МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине: «Современные платформы программирования»

Выполнил:
Студент ФЭИС
3-го курса, группы ПО-5
Прокопчик Е.А.
Проверил:
Крощенко А.А.

Брест 2021

Цель работы:

научиться создавать и использовать классы в программах на языке программирования Java

Ход работы:

Вариант 12

Задание 1

Равносторонний 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.

Код:

Class Main:

```
public class Main {

    public static void main(String[] args) throws InvalidTriangleException {
        EquilateralTriangle firstTriangle = new EquilateralTriangle(13);

        System.out.println(firstTriangle.getLegLength());
        System.out.println(firstTriangle.getPerimeter());
        System.out.println(firstTriangle.getArea());
        System.out.println(firstTriangle);

        EquilateralTriangle secondTriangle = new EquilateralTriangle(13);
        EquilateralTriangle thirdTriangle = new EquilateralTriangle(14);

        System.out.println(firstTriangle.equals(secondTriangle));
        System.out.println(firstTriangle.equals(thirdTriangle));
    }
}
```

class EquilateralTriangle:

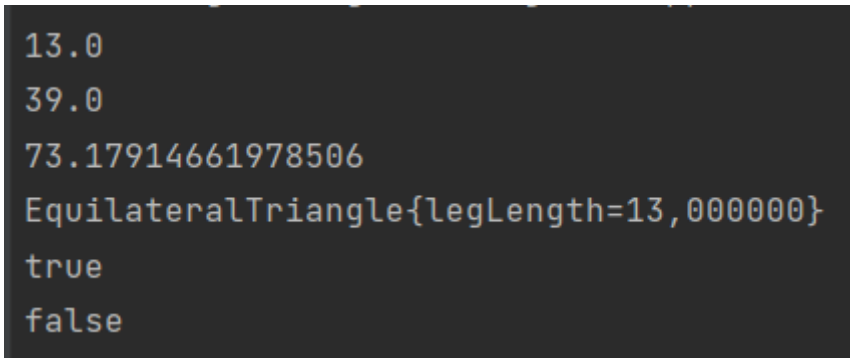
```
public class EquilateralTriangle
{
    private double legLength;

    public EquilateralTriangle(double legLength) throws
InvalidTriangleException {
        ThrowExceptionIfSideLengthIsInvalid(legLength);

        this.legLength = legLength;
    }

    public double getLegLength() {
        return legLength;
    }
```

```
    public double getPerimeter() {
        return legLength * 3;
    }

    public double getArea() {
        return ((Math.pow(legLength,2) * Math.sqrt(3))/4);
    }

    @Override
    public String toString() {
        return String.format("EquilateralTriangle{legLength=%f}", legLength);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }

        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }

        EquilateralTriangle equilateralTriangle = (EquilateralTriangle) obj;

        return Double.compare(equilateralTriangle.legLength, legLength) == 0;
    }

    private boolean isTriangleValid(double legLength) {
        return legLength > 0;
    }

    private void ThrowExceptionIfSideLengthIsInvalid(double sideLength) {
        if (sideLength <= 0) {
            throw new IllegalArgumentException("Side length must be greater
than 0. Provided value: " + sideLength);
        }
    }

    private void ThrowExceptionIfTriangleIsInvalid(double legLength) throws
InvalidTriangleException {
        if (!isTriangleValid(legLength)) {
            throw new InvalidTriangleException("Equilateral triangle is valid
if length of its side its greater than 0. " +
                    String.format("Provided values: leg length: %f",
legLength));
        }
    }
}
```

Class EquilateralTriangleException:

```
public class InvalidTriangleException extends Exception
{
    public InvalidTriangleException(String message)
    {
        super(message);
    }
}
```

Результат:



Задание 2

Автоматизированная система проката автомобилей Составить программу, которая хранит и обрабатывает информацию о прокате автомобилей. О каждом автомобиле (Car) содержится следующая информация:

• id;

• Марка;

• Модель;

• Год выпуска;

• Цвет;

• Цена;

• Регистрационный номер;

• Номер машины.

• ФИО лица, взявшего на прокат (при наличии);

• Номер паспорта лица-арендатора (при наличии).

Программа должна обеспечить вывод списков:

• автомобилей;

• автомобилей заданной марки;

• автомобилей заданной модели, которые эксплуатируются больше n лет;

• автомобилей заданного года выпуска, цена которых больше указанной;

• автомобилей, взятых на прокат;

