

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчёт
по лабораторной работе №6

Выполнила:
студентка группы ПО-9
Матюшик Е.П.

Проверил:
Крощенко А. А.

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java

- Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.

- Реализовать фрагмент программной системы, используя выбранный паттерн.

Реализовать все необходимые дополнительные классы.

Номер зачётной книжки: 210662

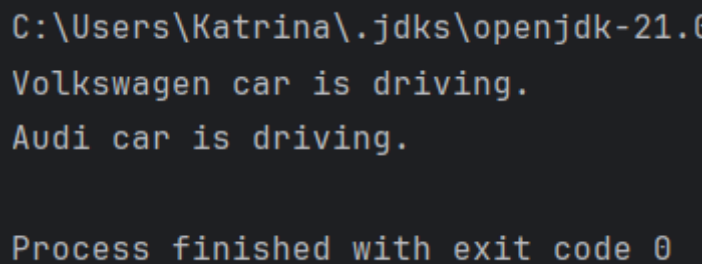
Вариант 2

Задание 1

Заводы по производству автомобилей. Реализовать возможность создавать автомобили различных типов на различных заводах.

Был выбран паттерн Абстрактная фабрика, т.к. он позволяет создавать семейства связанных объектов. В нашем случае могут создаваться различные типы автомобилей на различных заводах. А также, паттерн Абстрактная фабрика обеспечивает гибкую архитектуру, которая легко расширяется для добавления новых типов автомобилей и заводов.

Результат программы:



```
C:\Users\Katrina\.jdk\openjdk-21.0
Volkswagen car is driving.
Audi car is driving.

Process finished with exit code 0
```

Код программы:

```
// Абстрактный класс для завода автомобилей
abstract class CarFactory {
    public abstract Car createCar();
}

// Конкретные заводы и их реализации
class VolkswagenFactory extends CarFactory {
    @Override
    public Car createCar() {
        return new VolkswagenCar();
    }
}

class AudiFactory extends CarFactory {
    @Override
    public Car createCar() {
        return new AudiCar();
    }
}

// Абстрактный класс для автомобиля
interface Car {
```

```

        void drive();
    }

    // Конкретные реализации автомобилей
    class VolkswagenCar implements Car {
        @Override
        public void drive() {
            System.out.println("Volkswagen car is driving.");
        }
    }

    class AudiCar implements Car {
        @Override
        public void drive() {
            System.out.println("Audi car is driving.");
        }
    }

    public class Main1 {
        public static void main(String[] args) {
            CarFactory volkswagenFactory = new VolkswagenFactory();
            CarFactory audiFactory = new AudiFactory();

            Car volkswagenCar = volkswagenFactory.createCar();
            Car audiCar = audiFactory.createCar();

            volkswagenCar.drive();
            audiCar.drive();
        }
    }

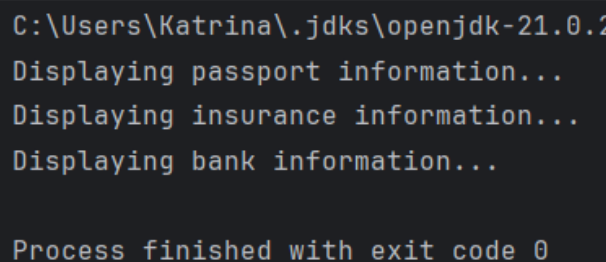
```

Задание 2

Проект «Универсальная электронная карта». В проекте должна быть реализована универсальная электронная карта, в которой есть функции паспорта, страхового полиса, банковской карты и т. д.

Для выполнения задания был выбран паттерн Фасад для создания унифицированного интерфейса для более крупного и сложного набора интерфейсов, который скрывает детали взаимодействия с внутренними компонентами (паспорт, страховка, банк). Это обеспечивает упрощенный и единообразный интерфейс для работы с универсальной картой.

Результат программы:



```

C:\Users\Katrina\.jdk\openjdk-21.0.2
Displaying passport information...
Displaying insurance information...
Displaying bank information...

Process finished with exit code 0

```

Код программы:

```
// Фасад для универсальной электронной карты
class ElectronicCardFacade {
    private Passport passport;
    private Insurance insurance;
    private Bank bank;

    public ElectronicCardFacade() {
        this.passport = new Passport();
        this.insurance = new Insurance();
        this.bank = new Bank();
    }

    // Методы для отображения информации о различных функциях карты
    public void displayPassportInfo() {
        passport.displayInfo();
    }

    public void displayInsuranceInfo() {
        insurance.displayInfo();
    }

    public void displayBankInfo() {
        bank.displayInfo();
    }
}

// Классы для различных функций на карте
class Passport {
    public void displayInfo() {
        System.out.println("Displaying passport information...");
    }
}

class Insurance {
    public void displayInfo() {
        System.out.println("Displaying insurance information...");
    }
}

class Bank {
    public void displayInfo() {
        System.out.println("Displaying bank information...");
    }
}

public class Client {
    public static void main(String[] args) {
        ElectronicCardFacade facade = new ElectronicCardFacade();
    }
}
```

```

        // Отображение информации о паспорте, страховке и банке
        facade.displayPassportInfo();
        facade.displayInsuranceInfo();
        facade.displayBankInfo();
    }
}

```

Задание 3

Вариант 3

Проект «Принтеры». В проекте должны быть реализованы разные модели принтеров, которые выполняют разные виды печати.

Был выбран паттерн Фабричный метод. В данном случае, каждая модель принтера может быть рассмотрена как отдельный тип объекта, создание которого может зависеть от определенных условий.

Результат программы:

```

C:\Users\Katrina\.jdk\openjdk-21.0.2\bin\java.exe "-javaage
Printing document on a regular printer: Document 1
Printing document in color on a color printer: Document 2

Process finished with exit code 0

```

Код программы:

```

// Интерфейс принтера
interface Printer {
    void print(String document);
}

// Конкретная реализация принтера для обычной печати
class RegularPrinter implements Printer {
    @Override
    public void print(String document) {
        System.out.println("Printing document on a regular printer: "
+ document);
    }
}

// Конкретная реализация принтера для цветной печати
class ColorPrinter implements Printer {
    @Override
    public void print(String document) {
        System.out.println("Printing document in color on a color
printer: " + document);
    }
}

// Фабрика для создания принтеров
interface PrinterFactory {
    Printer createPrinter();
}

```

```

}

// Фабрика для создания обычных принтеров
class RegularPrinterFactory implements PrinterFactory {
    @Override
    public Printer createPrinter() {
        return new RegularPrinter();
    }
}

// Фабрика для создания цветных принтеров
class ColorPrinterFactory implements PrinterFactory {
    @Override
    public Printer createPrinter() {
        return new ColorPrinter();
    }
}

// Пример использования
public class Main3 {
    public static void main(String[] args) {
        // Создание фабрик для разных типов принтеров
        PrinterFactory regularPrinterFactory = new
RegularPrinterFactory();
        PrinterFactory colorPrinterFactory = new
ColorPrinterFactory();

        // Создание принтеров с использованием соответствующих фабрик
        Printer regularPrinter =
regularPrinterFactory.createPrinter();
        Printer colorPrinter = colorPrinterFactory.createPrinter();

        // Печать документов
        regularPrinter.print("Document 1");
        colorPrinter.print("Document 2");
    }
}

```

Вывод: в ходе выполнения лабораторной были приобретены навыки применения паттернов проектирования при решении практических задач с использованием языка Java.