configureApplication() {
        Application app;
        GUIFactory factory;
        String osName = System.getProperty("os.name").toLowerCase();
        if (osName.contains("mac")) {
            factory = new MacOSFactory();
        } else {
            factory = new WindowsFactory();
        }
        app = new Application(factory);
```

```
            return app;
        }

        public static void main(String[] args) {
            Application app = configureApplication();
            app.paint();
        }
    }
```

OutputDemo.txt: Результат выполнения

```
You create WindowsButton.
You created WindowsCheckbox.
```

ЧИТАЕМ ДАЛЬШЕ

Строитель на Java →

Главная
Рефакторинг
Паттерны
Премиум контент
Форум
Связаться

Хмельницкое шоссе 19 / 27, Каменец-Подольский, Украина, 32305
Email: support@refactoring.guru

Условия использования

Политика конфиденциальности

Использование контента