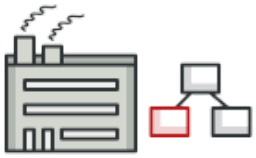

Фабричный метод на Java

Фабричный метод — это порождающий паттерн проектирования, который решает проблему создания различных продуктов, без указания конкретных классов продуктов.

Фабричный метод задаёт метод, который следует использовать вместо вызова оператора `new` для создания объектов-продуктов. Подклассы могут переопределить этот метод, чтобы изменять тип создаваемых продуктов.

Подробней о паттерне Фабричный метод →

Навигация

Сложность: ★☆☆

Популярность: ★★★

Применимость: Паттерн можно часто встретить в любом Java-коде, где требуется гибкость при создании продуктов.

Паттерн широко используется в стандартных библиотеках Java:

- `java.util.Calendar#getInstance()`
- `java.util.ResourceBundle#getBundle()`
- `java.text.NumberFormat#getInstance()`
- `java.nio.charset.Charset#forName()`
- `java.net.URLStreamHandlerFactory#createURLStreamHandler(String)` (Возвращает разные объекты-одиночки, в зависимости от протокола)
- `java.util.EnumSet#of()`
- `javax.xml.bind.JAXBContext#createMarshaller()` и другие похожие методы.

Признаки применения паттерна: Фабричный метод можно определить по создающим методам, которые возвращают объекты продуктов через абстрактные типы или интерфейсы. Это позволяет переопределять типы создаваемых продуктов в подклассах.

Производство кросс-платформенных элементов GUI

В этом примере в роли продуктов выступают кнопки, а в роли создателя — диалог.

Разным типам диалогов соответствуют свои собственные типы элементов. Поэтому для каждого типа диалога мы создаём свой подкласс и переопределяем в нём фабричный метод.

Каждый конкретный диалог будет порождать те кнопки, которые к нему подходят. При этом базовый код диалогов не сломается, так как он работает с продуктами только через их общий интерфейс.

buttons

buttons/Button.java: Общий интерфейс кнопок

```
package refactoring_guru.factory_method.example.buttons;

/**
 * Общий интерфейс для всех продуктов.
 */
public interface Button {
    void render();
    void onClick();
}
```

buttons/HtmlButton.java: Конкретный класс кнопок

```
package refactoring_guru.factory_method.example.buttons;

/**
 * Реализация HTML кнопок.
 */
public class HtmlButton implements Button {

    public void render() {
        System.out.println("<button>Test Button</button>");
        onClick();
    }

    public void onClick() {
        System.out.println("Click! Button says - 'Hello World!'");
    }
}
```

buttons/WindowsButton.java: Ещё один класс кнопок

```
package refactoring_guru.factory_method.example.buttons;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Реализация нативных кнопок операционной системы.
 */
public class WindowsButton implements Button {
    JPanel panel = new JPanel();
    JFrame frame = new JFrame();
```

```
    JButton button;

    public void render() {
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Hello World!");
        label.setOpaque(true);
        label.setBackground(new Color(235, 233, 126));
        label.setFont(new Font("Dialog", Font.BOLD, 44));
        label.setHorizontalAlignment(SwingConstants.CENTER);
        panel.setLayout(new FlowLayout(FlowLayout.CENTER));
        frame.getContentPane().add(panel);
        panel.add(label);
        onClick();
        panel.add(button);

        frame.setSize(320, 200);
        frame.setVisible(true);
        onClick();
    }

    public void onClick() {
        button = new JButton("Exit");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                frame.setVisible(false);
                System.exit(0);
            }
        });
    }
}
```

factory

factory/Dialog.java: Базовый диалог

```
package refactoring_guru.factory_method.example.factory;

import refactoring_guru.factory_method.example.buttons.Button;

/**
 * Базовый класс фабрики. Заметьте, что "фабрика" - это всего лишь
 * дополнительная роль для класса. Он уже имеет какую-то бизнес-логику, в
 * которой требуется создание разнообразных продуктов.
 */
public abstract class Dialog {

    public void renderWindow() {
```

```
        // ... остальной код диалога ...

        Button okButton = createButton();
        okButton.render();
    }

    /**
     * Подклассы будут переопределять этот метод, чтобы создавать конкретные
     * объекты продуктов, разные для каждой фабрики.
     */
    public abstract Button createButton();
}
```

factory/HtmlDialog.java: Конкретный класс диалогов

```
package refactoring_guru.factory_method.example.factory;

import refactoring_guru.factory_method.example.buttons.Button;
import refactoring_guru.factory_method.example.buttons.HtmlButton;

/**
 * HTML-диалог.
 */
public class HtmlDialog extends Dialog {

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

factory/WindowsDialog.java: Ещё один класс диалогов

```
package refactoring_guru.factory_method.example.factory;

import refactoring_guru.factory_method.example.buttons.Button;
import refactoring_guru.factory_method.example.buttons.WindowsButton;

/**
 * Диалог на элементах операционной системы.
 */
public class WindowsDialog extends Dialog {

    @Override
```

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

Demo.java: Клиентский код

```
package refactoring_guru.factory_method.example;

import refactoring_guru.factory_method.example.factory.Dialog;
import refactoring_guru.factory_method.example.factory.HtmlDialog;
import refactoring_guru.factory_method.example.factory.WindowsDialog;

/**
 * Демо-класс. Здесь всё сводится воедино.
 */
public class Demo {
    private static Dialog dialog;

    public static void main(String[] args) {
        configure();
        runBusinessLogic();
    }

    /**
     * Приложение создаёт определённую фабрику в зависимости от конфигурации или
     * окружения.
     */
    static void configure() {
        if (System.getProperty("os.name").equals("Windows 10")) {
            dialog = new WindowsDialog();
        } else {
            dialog = new HtmlDialog();
        }
    }

    /**
     * Весь остальной клиентский код работает с фабрикой и продуктами только
     * через общий интерфейс, поэтому для него неважно какая фабрика была
     * создана.
     */
    static void runBusinessLogic() {
        dialog.renderWindow();
    }
}
```

OutputDemo.txt: Результат с фабрикой HtmlDialog

```
<button>Test Button</button>
Click! Button says - 'Hello World!'
```

OutputDemo.png: Результат с фабрикой WindowsDialog