Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Отчёт
По лабораторной работе №3
По дисциплине СПП

Выполнил
Студент группы ПО-3
3-го курса
Кулинкович И. Т.

Проверил
Крощенко А. А.

Брест 2020

Лабораторная работа №3

ВАРИАНТ 13

Задание 1. Реализовать простой класс.
Требования к выполнению:
- Реализовать пользовательский класс по варианту.
- Создать другой класс с методом main, в котором будут находится примеры использования пользовательского класса.

Для каждого класса:
- Создать поля классов.
- Создать методы классов.
- Добавьте необходимые `get` и `set` методы (по необходимости).
- Укажите соответствующие модификаторы видимости.
- Добавьте конструкторы.
- Переопределите методы `toString()` и `equals()`.

Прямоугольный треугольник, заданный длинами сторон – Предусмотреть возможность определения площади и периметра, а так же логический метод, определяющий существует или такой треугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Задание 2.
Автоматизированная система в автобусном парке.
Составить программу, которая содержит информацию о наличие автобусов в автобусном парке:
- Сведения о каждом автобусе содержат (Bus) содержат:
- Фамилия и инициалы водителя;
- Номер автобуса;
- Номер маршрута;
- Марка;
- Год начала эксплуатации;
- Пробег;
- Местонахождение в настоящий момент времени (парк/маршрут).

Код программы

live.ilyusha.spp3.Triangle

```
package live.ilyusha.spp3;

class Triangle {

    private int a, b, c;

    Triangle(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    /* helper methods */

    public boolean exists() {
        return Math.pow(a, 2) + Math.pow(b, 2) == Math.pow(c, 2) && a > 0 &&
b > 0 && c > 0;
    }
```

```
    public double perimeter() {
        return a + b + c;
    }

    public double area() {
        double p = perimeter() / 2;
        return Math.sqrt(p * (p - a) * (p - b) * (p - c));
    }

    public void log() {
        System.out.printf(
            "triangle = %s\nexists = %s\nperimeter = %s\narea = %s\n\n",
            this.toString(), this.exists(), this.perimeter(), this.area()
        );
    }

    /* java.lang.Object */

    @Override
    public String toString() {
        return String.format("<Triangle a=%d b=%d c=%d>", a, b, c);
    }

    public boolean equals(Triangle t) {
        return t.a == a && t.b == b && t.c == c;
    }

    /* codegen */

    public int getA() {
        return a;
    }

    public void setA(int a) {
        this.a = a;
    }

    public int getB() {
        return b;
    }

    public void setB(int b) {
        this.b = b;
    }

    public int getC() {
        return c;
    }

    public void setC(int c) {
        this.c = c;
    }

}
```

live.ilyusha.spp3.Bus

```
package live.ilyusha.spp3;

class Bus {
```

```
    private String driver;
    private int plateNumber;
    private int routeNumber;
    private String brand;
    private int year;
    private double mileage;
    private boolean parked;

    public Bus() {}

    public Bus(String driver, int plateNumber, int routeNumber, String brand,
int year, double mileage, boolean parked) {
        this.setDriver(driver);
        this.setPlateNumber(plateNumber);
        this.setRouteNumber(routeNumber);
        this.setBrand(brand);
        this.setYear(year);
        this.setMileage(mileage);
        this.setParked(parked);
    }

    @Override
    public String toString() {
        return String.format(
            "<Bus driver=\"%s\" plateNumber=%d routeNumber=%d brand=\"%s\"
year=%d mileage=%f parked=%s>",
            getDriver(), getPlateNumber(), getRouteNumber(), getBrand(),
getYear(), getMileage(), isParked()
        );
    }

    /* codegen */

    public String getDriver() {
        return driver;
    }

    public void setDriver(String driver) {
        this.driver = driver;
    }

    public int getPlateNumber() {
        return plateNumber;
    }

    public void setPlateNumber(int plateNumber) {
        this.plateNumber = plateNumber;
    }

    public int getRouteNumber() {
        return routeNumber;
    }

    public void setRouteNumber(int routeNumber) {
        this.routeNumber = routeNumber;
    }

    public String getBrand() {
        return brand;
```

```
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public double getMileage() {
        return mileage;
    }

    public void setMileage(double mileage) {
        this.mileage = mileage;
    }

    public boolean isParked() {
        return parked;
    }

    public void setParked(boolean parked) {
        this.parked = parked;
    }

}
```

