Можно сказать, что разработанная система пожарной охраны и дымоудаления является актуальной, как минимум, на данном объекте, а также эффективной в рамках возникновения потенциальны угроз, описанных в данной работе.

Литература

1. [Электронный ресурс] – Система пожарной сигнализации URL https://propb.ru/library/wiki/sistema-pozharnoy-signalizatsii/. – Дата доступа: 09.03.2024.
2. [Электронный ресурс] – Строительные нормы, пожарная автоматика зданий и сооружений URL https://geopartner.by/assets/docs/12-sn-2-02-03-2019-pozharnaya-avtomatika-zdanii%CC%86-i-sooruzhenii%CC%86.pdf. – Дата доступа: 08.03.2024.
3. [Электронный ресурс] – Система дымоудаления URL https://www.airfresh.ru/Dymoudalenie-pomescheniy.htm- Дата доступа: 10.03.2024.

А. В. Коваленко
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **О. М. Дерюжкова**, канд. физ.-мат. наук, доцент

ШАБЛОН ПРОЕКТИРОВАНИЯ PROTOTYPE В JAVA

Термин “designpatterns” в переводе с английского означает шаблоны проектирования. Их применяют при создании информационных систем, при периодичном возникновении типовых задач, а также шаблоны помогают в использовании полученных решений в некоторых ситуациях. Нужно знать, что каждый из шаблонов проектирования имеет своё название.

Когда разработчик корректно использует паттерны, то это становится очень весомым преимуществом. Общими словами модель, построенная на основе шаблона проектирования, имеет более простой вид даже в сравнении с простой моделью. Плюс ко всему, такая модель будет выделять особенно важные для решения элементы. Также она позволяет прорабатывать архитектуру системы.

Существует три типа паттернов проектирования: порождающие, структурные и поведенческие [1].

Рассмотрим порождающий паттерн ***Prototype***. Это шаблон, который дает возможность клонировать объекты, не разбираясь в том, как они были созданы.

Проблемы. Был создан объект, который необходимо клонировать. Как это реализовать? Суть в том, что необходимо создать еще один объект того же класса, но уже пустой, далее нужно скопировать данные всех переменных из первого объекта во второй. Тут возникает первая проблема. Данный способ не всегда сработает, так как некоторые поля могут быть приватными, а значит недоступными вне этого класса.

Это не единственная проблема. Код, который копируют, будет привязан к исходному классу. Как же решить эти две проблемы?

Решение. Шаблон Prototype делает ответственным за создание копий сам объект, который клонируют. Он создает интерфейс, который имплементирует все объекты, имеющие отношение к копированию. Это решает проблему привязки к классу. Как правило, такой интерфейс содержит один единственный метод clone.

Метод clone реализуется идентично в разных классах. Он берет на себя создание нового объекта класса и так же сам копирует данные переменных старого объекта. Таким образом, решается первая проблема, связанная с приватными полями, так как язык программирования java дает доступ к приватным переменным любого объекта исходного класса.

Название Prototype связано с тем, что объект, чью копию необходимо создать, называется прототипом [2].

Проект. Предположим, что есть некий удалённый репозиторий, на котором находится разрабатываемый проект. Необходимо получить копию данного проекта так, чтобы исходный файл оставался не изменённым.

Это возможно, если создать интерфейс Copeable,в котором будет один единственный метод (рисунок 1).

```
public interface Copyable {
    Object copy();
}
```

Рисунок 1 – Интерфейс Copyable

Далее создается класс Projectc полями idProjectName и sourceCode.

