**Федеральное государственное автономное образовательное учреждение**

**высшего профессионального образования**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ «МИСИС»**

**Лабораторная работа №3**

**по курсу:**

**«Технологии программирования»**

**по теме:**

**«Порождающие паттерны проектирования»**

**Студент:** Князев Иван Викторович

**Группа:** БИВТ-23-8

**Преподаватель:** Гласов Александр Владимирович

**Цель работы**

Изучить порождающие паттерны проектирования и реализовать их на каком-либо ЯП.

**Порядок выполнения роботы:**

1) Продумать реализацию примеров использования паттернов согласно техническому заданию;

2) Выбрать ЯП (язык программирования) для разработки;

3) Реализовать заданный функционал;

4) Написать тесты для проверки корректности работы системы;

5) Запушить проект в удалённый репозиторий.

**Ход работы:**

Необходимо было реализовать следующие порождающие паттерны проектирования:

* Singleton (одиночка);
* Factory method (Фабричный метод);
* Abstract factory (Абстрактная фабрика);
* Builder (Строитель).

Для реализации был выбран ЯП **Golang** и следующая структура проекта:

.

├── creational

│ ├── cmd

│ │ ├── absfactory

│ │ │ ├── main.go

│ │ │ └── output.txt

│ │ ├── builder

│ │ │ ├── main.go

│ │ │ └── output.txt

│ │ ├── factory

│ │ │ ├── main.go

│ │ │ └── output.txt

│ │ └── singleton

│ │ ├── main.go

│ │ └── output.txt

│ └── internal

│ ├── absfactory

│ │ └── absfactory.go

│ ├── builder

│ │ └── builder.go

│ ├── factory

│ │ └── factory.go

│ └── singleton

│ └── singleton.go

├── go.mod

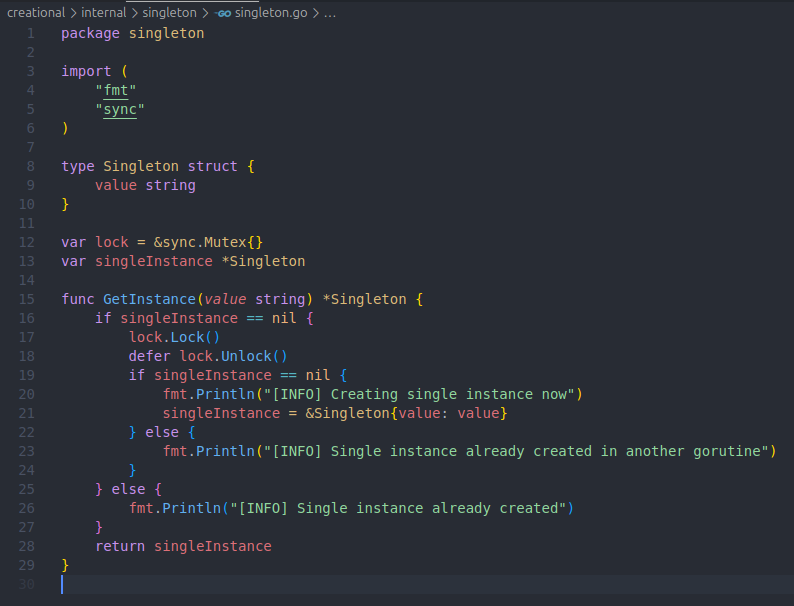
├── README.md

└── Лаба\_3\_Князев\_Иван.docx

Реализация всех паттернов находится в папке internal. В директории cmd располагается клиентский код, тестирующий написанные паттерны.

**1. Singleton** гарантирует существование только одного объекта определённого класса, а также предоставляет единый интерфейс для доступа к этому объект из любого места программы.

Он был написан на Go потокобезопасно с использованием примитивов синхронизации Mutex в его реализации и WaitGroup в клиентском коде.



**2. Factory method** определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

Реализация приведена в репозитории.

**3. Abstract factory** позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.

Реализация также приведена в репозитории.

**4. Builder** позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.

Реализация также приведена в репозитории.

Также в корневой папке был создан файл **README.md**, в который была добавлена инструкция для запуска каждого клиентского приложения и информация о расположении исходных файлов и выходных данных работы программ.

После в GitHub был создан новый репозиторий, в который в последующем запушен весь код.

Репозиторий располагается по ссылке:

https://github.com/Ivan-Knyazev/Labs\_3-5\_programming\_technologies