**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 представлений о разработке программ. Объектно-ориентированное программирование в Java»

Выполнил: студент гр. ИТП-21

Гаврильченко А.Е.

Принял: преподаватель

Малиновский И.Л.

Гомель 2024

**Задание:**

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).

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).

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.

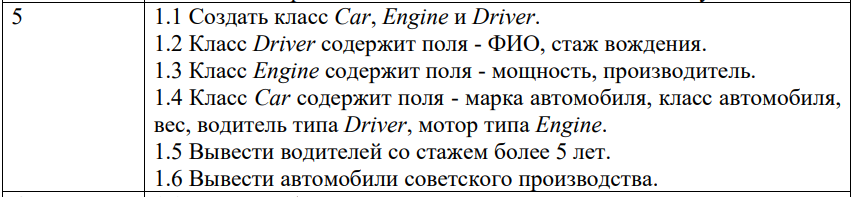


Рисунок 1. Задание для Варианта 5

Данная библиотека классов реализует список различных деталей, различающихся по виду и форме. Все детали наследуются от базового абстрактного класса. В работе представлены примеры реализации механизмов наследования и композиции. Для упрощения работы с библиотекой присутствует класс *CarProcessor.*

Работа программы представлена на рисунке 2



Рисунок 2. Работа программы

Диаграмма иерархии классов представлена на рисунке 3.

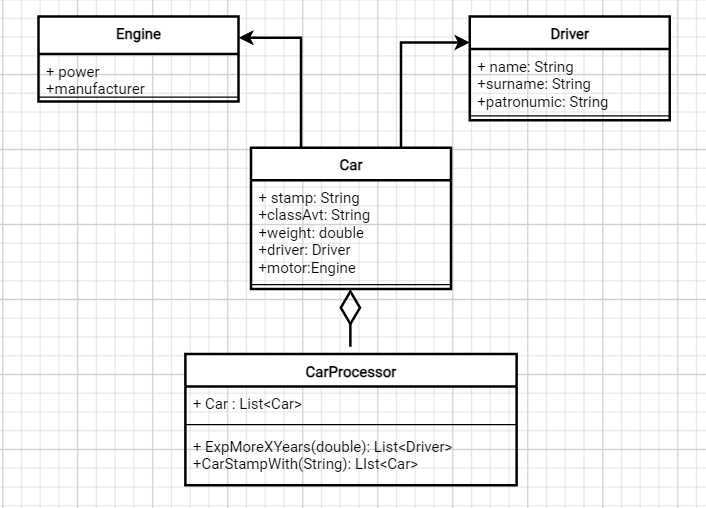


Рисунок 3. UML-диаграмма

Листинг программы представлен в приложениях А-Г.

**Вывод:** в ходе лабораторной работы изучили методы разработки библиотек классов на языке “*Java*”.

.

**ПРИЛОЖЕНИЕ А**

**Листинг класса «*CarProcessor»***

import java.util.ArrayList;  
import java.util.List;  
  
public class CarProcessor {  
 private final List<Car> cars;  
 public CarProcessor(List<Car> cars){  
 this.cars = cars;  
 }  
  
 public List<Driver> ExpMoreXyears(double Exyears){  
 List<Driver> drivers = new ArrayList<>();  
 for(Car car : cars){  
 if(car.getDriver().getDrivingExperience() > Exyears){  
 drivers.add(car.getDriver());  
 }  
 }  
 return drivers;  
 }  
  
 public List<Car> CarStampWith(String stamp){  
 List<Car> carsSt = new ArrayList<>();  
 for(Car car : cars){  
 if(stamp.equals(car.getStamp())){  
 carsSt.add(car);  
 }  
 }  
 return carsSt;  
 }  
}

**ПРИЛОЖЕНИЕ Б**

**Листинг класса Car**

public class Car{  
 private String stamp;  
 private String classAvt;  
 private double weight;  
 private Driver driver;  
 private Engine motor;  
 public Car(){}  
 public Car(String stamp, String classAvt, double weight, Driver driver, Engine motor){  
 this.stamp = stamp;  
 this.classAvt = classAvt;  
 this.weight = weight;  
 this.driver = driver;  
 this.motor = motor;  
 }  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public Driver getDriver() {  
 return driver;  
 }  
  
 public Engine getMotor() {  
 return motor;  
 }  
  
 public String getClassAvt() {  
 return classAvt;  
 }  
  
 public String getStamp() {  
 return stamp;  
 }  
}

**ПРИЛОЖЕНИЕ В**

**Листинг класса “Driver”**

public class Driver extends Car{  
 private final String name;  
 private final String surname;  
 private final String patronymic;  
 private double drivingExperience;  
  
 public Driver(String name, String surname, String patronymic, double drivingExperience) {  
 super();  
 this.name = name;  
 this.surname = surname;  
 this.patronymic = patronymic;  
 this.drivingExperience = drivingExperience;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public double getDrivingExperience() {  
 return drivingExperience;  
 }  
  
 public void setDrivingExperience(double drivingExperience) {  
 this.drivingExperience = drivingExperience;  
 }  
}

**ПРИЛОЖЕНИЕ Г**

**Листинг класса «** **Engine »**

public class Engine extends Car{  
 private final double power;  
 private final String manufacturer;  
 public Engine(double power, String manufacturer){  
 super();  
 this.power = power;  
 this.manufacturer = manufacturer;  
 }  
  
 public double getPower() {  
 return power;  
 }  
  
 public String getManufacturer() {  
 return manufacturer;  
 }  
}