

**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 работы:

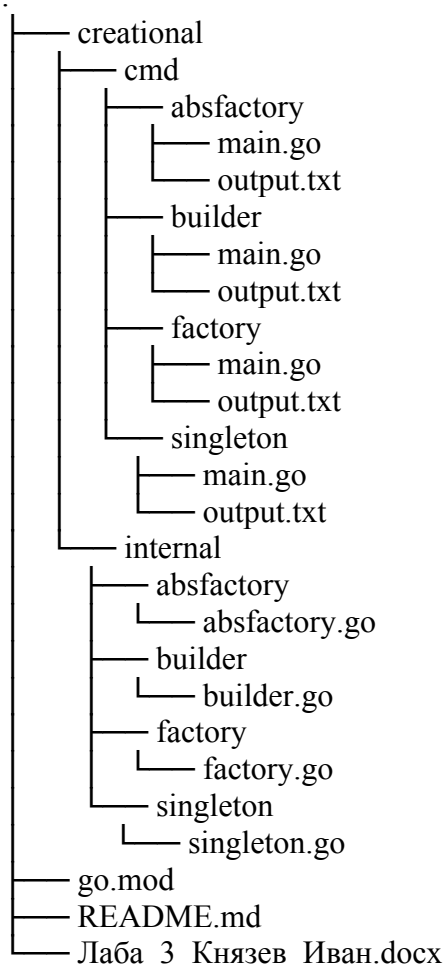
- 1) Продумать реализацию примеров использования паттернов согласно техническому заданию;
- 2) Выбрать ЯП (язык программирования) для разработки;
- 3) Реализовать заданный функционал;
- 4) Написать тесты для проверки корректности работы системы;
- 5) Запустить проект в удалённый репозиторий.

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:



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.

```
creational > internal > singleton > -o singleton.go > ...
1  package singleton
2
3  import (
4      "fmt"
5      "sync"
6  )
7
8  type Singleton struct {
9      value string
10 }
11
12 var lock = &sync.Mutex{}
13 var singleInstance *Singleton
14
15 func GetInstance(value string) *Singleton {
16     if singleInstance == nil {
17         lock.Lock()
18         defer lock.Unlock()
19         if singleInstance == nil {
20             fmt.Println("[INFO] Creating single instance now")
21             singleInstance = &Singleton{value: value}
22         } else {
23             fmt.Println("[INFO] Single instance already created in another goroutine")
24         }
25     } else {
26         fmt.Println("[INFO] Single instance already created")
27     }
28     return singleInstance
29 }
30
```

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 запущен весь код.

Репозиторий располагается по ссылке:

https://github.com/Ivan-Knyazev/Labs_3-5_programming_technologies