Обязательным моментом является то, что этот класс должен имплементировать интерфейс Copyable, созданный на предыдущем шаге (рисунок 2).

```
public class Project implements Copyable{
    private int id;
    private String projectName;
    private String sourceCode;
```

Рисунок 2 – Имплементация интерфейса

Следующий этап – это создание основного класса, в котором будет показана работа приложения (рисунок 3).

```
public class VersionControlRunner {
    public static void main(String[] args) {
        Project master = new Project( id: 1,  projectName: "SuperProject",  sourceCode: "SourceCode sourceCode = new SourceCode();");

        System.out.println(master);
    }
}
```

Рисунок 3 – Начало основного класса

Последнее дополнение (рисунок 4).

```
public class VersionControlRunner {
    public static void main(String[] args) {
        Project master = new Project( id: 1,  projectName: "SuperProject",  sourceCode: "SourceCode sourceCode = new SourceCode();");

        System.out.println(master);

        ProjectFactory factory = new ProjectFactory(master);
        Project masterClone = factory.cloneProject();
        System.out.println("\n============================\n");
        System.out.println(masterClone);
```

Рисунок 4 – Окончательный вид программы

И наконец, получаем результат (рисунок 5).

```
Project{id=1, projectName='SuperProject', sourceCode='SourceCode sourceCode = new SourceCode();'}

==============================

Project{id=1, projectName='SuperProject', sourceCode='SourceCode sourceCode = new SourceCode();'}

Process finished with exit code 0
```

Рисунок 5 – Результат проектирования

Таким образом, при выполнении данного проекта был использован шаблон проектирования Prototype и его реализация в языке Java. В работе рассмотрены проблемы, которые появляются при использовании данного паттерна и пути их решения. Шаблон проектирования Prototype оказывает существенную помощь разработчикам при работе с любыми языками программирования.

Литература

1. Паттерны проектирования: какие бывают и как выбрать нужный [Электронный ресурс]. – 2023. – URL:https://gb.ru/blog/patterny-proektirovaniya/. – Дата доступа: 10.02.2024.

2. Прототип [Электронный ресурс]. – 2023. – URL: https://refactoring.guru/ru/design-patterns/prototype. – Дата доступа: 10.02.2024.

В. В. Козликовская
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **Е. М. Березовская**, канд. физ.-мат. наук, доцент

РЕАЛИЗАЦИЯ ФУНКЦИИ АВТОМАТИЧЕСКОЙ СМЕНЫ ЯЗЫКА В ПРИЛОЖЕНИИ INSEARCHOFFOOD ПРИ ИЗМЕНЕНИИ ЯЗЫКА СИСТЕМЫ

Растущая глобализация и разнообразие языков создают потребность в разработке многоязычных приложений. Данная работа посвящена разработке приложения InSearchOfFood для поиска рецептов с использованием Android Studio, Kotlin и XML. На текущем этапе написания приложения основной акцент был сделан на реализации функции смены языка приложения при изменении языка системы устройства.

Был проведен анализ существующих методов и принято решение о выборе наиболее эффективного подхода к реализации функции автоматической смены языка, учитывая как технические, так и пользовательские аспекты. Для осуществления вышеописанного функционала приложения были использованы стандартные инструменты Android, ресурсы приложения и системные настройки.

Разработанная функция была успешно интегрирована в приложение и протестирована на различных устройствах с разными языковыми настройками (рисунок 1).

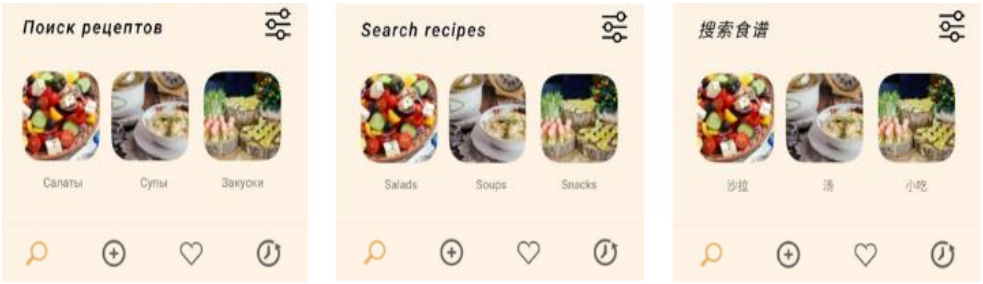

Рисунок 1 – Фрагмент перевода интерфейса приложения на разные языки