

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

**ОТЧЁТ**  
по лабораторной работе №6

Выполнила  
студентка группы ПО-9  
Тупик Ю. Л.

Проверил:  
Крощенко А. А.

Брест 2024

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

### **Задание 1**

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

### **Выполнение задания**

Для реализации возможности создания автомобилей различных типов на различных заводах можно использовать паттерн проектирования "Абстрактная фабрика".

Паттерн "Абстрактная фабрика" позволяет создавать семейства связанных объектов без указания их конкретных классов. В данном случае, автомобили различных типов (например, легковые, внедорожники) будут представлять семейства связанных объектов. Каждый завод будет реализовывать свою конкретную фабрику, специализированную на производстве определенных типов автомобилей.

### **Код программы**

```
// Абстрактный класс автомобиля
abstract class Car {
    abstract String getType();
}

// Конкретные классы автомобилей
class SedanCar extends Car {
    @Override
    String getType() {
        return "Седан";
    }
}

class SUVCar extends Car {
    @Override
    String getType() {
        return "Внедорожник";
    }
}

// Абстрактная фабрика для производства автомобилей
interface CarFactory {
    Car createCar();
}

// Конкретные фабрики для производства различных типов автомобилей
class SedanCarFactory implements CarFactory {
    @Override
    public Car createCar() {
        return new SedanCar();
    }
}

class SUVCarFactory implements CarFactory {
    @Override
```

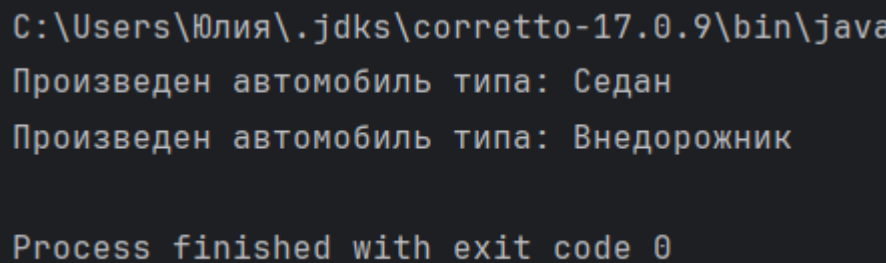
```
        public Car createCar() {
            return new SUVCar();
        }
    }

    public class Task1 {
        public static void main(String[] args) {
            CarFactory sedanFactory = new SedanCarFactory();
            CarFactory suvFactory = new SUVCarFactory();

            Car sedan = sedanFactory.createCar();
            System.out.println("Произведен автомобиль типа: " +
            sedan.getType());

            Car suv = suvFactory.createCar();
            System.out.println("Произведен автомобиль типа: " + suv.getType());
        }
    }
}
```

### Результат



```
C:\Users\Юлия\.jdk\corretto-17.0.9\bin\java
Произведен автомобиль типа: Седан
Произведен автомобиль типа: Внедорожник

Process finished with exit code 0
```

## Задание 2

10) Учетная запись покупателя книжного интернет-магазина. Предусмотреть различные уровни учетки в зависимости от активности покупателя. Дополнительные уровни добавляют функциональные возможности и открывают доступ к уникальным предложениям.

### Выполнение задания

Для данного задания можно использовать паттерн «Стратегия». Этот паттерн позволяет определить семейство алгоритмов, инкапсулировать каждый из них и делать их взаимозаменяемыми. Таким образом, можно легко добавлять новые уровни активности и изменять их поведение без изменения кода самой учетной системы.

### Код программы

```
// Интерфейс Уровня Активности
interface ActivityLevel {
    void accessFunctionality();
    void accessUniqueOffers();
}

// Конкретный класс Уровня Активности: Базовый уровень
class BasicLevel implements ActivityLevel {
    public void accessFunctionality() {
        System.out.println("Доступ к базовым функциям.");
    }

    public void accessUniqueOffers() {
        System.out.println("Доступ к базовым уникальным предложениям.");
    }
}

// Конкретный класс Уровня Активности: Премиум уровень
class PremiumLevel implements ActivityLevel {
    public void accessFunctionality() {
        System.out.println("Доступ к расширенным функциям.");
    }

    public void accessUniqueOffers() {
        System.out.println("Доступ к премиум уникальным предложениям.");
    }
}

// Класс Учетной Записи Покупателя
class CustomerAccount {
    private ActivityLevel activityLevel;

    public void setActivityLevel(ActivityLevel activityLevel) {
        this.activityLevel = activityLevel;
    }

    public void accessFunctionality() {
        activityLevel.accessFunctionality();
    }

    public void accessUniqueOffers() {
```

```

        activityLevel.accessUniqueOffers();
    }
}

public class Task2 {
    public static void main(String[] args) {
        // Создаем учетную запись покупателя
        CustomerAccount customerAccount = new CustomerAccount();

        // Устанавливаем уровень активности на базовый
        ActivityLevel basicLevel = new BasicLevel();
        customerAccount.setActivityLevel(basicLevel);

        // Покупатель получает доступ к функциональности и уникальным
        // предложениям базового уровня
        System.out.println("Базовый уровень:");
        customerAccount.accessFunctionality();
        customerAccount.accessUniqueOffers();
        System.out.println();

        // Переключаем уровень активности на премиум
        ActivityLevel premiumLevel = new PremiumLevel();
        customerAccount.setActivityLevel(premiumLevel);

        // Покупатель получает доступ к расширенной функциональности и
        // уникальным предложениям премиум уровня
        System.out.println("Премиум уровень:");
        customerAccount.accessFunctionality();
        customerAccount.accessUniqueOffers();
    }
}

