**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине: «Технологии разработки программного обеспечения»

на тему: «Развитие представления о разработке программ. Объектно-ориентированное программирование в *Java*»

Выполнил: студент гр. ИТП-21

Царьков В.П.

Принял: преподаватель-стажер

Малиновский И.Л.

Гомель 2023

**Цель работы:**

1. Разработать *UML*-диаграмму иерархии классов, согласно варианта (таблица 1).

2. При наименовании компонентов руководствоваться соглашением о наименовании ([*https://www.oracle.com/technetwork/java/codeconventions-150003.pdf*](https://www.oracle.com/technetwork/java/codeconventions-150003.pdf)).

3. При описании иерархии использовать наследование и композицию.

4. На основе *UML*-диаграммы разработать иерархию классов на языке *Java*.

5. Весь код должен быть снабжен элементами документирования ([*https://www.jetbrains.com/help/idea/working-with-code-documentation.html*](https://www.jetbrains.com/help/idea/working-with-code-documentation.html)).

6. Разработанную иерархию поместить в .*jar* файл для дальнейшего использования в качестве библиотечных классов.

7. Создать консольное приложение для демонстрации работы созданных классов.

8. Составить отчет о проделанной работе

**Вариант 5**

**Условие:**

1.1 Создать класс Car, Engine и Driver.

1.2 Класс Driver содержит поля - ФИО, стаж вождения.

1.3 Класс Engine содержит поля - мощность, производитель.

1.4 Класс Car содержит поля - марка автомобиля, класс автомобиля, вес, водитель типа Driver, мотор типа Engine.

1.5 Вывести водителей со стажем более 5 лет.

1.6 Вывести автомобили советского производства.

**Решение:**

В ходе выполнения лабораторной работы были разработаны классы *Car*, *Engine* и *Driver*. В котором были реализованы поля исходя из условия. Был реализован метод *getStage* для вывода водителей со стажем больше 5 лет и метод *getManufacture* для вывода автомобилей советского производства.

На рисунке 1 изображена *UML* схема классов

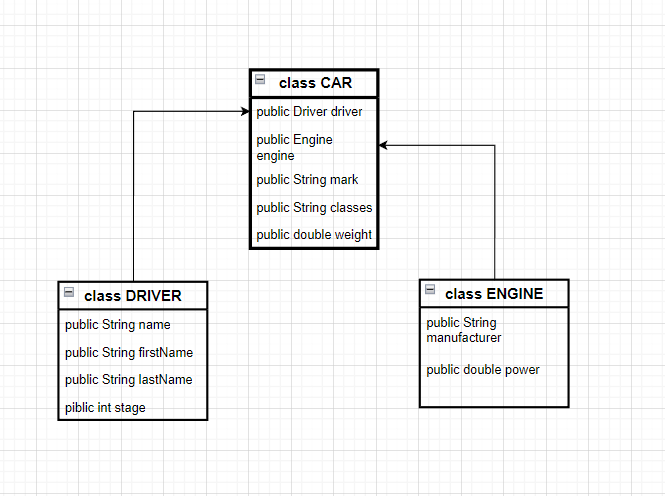


Рисунок 1 – *UML* схема классов

После создании архитектуры классов в методе *main* был создан список автомобилей со всеми данными параметрами. Результат работы метода main указан на рисунке 2.

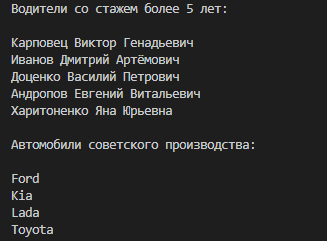


Рисунок 2 – Результат работы метода main

**Вывод**: в ходе выполнения лабораторной работы были изучены основные принципы построения приложений с помощью Объектно-Ориентированного подхода. Была разработана *UML* схема для построения иерархии классов. Был изучен способ сохранения иерархии классов в виде файла *jar*. Также были изучены основные синтаксические правила языка *Java*.

**ПРИЛОЖЕНИЕ A**

Листинг класса *Main*

import java.lang.reflect.Array;

import java.time.LocalTime;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

Car car1 = new Car("Audi", "A", 2700, new Driver("Царьков", "Даниил", "Константинович", 5), new Engine("Германия", 400));

Car car2 = new Car("BMW", "B", 2500, new Driver("Лесков", "Антон", "Андреевич", 3), new Engine("Польша", 500));

Car car3 = new Car("Ford", "C", 2450, new Driver("Карповец", "Виктор", "Генадьевич", 10), new Engine("Латвия", 550));

Car car4 = new Car("Kia", "E", 2300, new Driver("Иванов", "Дмитрий", "Артёмович", 6), new Engine("Россия", 300));

Car car5 = new Car("Honda", "F", 2800, new Driver("Доценко", "Василий", "Петрович", 8), new Engine("Австралия", 600));

Car car6 = new Car("Lada", "S", 2200, new Driver("Андропов", "Евгений", "Витальевич", 9), new Engine("Беларусь", 590));

Car car7 = new Car("Mazda", "A", 3000, new Driver("Измайлов", "Вадим", "Евгеньевич", 3), new Engine("Китай", 340));

Car car8 = new Car("Skoda", "B", 2480, new Driver("Амохин", "Никита", "Васильевич", 2), new Engine("Япония", 440));

Car car9 = new Car("Nissan", "C", 2100, new Driver("Карпович", "Пётр", "Сергеевич", 5), new Engine("Швейцария", 560));

Car car10 = new Car("Toyota", "A", 2950, new Driver("Харитоненко", "Яна", "Юрьевна", 7), new Engine("Казахстан", 410));

ArrayList<Car> cars = new ArrayList<>();

cars.add(car1);

cars.add(car2);

cars.add(car3);

cars.add(car4);

cars.add(car5);

cars.add(car6);

cars.add(car7);

cars.add(car8);

cars.add(car9);

cars.add(car10);

System.out.println("\nВодители со стажем более 5 лет:\n");

getStage(cars);

System.out.println("\nАвтомобили советского производства:\n");

getManufacture(cars);

}

public static void getStage(ArrayList<Car>cars){

for (int i = 0; i < cars.size(); i++) {

if (cars.get(i).driver.stage > 5) {

System.out.println(cars.get(i).driver.name + " " + cars.get(i).driver.firstName + " " + cars.get(i).driver.lastName);

}

}

}

public static void getManufacture(ArrayList<Car>cars){

for (int i = 0; i < cars.size(); i++) {

if (cars.get(i).engine.manufacturer == "Россия"

|| cars.get(i).engine.manufacturer == "Казахстан"

|| cars.get(i).engine.manufacturer == "Беларусь"

|| cars.get(i).engine.manufacturer == "Латвия")

{

System.out.println(cars.get(i).mark);

}

}

}

}

Листинг класса *Car*

public class Car {

public Driver driver;

public Engine engine;

public String mark;

public String classes;

public double weight;

public Car( String mark, String classes, double weight, Driver driver, Engine engine) {

this.driver = driver;

this.engine = engine;

this.mark = mark;

this.classes = classes;

this.weight = weight;

}

Листинг класса *Engine*

public class Engine {

public String manufacturer;

public double power;

public Engine(String manufacturer, double power) {

this.manufacturer = manufacturer;

this.power = power;

}

}

Листинг класса *Driver*

public class Driver {

public String name;

public String firstName;

public String lastName;

public int stage;

public Driver(String name, String firstName, String lastName, int stage){

this.name = name;

this.firstName = firstName;

this.lastName = lastName;

this.stage = stage;

}

}