УДК 004

Использование Factory Method для реализации интерфейс формы на Java

Андриенко Иван Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье представлено эффективное использование шаблона проектирования Factory Method для создания гибкого и расширяемого интерфейса форм на языке программирования Java. Рассмотрены основные концепции, включая интерфейс Form, а также абстрактный класс с фабричным методом. Каждая конкретная фабрика реализует фабричный метод для создания соответствующей формы. Приведен пример использования шаблона для управления процессом создания объектов форм.
Ключевые слова: Factory Method, Java, интерфейс формы, объекты формы, дизайн паттерн.

Using the Factory Method to implement the form interface in Java

Andrienko Ivan Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article presents the effective use of the Factory Method design pattern to create a flexible and extensible form interface in the Java programming language. The basic concepts are considered, including the Form interface, as well as an abstract class with a factory method. Each specific factory implements a factory method to create the appropriate form. An example of using a template to control the process of creating form objects is given.
Keywords: Factory Method, Java, form interface, form objects, design pattern.

1 Введение

1.1 Актуальность

Использование шаблона проектирования Factory Method в разработке интерфейсов форм на платформе Java актуально благодаря своей способности обеспечивать гибкость и расширяемость приложений. Этот подход улучшает структуру кода, облегчает его сопровождение, разделяет ответственности между компонентами, повышает тестируемость и обеспечивает поддержку различных типов форм. Factory Method способствует созданию модульных, масштабируемых и устойчивых к изменениям приложений, что является важным фактором в современной разработке программного обеспечения.

1.2 Обзор исследований

В своей работе С.Г. Чёрный рассмотрел возможность использования сегментации кода для проектирования модуля информационной системы в структуре базы данных. Предложено использование технологий рефакторинга для улучшения и оптимизации программных модулей [1]. В.Л. Волушкова рассмотрел технологии программирования на примере языка java. Многие программисты решают идентичные задачи и находят похожие решения. Для того, чтобы не создавать проблем при проектировании, можно воспользоваться уже готовыми решениями – фреймворками и шаблонами проектирования. В книге приведены описание и примеры использования фреймворка Spring для реализации доступа к данным [2]. В своей работе В.Я. Израилев описывал паттерны проектирования. Проводится обзор существенных дополнений нового стандарта языка программирования C++. Описаны объективные метрики измерения кода [3]. В своей работе Е.А. Крайнова представил тенденции развития технологий программирования, современные парадигмы программирования. Дано понятие объектно-ориентированного программирования, паттернов проектирования на его основе. Структурированы наиболее часто применяемые паттерны проектирования систем и дана их характеристика [4]. В статье Д.А. Кравченко рассмотрел порождающие паттерны проектирования для объектно-ориентированного языка программирования Java, случаи их использования и проблемы, решающиеся с их помощью [5].

1.3 Цель исследования

Цель исследования – Изучить и продемонстрировать эффективное применение шаблона проектирования Factory Method для реализации гибкого и расширяемого интерфейса форм на языке программирования Java.

2 Материалы и методы

Реализация шаблона проектирования Factory Method для интерфейсов форм выполнена на Java, используя среду программирования IntelliJ IDEA.

3 Результаты и обсуждения

Использование шаблона проектирования Factory Method обосновано несколькими важными преимуществами, которые содействуют лучшей организации и гибкости кода в процессе разработки программного обеспечения:

- Гибкость и расширяемость: Factory Method обеспечивает гибкую архитектуру, позволяя добавлять новые типы объектов (в данном случае, форм) без изменения существующего кода. Это существенно упрощает внесение изменений и расширение функциональности приложения.
- Разделение ответственностей: Шаблон позволяет выделить процесс создания объектов в отдельный компонент (фабрику), что способствует разделению ответственностей между различными частями кода.

Это соответствует принципу единственной ответственности и делает код более модульным.

- Повышение читаемости и сопровождаемости: Фабричный метод делает структуру кода более ясной и читаемой. Каждая конкретная фабрика ответственна за создание конкретного типа объекта, что облегчает понимание и сопровождение кода.
- Возможность поддержки различных типов объектов: Factory Method позволяет динамически создавать различные типы объектов в зависимости от контекста или условий, что особенно полезно, например, при работе с разными формами в графическом интерфейсе приложения.
- Тестирование и поддержка кода: Код, основанный на Factory Method, обычно более легко тестируется, так как каждая фабрика может быть протестирована отдельно. Это способствует созданию более надежных и поддерживаемых приложений.

Первым шагом реализации Factory Method будет определение интерфейса Form, который служит абстракцией для всех типов форм. Интерфейс содержит один метод display(), который будет реализован в конкретных примерах для отображения информации о типе формы. Далее реализуем класс LoginForm. Он реализует интерфейс, предоставляя конкретную реализацию метода display(). В данном случае, метод display() просто выводит строку, сообщающую о том, что это форма входа в систему. Аналогично реализуем класс для регистрации.

Main.java

```
7 usages  2 implementations
interface Form {
    2 usages  2 implementations
    void display();
}

1 usage
class LoginForm implements Form {
    2 usages
    @Override
    public void display() {
        System.out.println("Форма входа в систему");
    }
}

1 usage
class RegistrationForm implements Form {
    2 usages
    @Override
    public void display() {
        System.out.println("форма регистрации");
    }
}
```

Рисунок 1 – Определение интерфейса Form и реализация классов

Далее создаем абстрактный класс. Этот класс представляет собой абстрактную фабрику, которая содержит фабричный метод createForm().

