**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Технологии разработки программного обеспечения»

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

Выполнил: студент гр. ИТИ-22

Ковшаров Г. Ю.

Принял: преподаватель

Карась О. В.

Гомель 2024

**Цель работы:** изучить основы объектно-ориентированного программирования на языке *Java*: синтаксис, создание библиотечных классов. Научиться разрабатывать *UML*-диаграммы.

**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.

|  |  |
| --- | --- |
| **Вариант** | **Условие задачи** |
| 5 | 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. |

**Ход работы:**

Для начала создадим *UML* схему классов, при помощи возможностей среды разработки *Intellij IDE*. Реализация *UML* представлена на рисунке 1.

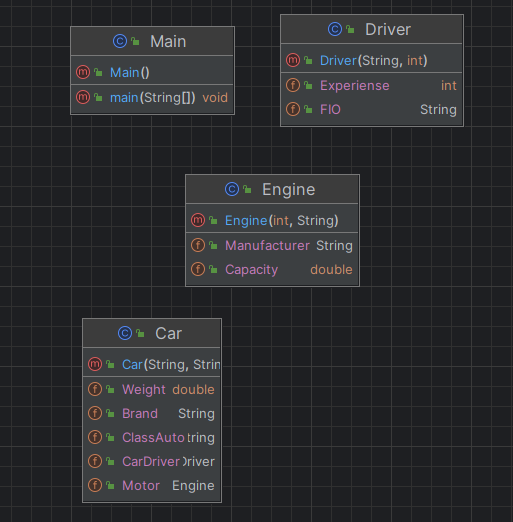


Рисунок 1 – реализация *UML* диаграммы

Код классов полученный из *UML* диаграммы прикреплён в Приложении А.

Теперь для создания *jar* файла выполним следующие шаги:

1. Перейдите в File -> Project Structure.
2. В открывшемся окне выберите Project Settings -> Artifacts.
3. Нажмите на знак + и выберите Jar -> From modules with dependencies.
4. Выберите основной класс (тот, в котором есть метод main), если вы хотите сделать jar-файл исполняемым.
5. Выберите Extract to the target Jar.
6. Нажмите OK, затем Apply / OK.

Теперь, чтобы фактически создать и сохранить jar-файл, выполните следующие действия:

1. Перейдите в Build -> Build Artifact.
2. Выберите ваш артефакт и нажмите Build.

Пример созданного *jar* файла приведён на рисунке 2.