```

## Результат

```

C:\Users\Юлия\.jdk\corretto-17.0.9\bin\java.
Базовый уровень:
Доступ к базовым функциям.
Доступ к базовым уникальным предложениям.

Премиум уровень:
Доступ к расширенным функциям.
Доступ к премиум уникальным предложениям.

Process finished with exit code 0

```

### Задание 3

10) Проект «Банкомат». Предусмотреть выполнение основных операций (ввод пин-кода, снятие суммы, завершение работы) и наличие различных режимов работы (ожидание, аутентификация, выполнение операции, блокировка – если нет денег). Атрибуты: общая сумма денег в банкомате, ID.

#### Выполнение задания

Для данного задания можно использовать паттерн проектирования "Состояние". Этот паттерн позволяет объекту изменять свое поведение в зависимости от его внутреннего состояния. В данном случае, банкомат может находиться в различных состояниях, таких как ожидание, аутентификация, выполнение операции и блокировка.

#### Код программы

```
// Интерфейс для всех состояний
interface ATMState {
    void insertPin(int pin);
    void withdrawCash(double amount);
    void ejectCard();
}

// Состояние ожидания
class IdleState implements ATMState {
    private final ATM atm;

    public IdleState(ATM atm) {
        this.atm = atm;
    }

    @Override
    public void insertPin(int pin) {
        if (pin == atm.correctPin) {
            System.out.println("PIN введен верно.");
            atm.setState(atm.getAuthenticatedState());
        } else {
            System.out.println("Неверный PIN.");
        }
    }

    @Override
    public void withdrawCash(double amount) {
        System.out.println("Пожалуйста, введите PIN.");
    }

    @Override
    public void ejectCard() {
        System.out.println("Карта извлечена.");
    }
}

// Состояние аутентификации
class AuthenticatedState implements ATMState {
    private final ATM atm;

    public AuthenticatedState(ATM atm) {
        this.atm = atm;
    }

    @Override
    public void insertPin(int pin) {
        System.out.println("Вы уже ввели PIN.");
    }
}
```

```

    }

    @Override
    public void withdrawCash(double amount) {
        if (amount <= atm.availableFunds) {
            System.out.println("Выдача наличных: " + amount);
            atm.availableFunds -= amount;
        } else {
            System.out.println("Недостаточно средств.");
            atm.setState(atm.getOutOfCashState());
        }
    }

    @Override
    public void ejectCard() {
        System.out.println("Карта извлечена.");
        atm.setState(atm.getIdleState());
    }
}

// Состояние блокировки
class OutOfCashState implements ATMState {
    private final ATM atm;

    public OutOfCashState(ATM atm) {
        this.atm = atm;
    }

    @Override
    public void insertPin(int pin) {
        System.out.println("Банкомат заблокирован.");
    }

    @Override
    public void withdrawCash(double amount) {
        System.out.println("Банкомат заблокирован.");
    }

    @Override
    public void ejectCard() {
        System.out.println("Банкомат заблокирован.");
    }
}

// Класс ATM
class ATM {
    private ATMState idleState;
    private ATMState authenticatedState;
    private ATMState outOfCashState;
    private ATMState currentState;

    int correctPin;
    double availableFunds;

    public ATM(int correctPin, double availableFunds) {
        this.correctPin = correctPin;
        this.availableFunds = availableFunds;

        idleState = new IdleState(this);
        authenticatedState = new AuthenticatedState(this);
        outOfCashState = new OutOfCashState(this);
    }
}

```

```

        currentState = idleState;
    }

    public void setState(ATMState state) {
        currentState = state;
    }

    public ATMState getIdleState() {
        return idleState;
    }

    public ATMState getAuthenticatedState() {
        return authenticatedState;
    }

    public ATMState getOutOfCashState() {
        return outOfCashState;
    }

    public void insertCard() {
        System.out.println("Карта вставлена.");
    }

    public void insertPin(int pin) {
        currentState.insertPin(pin);
    }

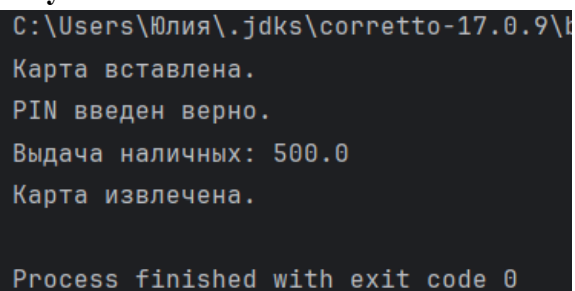
    public void withdrawCash(double amount) {
        currentState.withdrawCash(amount);
    }

    public void ejectCard() {
        currentState.ejectCard();
    }
}

public class Task3 {
    public static void main(String[] args) {
        ATM atm = new ATM(1234, 1000);
        atm.insertCard();
        atm.insertPin(1234);
        atm.withdrawCash(500);
        atm.ejectCard();
    }
}

```

### Результат



```

C:\Users\Юлия\.jdk\corretto-17.0.9\bin>java Task3
Карта вставлена.
PIN введен верно.
Выдача наличных: 500.0
Карта извлечена.

Process finished with exit code 0

```

**Вывод:** приобрела навыки применения паттернов проектирования при решении практических задач с использованием языка Java.