data.json

```
[

    {
        "driver": "Ilya Kulinkovich",
        "plateNumber": 143091,
        "routeNumber": 11,
        "brand": "Bugatti",
        "year": "2017",
        "mileage": 80.3,
        "parked": false
    },

    {
        "driver": "Daniil Kabachuk",
        "plateNumber": 43143,
        "routeNumber": 9,
        "brand": "MAZ",
        "year": "1999",
        "mileage": 30123.32,
        "parked": true
    },

    {
        "driver": "Anastasia Kovalyova",
        "plateNumber": 32601,
        "routeNumber": 22,
```

```
        "brand": "BMW",
        "year": "2018",
        "mileage": 302.2,
        "parked": false
    }

]
```

live.ilyusha.spp3.Main

```
package live.ilyusha.spp3;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Scanner;
import com.fasterxml.jackson.databind.ObjectMapper;

public class Main {

    private static final ObjectMapper mapper = new ObjectMapper();
    private static final Scanner scanner = new Scanner(System.in);
    private static ArrayList<Bus> buses = new ArrayList<>();

    public static void main(String[] args) throws IOException {
        if (args.length > 0 && args[0].equals("1")) {
            task1();
        } else if (args.length > 0 && args[0].equals("2")) {
            task2();
        } else {
            throw new IllegalArgumentException();
        }
    }

    private static void task1() {
        Triangle t1 = new Triangle(2,3,4);
        t1.log();

        Triangle t2 = new Triangle(3,4,5);
        t2.log();

        Triangle t3 = new Triangle(3, 6, 5);
        System.out.printf("%s and %s are%s equal", t2, t3, t2.equals(t3) ? ""
: "n't");
    }

    public static void task2() throws IOException {
        showMenu();
        buses = readInfo("resources/data.json");
        var actions = new Runnable[] {
            Main::listBuses,
            Main::listEnRouteBuses,
            Main::listParkedBuses,
            Main::lookUpBusesByRoute,
            Main::listsBusesOperating10Years,
            Main::listBusesOperating10000Km
        };
        while (true) {
            int i = Integer.parseInt(scanner.nextLine());
```

```
            if (i > 0 && i <= actions.length) {
                actions[i - 1].run();
            } else {
                System.out.println("User abort");
                break;
            }
        }
    }

    /* task2 methods */

    private static void showMenu() {
        System.out.println("[0] exit");
        System.out.println("[1] list buses");
        System.out.println("[2] list en route buses");
        System.out.println("[3] list parked buses");
        System.out.println("[4] look up buses by route");
        System.out.println("[5] list buses operating for ≥ 10 years");
        System.out.println("[6] list buses operating for ≥ 100000 km");
    }

    private static void listBuses() {
        for (Bus bus: buses) {
            System.out.println(bus.toString());
        }
    }

    private static void listEnRouteBuses() {
        for (Bus bus: buses) {
            if (!bus.isParked()) {
                System.out.println(bus);
            }
        }
    }

    private static void listParkedBuses() {
        for (Bus bus: buses) {
            if (bus.isParked()) {
                System.out.println(bus);
            }
        }
    }

    private static void lookUpBusesByRoute() {
        System.out.println("Enter the route:");
        int route = Integer.parseInt(scanner.nextLine());

        for (Bus bus: buses) {
            if (bus.getRouteNumber() == route) {
                System.out.println(bus);
            }
        }
    }

    private static void listsBusesOperating10Years() {
        int year = Calendar.getInstance().get(Calendar.YEAR);
        for (Bus bus: buses) {
            if (year - bus.getYear() >= 10) {
                System.out.println(bus);
            }
```

```
        }
    }

    private static void listBusesOperating10000Km() {
        for (Bus bus: buses) {
            if (bus.getMileage() >= 100000) {
                System.out.println(bus);
            }
        }
    }

    private static ArrayList<Bus> readInfo(String fileName) throws
IOException {
        return mapper.readValue(
            new File(fileName),
            mapper.getTypeFactory().constructCollectionType(ArrayList.class,
Bus.class)
        );
    }

}
```

