МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ

по лабораторной работе №3

Выполнила:
студентка 3 курса
группы ПО-9
Бердникова В.А.

Проверил:
Крощенко А.А.

Брест 2024

Цель работы: научиться создавать и использовать классы в программах на языке программирования Java.

Вариант 2

Задание 1

Реализовать простой класс. ***Равносторонний треугольник, заданный длинами сторон***. Предусмотреть возможность определения площади и периметра, а также логический метод, определяющий существует или такой треугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Входные данные:

```
EquilateralTriangle triangle = new EquilateralTriangle(5.0,5.0,5.0);
EquilateralTriangle triangle2 = new EquilateralTriangle(7.0,9.0,12.0);
```

Выходные данные:

```
D:\ForJava\Java\jdk-17.0.5\bin\java.e
Длина сторон: [5.0, 5.0, 5.0]
Это равносторонний треугольник
Площадь: 10.825317547305483
Периметр: 15.0
Существует такой треугольник: true

Длина сторон: [7.0, 9.0, 12.0]
Это неравносторонний треугольник
Площадь: 31.304951684997057
Периметр: 28.0
Существует такой треугольник: true

Треугольники равны: false
```

Код программы:

Main.java

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {

        EquilateralTriangle triangle = new EquilateralTriangle(5.0,5.0,5.0);
        System.out.println("Длина сторон: " +
Arrays.toString(triangle.getSideLengths()));
        System.out.println(triangle);
        System.out.println("Площадь: " + triangle.calculateArea());
        System.out.println("Периметр: " + triangle.calculatePerimeter());
```

```
        System.out.println("Существует такой треугольник: " +
triangle.isTriangle() + "\n");

        EquilateralTriangle triangle2 = new
EquilateralTriangle(7.0,9.0,12.0);
        System.out.println("Длина сторон: " +
Arrays.toString(triangle2.getSideLengths()));
        System.out.println(triangle2);
        System.out.println("Площадь: " + triangle2.calculateArea());
        System.out.println("Периметр: " + triangle2.calculatePerimeter());
        System.out.println("Существует такой треугольник: " +
triangle2.isTriangle());

        System.out.println("\nТреугольники равны: " +
triangle.equals(triangle2));
    }
}
```

EquilateralTriangle.java

```
public class EquilateralTriangle {
    private final double side1;
    private final double side2;
    private final double side3;

    public EquilateralTriangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    public boolean isEquilateral() {
        return side1 == side2 && side2 == side3;
    }


    public double[] getSideLengths() {
        return new double[] {side1, side2, side3};
    }


    public double calculatePerimeter() {
        if (!isTriangle()) {
            System.out.println("Треугольник не существует. Невозможно
вычислить периметр.");
            return 0;
        }
        return side1 + side2 + side3;
    }


    public double calculateArea() {
        if (!isTriangle()) {
            System.out.println("Треугольник не существует. Невозможно
вычислить площадь.");
            return 0;
        }
        double s = calculatePerimeter() / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }


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

```
        if (obj == null || getClass() != obj.getClass()) return false;
        EquilateralTriangle that = (EquilateralTriangle) obj;
        return Double.compare(that.side1, side1) == 0 &&
                Double.compare(that.side2, side2) == 0 &&
                Double.compare(that.side3, side3) == 0;
    }


    @Override
    public String toString() {
        return isEquilateral()?"Это равносторонний треугольник" : "Это
неравносторонний треугольник";
    }


    public boolean isTriangle() {
        boolean condition1 = (side1 + side2 > side3);
        boolean condition2 = (side2 + side3 > side1);
        boolean condition3 = (side1 + side3 > side2);
        return condition1 && condition2 && condition3;
    }
}
```

Задание 2

Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных. 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.