Фабричный метод объявлен абстрактным, что означает, что конкретные реализации этого метода будут предоставлены в подклассах. Абстрактный класс также может содержать другие методы, предназначенные для работы с созданными формами.

После создания абстрактного класса переходим к реализации фабричного метода для создания формы входа LoginFormFactory. Этот класс наследуется от абстрактного класса и предоставляет конкретную реализацию фабричного метода createForm(). В данной реализации метод просто создает и возвращает новый объект формы входа. Аналогично создаем форму регистрации.

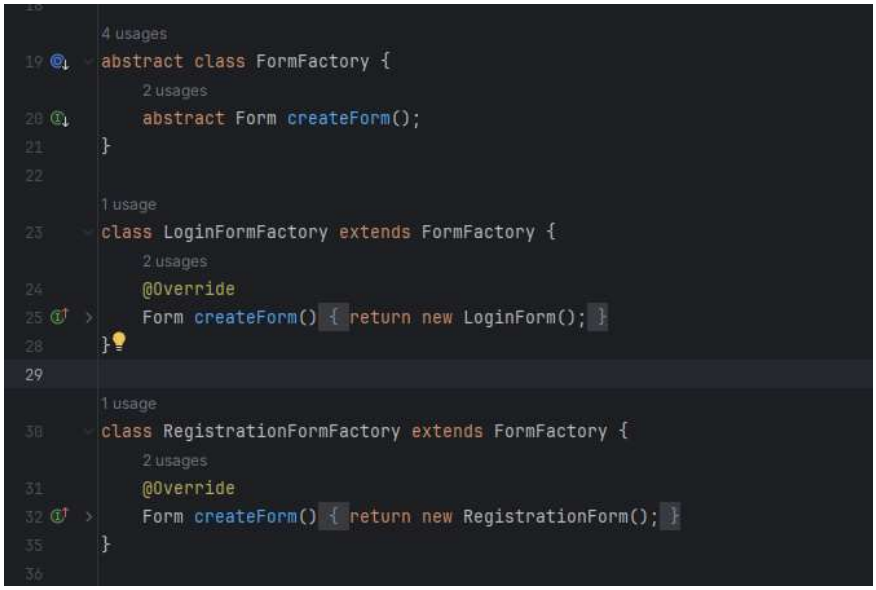

Рисунок 2 – Создание абстрактного класса и его использование

Пропишем код для использования шаблона проектирования Factory Method. Создаем основной класс. Этот класс содержит метод main, который является точкой входа программы. Внутри метода main создаются объекты фабрик и используются для создания конкретных объектов. Для создания частного объекта создадим объект фабрики для формы входа и регистрации. Затем вызывается фабричный метод createForm(), который возвращает объект формы. Этот объект затем используется для вызова метода display(), который выводит информацию о формах (рис. 3).

```
48
49  public class Main {
50      public static void main(String[] args) {
51
52          FormFactory loginFormFactory = new LoginFormFactory();
53          Form loginForm = loginFormFactory.createForm();
54          loginForm.display();
55
56
57          FormFactory registrationFormFactory = new RegistrationFormFactory();
58          Form registrationForm = registrationFormFactory.createForm();
59          registrationForm.display();
60      }
61  }
62
```

Рисунок 3 – Применение шаблона Factory Method

Тестируем код (рис. 4). В выводе можно увидеть строки "Форма входа в систему" и "Форма регистрации", которые являются результатами вызовов метода display() для объектов. Фабричный метод реализован.

Рисунок 4 – Успешное выполнение программы

Выводы

В данной статье был рассмотрен процесс использования шаблона проектирования Factory Method для реализации интерфейсов форм на языке программирования Java. Созданные фабрики позволяют динамически создавать объекты форм, а клиентский код может взаимодействовать с ними, не завися от конкретных классов форм. Использование Factory Method способствует созданию чистого, модульного кода. Применение данного шаблона обеспечивает гибкость, расширяемость и упрощенное добавление новых типов форм без необходимости изменения основного кода.

Библиографический список

1. Чёрный С.Г. Оптимизация процесса структуризации кода. // Восточно-Европейский журнал передовых технологий. 2011. Т. 2. № 2 (50). С. 31-34.
2. Волушкова В.Л. Архитектурные решения java для доступа к данным. //

Учебное пособие. Тверь, 2019.
3. Израилев В.Я. Реализация порождающих и поведенческих паттернов проектирования с использованием современных возможностей языка программирования C++. // Наука и высшая школа в инновационной деятельности. Сборник статей Международной научно-практической конференции. 2018. С. 10-12.
4. Крайнова Е.А. Теоретические аспекты паттерного программирования. // Вестник Волжского университета им. В.Н. Татищева. 2013. № 2 (21). С. 82-90.
5. Кравченко Д.А. Порождающие паттерны проектирования в java. // Неделя науки Санкт-Петербургского государственного морского технического университета. 2020. Т. 1. № 3-1. С. 27.