• автомобилей, взятых на прокат с выводом личной информации об арендаторах

Код:

```
import java.io.*;
import java.util.*;
import java.util.Scanner;
import java.util.stream.Collectors;

public class Autopark
{
    private Car[] cars;

    Map<String, Integer> data = Map.of(
            "id", 0,
            "brand", 1,
            "model", 2,
            "year", 3,
            "color", 4,
            "price", 5,
            "regNumber", 6,
            "number", 7,
            "fioClient", 8,
            "idClient", 9
    );

    public Autopark(int size) {
       cars = new Car[size];
    }
    public static void main(String[] args) {
        List<String> sentences = Input_File("file.txt");
        int numberOfCars = sentences.size();
        String[][] array = new String[numberOfCars][];

        int i = 0;
        for (String str : sentences) {
            array[i++] = str.split("\\s");
        }

        Autopark lb = new Autopark(numberOfCars);
        lb.testAutopark(array);
    }

    void ListOfCars(Car[] Cars) {
        for (Car car : Cars) {
            System.out.print(car.getBrand() + " " + car.getModel() + " | ");
        }
    }

    void ListOfBrand(Car[] Cars){
        Scanner in = new Scanner(System.in);
        String b = in.next();
        for (Car car : Cars) {
            String carBrand = car.getBrand();
            if (Objects.equals(carBrand, b)) {
                System.out.print(car.getModel() + " | ");
            }
        }
    }

    void OlderCars(Car[] Cars) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int today = 2021;
        for (Car car : Cars) {
            int carYear = car.getYear();
```

```
            if (today - carYear >= n) {
                System.out.print(car.getBrand() + " " + car.getModel() + " |
");
            }
        }
    }

    void PriceYearCars( Car[] Cars){
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int p = in.nextInt();
        for (Car car : Cars) {
            int carYear = car.getYear();
            int carPrice = car.getPrice();
            if (n == carYear && carPrice > p ) {
                System.out.print(car.getBrand() + " " + car.getModel() + " |
");
            }
        }
    }


    void TakeCars(Car[] Cars) {
        for (Car car : Cars) {
            String FIOClient = car.getFIOClient();
            if (!FIOClient.equals("null")) {
                System.out.print(car.getBrand() + " " + car.getModel() + " |
");
            }
        }
    }

    void TakeCars_WithFIOClient(Car[] Cars) {
        for (Car car : Cars) {
            String FIOClient = car.getFIOClient();
            if (!FIOClient.equals("null")) {
                System.out.print(car.getBrand() + " " + car.getModel() + ": "
+ FIOClient + " | ");
            }
        }
    }

    void testAutopark(String[][] arr) {
        for (int i = 0; i < arr.length; i++) {
           cars[i] = new Car(arr[i][data.get("id")],
arr[i][data.get("brand")], arr[i][data.get("model")],
                    arr[i][data.get("year")], arr[i][data.get("color")],
arr[i][data.get("price")],
                    arr[i][data.get("regNumber")],
arr[i][data.get("number")], arr[i][data.get("fioClient")],
                    arr[i][data.get("idClient")]);
            System.out.println(cars[i].toString());
        }

        System.out.println("\n======== List of cars ========");
        ListOfCars(cars);

        System.out.println("\nInput car brand name: ");
        ListOfBrand(cars);

        System.out.print("\nInput how old your car should be: ");
        OlderCars(cars);

        System.out.print("\nInput car release year and its price: ");
```

```
        PriceYearCars(cars);

        System.out.println("\n======== Taken cars ========");
        TakeCars(cars);

        System.out.println("\n======== Clients fio ========");
        TakeCars_WithFIOClient(cars);

    }

    @Override
    public String toString() {
        return "AutoparkManager [cars=" + Arrays.toString(cars) + "]";
    }

    public static List<String> Input_File(String fileName) {
        List<String> sentences = new ArrayList<>();

        try(Scanner file = new Scanner(new File(fileName)))
        {
            while (file.hasNextLine()) {
                sentences.add(file.nextLine());
            }
            file.close();
        }
        catch(IOException ex){

            System.out.println(ex.getMessage());
        }
        return
sentences.stream().map(String::trim).collect(Collectors.toList());
    }
}

class Car {
    private String id, brand, model, year, color, price, regNumber, number,
fioClient, idClient;

    Car(String carID, String carBrand, String carName, String carYear, String
carColor,
         String carPrice, String carRegNumber, String carNumber, String
FIOClient, String IdClient) {
        this.id = carID;
        this.brand = carBrand;
        this.model = carName;
        this.year = carYear;
        this.color = carColor;
        this.price = carPrice;
        this.regNumber = carRegNumber;
        this.number = carNumber;
        this.fioClient = FIOClient;
        this.idClient = IdClient;
    }

    @Override
    public String toString() {
        return "Car [[id=" + this.id + "brand=" + this.brand + ", model=" +
this.model + ", year=" + this.year +
                "]\n      [color=" + this.color + ", price=" + this.price  +
", regNumber=" + this.regNumber +
                "]\n      [number=" + this.number + ", fioClient=" +
this.fioClient + ", idClient=" + this.idClient + "]]";
    }
```

```
    public String getModel() {
        return this.model;
    }
    public String getBrand() {
        return this.brand;
    }

    public Integer getYear() {
        return Integer.parseInt(this.year);
    }

    public Integer getPrice() {
        return Integer.parseInt(this.price);
    }

    public String getFIOClient() {
        return this.fioClient;
    }
}
```

File.txt

```
1 Audi A4 2011 silver 45 4598aa-7 11 ivanov 1234
2 Mazda RX-7 1997 yellow 70 1106jp-2 22 null null
3 Toyota Prius 2018 red 50 5695ao-1 33 prokopchik 1111
4 Audi Q7 2019 black 65 4786ae-4 44 null null
5 Mercedes-Benz e200 2021 black 70 1010aa-7 55 markov null
```

Результат:

```
======== List of cars ========
Audi A4 | Mazda RX-7 | Toyota Prius | Audi Q7 | Mercedes-Benz e200 |
Input car brand name: Audi
A4 | Q7 |
Input how old your car should be: 5
Audi A4 | Mazda RX-7 |
Input car release year and its price: 2021 60
Mercedes-Benz e200 |
======== Taken cars ========
Audi A4 | Toyota Prius | Mercedes-Benz e200 |
======== Clients fio ========
Audi A4: ivanov | Toyota Prius: vasyliev | Mercedes-Benz e200: markov |
```

Вывод: научился создавать и использовать классы в программах на языке программирования Java