**ЛАБОРАТОРНАЯ РАБОТА №7**

**ШАБЛОНЫ ПРОЕКТИРОВАНИЯ**

**Цель работы**: ознакомиться с основными шаблонами проектирования, научиться применять их при проектировании и разработке ПО.

**Программное обеспечение**: JDK 1.8+; IntelliJ IDEA; плагин PlantUML integration.

**Задания**: В каждом из вариантов указан шаблон для реализации и проект, использующий этот шаблон. Необходимо сделать следующее:

1. Нарисовать в PlantUML диаграмму классов реализуемой программы. (проектирование)

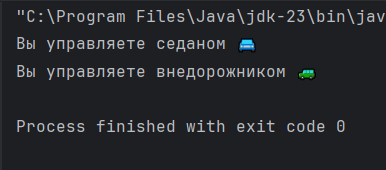
2. Реализовать программу на Java. (реализация)

Вариант №5, 13, 21, 29

Шаблон “Абстрактная фабрика”. Проект “Заводы по производству автомобилей”. В проекте должно быть реализована возможность создавать автомобили различных типов на разных заводах.

**Листинг**

public class Main {  
  
 // Интерфейс автомобиля  
 interface Car {  
 void drive();  
 }  
  
 // Класс седана  
 static class Sedan implements Car {  
 @Override  
 public void drive() {  
 System.*out*.println("Вы управляете седаном 🚘");  
 }  
 }  
  
 // Класс внедорожника  
 static class SUV implements Car {  
 @Override  
 public void drive() {  
 System.*out*.println("Вы управляете внедорожником 🚙");  
 }  
 }  
  
 // Фабричный интерфейс  
 interface CarFactory {  
 Car createCar();  
 }  
  
 // Завод по производству седанов  
 static class SedanFactory implements CarFactory {  
 @Override  
 public Car createCar() {  
 return new Sedan();  
 }  
 }  
  
 // Завод по производству внедорожников  
 static class SUVFactory implements CarFactory {  
 @Override  
 public Car createCar() {  
 return new SUV();  
 }  
 }  
  
 // Клиент, который заказывает автомобили  
 static class Client {  
 private final CarFactory factory;  
  
 public Client(CarFactory factory) {  
 this.factory = factory;  
 }  
  
 public void orderCar() {  
 Car car = factory.createCar();  
 car.drive();  
 }  
 }  
  
 // Главный метод (точка входа)  
 public static void main(String[] args) {  
 // Клиент заказывает седан  
 Client client1 = new Client(new SedanFactory());  
 client1.orderCar(); // Вывод: "Вы управляете седаном 🚘"  
  
 // Клиент заказывает внедорожник  
 Client client2 = new Client(new SUVFactory());  
 client2.orderCar(); // Вывод: "Вы управляете внедорожником 🚙"  
 }  
}



@startuml  
  
interface Car {  
 + drive(): void  
}  
note right of Car  
 Интерфейс автомобиля  
 с методом drive()  
end note  
  
class Sedan implements Car {  
 + drive(): void  
}  
note right of Sedan  
 Класс седана  
end note  
  
class SUV implements Car {  
 + drive(): void  
}  
note right of SUV  
 Класс внедорожника  
end note  
  
interface CarFactory {  
 + createCar(): Car  
}  
note right of CarFactory  
 Фабричный интерфейс  
 для создания автомобилей  
end note  
  
class SedanFactory implements CarFactory {  
 + createCar(): Car  
}  
note right of SedanFactory  
 Завод по производству седанов  
end note  
  
class SUVFactory implements CarFactory {  
 + createCar(): Car  
}  
note right of SUVFactory  
 Завод по производству внедорожников  
end note  
  
class Client {  
 - factory: CarFactory  
 + Client(factory: CarFactory)  
 + orderCar(): void  
}  
note left of Client  
 Клиент, заказывающий автомобиль  
 через фабрику  
end note  
  
Client --> CarFactory  
CarFactory <|.. SedanFactory  
CarFactory <|.. SUVFactory  
Car <|.. Sedan  
Car <|.. SUV  
SedanFactory --> Sedan  
SUVFactory --> SUV  
  
@enduml