Спецификация ввода

```
>java Main <номер задачи>
```

Пример

```
>java Main 1
```

Спецификация вывода

Для задачи 1:

```
<объект 1> and <объект 2> are[n't] equal
```

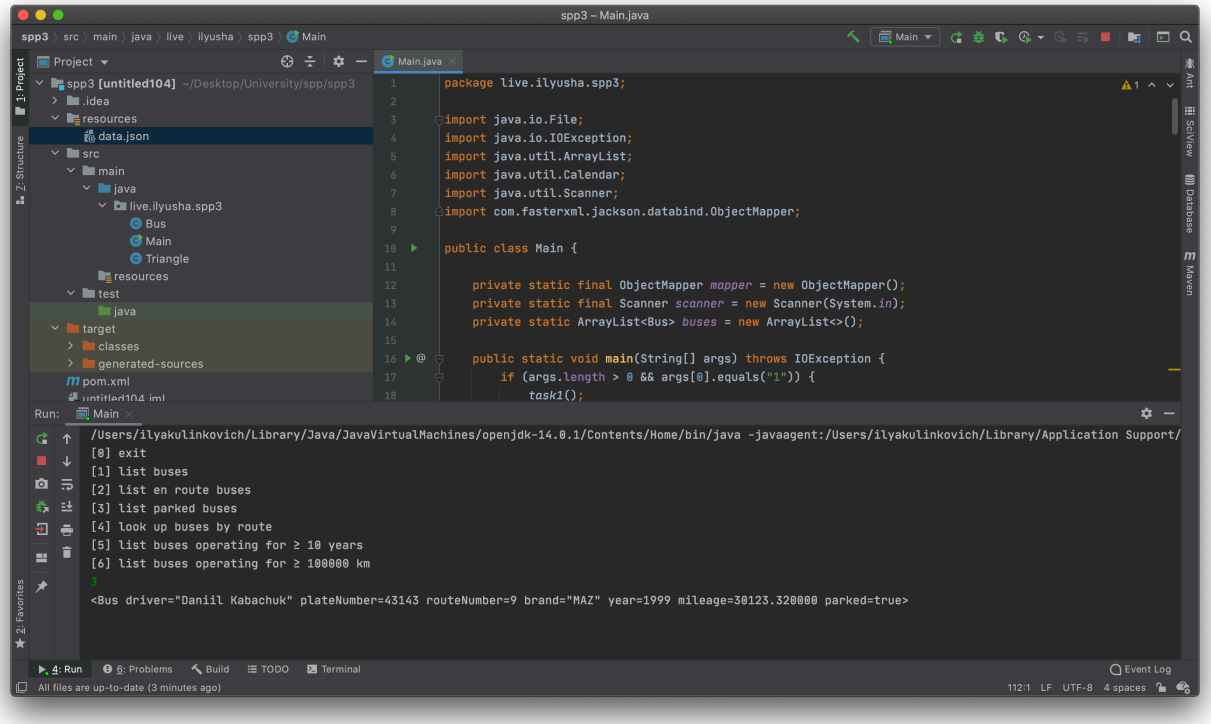
Для задачи 2:

```
<номер опции 1> опция 1
<номер опции 2> опция 2
…
<номер опции N> опция N
```

Пример

```
[0] exit
[1] list buses
[2] list en route buses
[3] list parked buses
[4] look up buses by route
[5] list buses operating for ≥ 10 years
[6] list buses operating for ≥ 100000 km
```

Рисунки с результатами работы программы



Вывод

В данной лабораторной работе я научился создавать и использовать классы в программах на языке программирования Java.