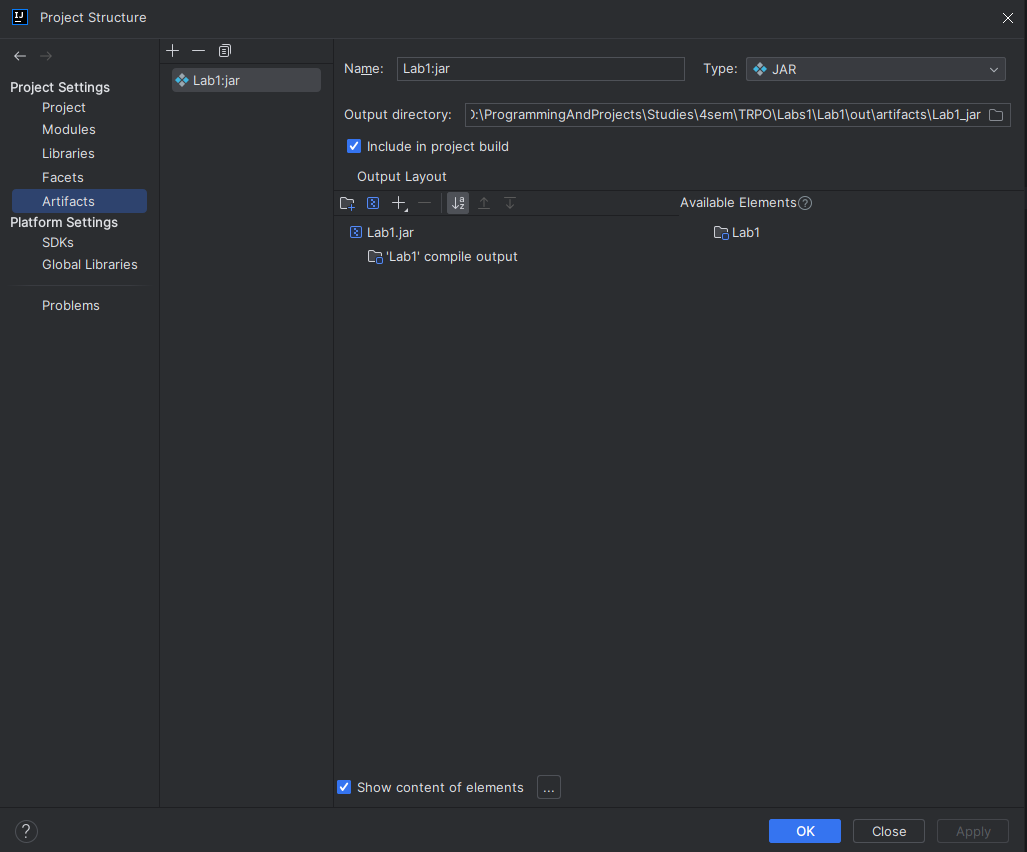


Рисунок 2 – созданный *jar* файл

Пример работы консольного приложения представлен на рисунке 3.

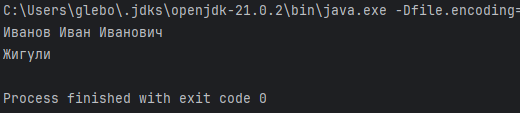


Рисунок 3 – пример работы консольного

**Выводы:** в ходе выполнения лабораторной работы мы познакомились с синтаксисом объектно-ориентированного языка *Java*. Научились разрабатывать классы, экземпляры, просты алгоритмы и иерархии классов. Также научились создавать диаграммы классов.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**Car**

package Classes;  
  
public class Car {  
 public String Brand;  
 public String ClassAuto;  
 public Driver CarDriver;  
 public double Weight;  
 public Engine Motor;  
  
 */\*\*  
 \* Конструктор  
 \* @param brand  
 \* @param classAuto  
 \* @param carDriver  
 \* @param weight  
 \* @param motor  
 \*/* public Car(String brand, String classAuto, Driver carDriver, double weight, Engine motor) {  
 Brand = brand;  
 ClassAuto = classAuto;  
 CarDriver = carDriver;  
 Weight = weight;  
 Motor = motor;  
 }  
}

**Driver**

package Classes;  
  
public class Driver {  
 public String FIO;  
 public int Experiense;  
  
 */\*\*  
 \* Конструктор  
 \* @param fio  
 \* @param experiense  
 \*/* public Driver(String fio, int experiense) {  
 FIO = fio;  
 Experiense = experiense;  
 }  
}

**Engine**

package Classes;  
  
public class Engine {  
 public double Capacity;  
 public String Manufacturer;  
  
 */\*\*  
 \* Конструктор  
 \* @param capacity  
 \* @param manufacturer  
 \*/* public Engine(int capacity, String manufacturer) {  
 Capacity = capacity;  
 Manufacturer = manufacturer;  
 }  
}

**Main**

package Classes;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
 */\*\*  
 \* Главный метод  
 \* @param args  
 \*/* public static void main(String[] args) {  
 List<Car> cars = new ArrayList<>();  
  
 // Создание водителей  
 Driver driver1 = new Driver("Иванов Иван Иванович", 6);  
 Driver driver2 = new Driver("Петров Петр Петрович", 4);  
  
 // Создание двигателей  
 Engine engine1 = new Engine(200, "СССР");  
 Engine engine2 = new Engine(150, "Toyota");  
  
 // Создание автомобилей и добавление их в список  
 cars.add(new Car("Жигули", "Седан", driver1, 1200, engine1));  
 cars.add(new Car("Toyota Corolla", "Седан", driver2, 1100, engine2));  
  
 // Вывести водителей со стажем более 5 лет  
 for (Car car : cars) {  
 if (car.CarDriver.Experiense > 5) {  
 System.*out*.println(car.CarDriver.FIO);  
 }  
 }  
  
 // Вывести автомобили советского производства  
 for (Car car : cars) {  
 if (car.Motor.Manufacturer.equals("СССР")) {  
 System.*out*.println(car.Brand);  
 }  
 }  
 }  
}