**Федеральное государственное автономное образовательное учреждение**

**высшего профессионального образования**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ «МИСИС»**

**Лабораторная работа №4**

**по курсу:**

**«Технологии программирования»**

**по теме:**

**«Поведенческие паттерны проектирования»**

**Студент:** Князев Иван Викторович

**Группа:** БИВТ-23-8

**Преподаватель:** Гласов Александр Владимирович

**Цель работы**

Изучить поведенческие 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:

* Strategy (Стратегия);
* Responsibility chain (Цепочка обязанностей);
* Iterator (Итератор).

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

.

├── behavioral

│ ├── cmd

│ │ ├── iterator

│ │ │ ├── main.go

│ │ │ └── output.txt

│ │ ├── responsibilityChain

│ │ │ ├── main.go

│ │ │ └── output.txt

│ │ └── strategy

│ │ ├── main.go

│ │ └── output.txt

│ └── internal

│ ├── iterator

│ │ └── iterator.go

│ ├── responsibilityChain

│ │ └── responsibilityChain.go

│ └── strategy

│ └── strategy.go

├── go.mod

├── README.md

└── Лаба\_4\_Князев\_Иван.docx

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

**1. Strategy** определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.

Реализация приведена в репозитории.

**2. Responsibility chain** позволяет передавать запросы последовательно по цепочке обработчиков. Каждый последующий обработчик решает, может ли он обработать запрос сам и стоит ли передавать запрос дальше по цепи.

Реализация также приведена в репозитории.

**3. Iterator** даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего представления.

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