Входные данные:

inputCars.txt

```
1,Toyota,Camry,2018,Black,20000,ABC123456,RUS1234,John Doe,1234567890
2,Honda,Accord,2017,White,18000,XYZ987654,RUS5678,Alice Smith,0987654321
3,Ford,Focus,2016,Blue,15000,DEF456789,RUS4321,
4,Toyota,Corolla,2015,Red,17000,GHI987654,RUS8765,Robert Johnson,1357924680
5,BMW,X5,2019,Silver,30000,JKL345678,RUS2468,
6,Toyota,Camry,2017,Gray,19000,MNO654321,RUS1357,Sarah Williams,2468135790
7,Honda,Civic,2018,Black,16000,PQR987654,RUS3579,
8,Ford,Fusion,2014,White,14000,STU456789,RUS7890,Michael Brown,9876543210
9,BMW,3 Series,2019,Blue,28000,VWX123456,RUS5793,Emily Davis,6543210987
10,Toyota,Rav4,2016,Green,20000,YZT789012,RUS2468,
```

Выходные данные:

All cars:

id=1, brand='Toyota', model='Camry', year=2018, color='Black', price=20000.0, registrationNumber='ABC123456', plateNumber='RUS1234', renterName='John Doe', renterPassport='1234567890'
id=2, brand='Honda', model='Accord', year=2017, color='White', price=18000.0, registrationNumber='XYZ987654', plateNumber='RUS5678', renterName='Alice Smith', renterPassport='0987654321'
id=3, brand='Ford', model='Focus', year=2016, color='Blue', price=15000.0, registrationNumber='DEF456789', plateNumber='RUS4321'
id=4, brand='Toyota', model='Corolla', year=2015, color='Red', price=17000.0, registrationNumber='GHI987654', plateNumber='RUS8765', renterName='Robert Johnson', renterPassport='1357924680'
id=5, brand='BMW', model='X5', year=2019, color='Silver', price=30000.0, registrationNumber='JKL345678', plateNumber='RUS2468'
id=6, brand='Toyota', model='Camry', year=2017, color='Gray', price=19000.0, registrationNumber='MNO654321', plateNumber='RUS1357', renterName='Sarah Williams', renterPassport='2468135790'
id=7, brand='Honda', model='Civic', year=2018, color='Black', price=16000.0, registrationNumber='PQR987654', plateNumber='RUS3579'
id=8, brand='Ford', model='Fusion', year=2014, color='White', price=14000.0, registrationNumber='STU456789', plateNumber='RUS7890', renterName='Michael Brown', renterPassport='9876543210'
id=9, brand='BMW', model='3 Series', year=2019, color='Blue', price=28000.0, registrationNumber='VWX123456', plateNumber='RUS5793', renterName='Emily Davis', renterPassport='6543210987'
id=10, brand='Toyota', model='Rav4', year=2016, color='Green', price=20000.0, registrationNumber='YZT789012', plateNumber='RUS2468'

Cars by brand 'Toyota':
id=1, brand='Toyota', model='Camry', year=2018, color='Black', price=20000.0, registrationNumber='ABC123456', plateNumber='RUS1234', renterName='John Doe', renterPassport='1234567890'
id=4, brand='Toyota', model='Corolla', year=2015, color='Red', price=17000.0, registrationNumber='GHI987654', plateNumber='RUS8765', renterName='Robert Johnson', renterPassport='1357924680'
id=6, brand='Toyota', model='Camry', year=2017, color='Gray', price=19000.0, registrationNumber='MNO654321', plateNumber='RUS1357', renterName='Sarah Williams', renterPassport='2468135790'
id=10, brand='Toyota', model='Rav4', year=2016, color='Green', price=20000.0, registrationNumber='YZT789012', plateNumber='RUS2468'

Cars by model 'Camry' and more than 5 years old:
id=1, brand='Toyota', model='Camry', year=2018, color='Black', price=20000.0, registrationNumber='ABC123456', plateNumber='RUS1234', renterName='John Doe', renterPassport='1234567890'
id=6, brand='Toyota', model='Camry', year=2017, color='Gray', price=19000.0, registrationNumber='MNO654321', plateNumber='RUS1357', renterName='Sarah Williams', renterPassport='2468135790'

Cars of 2018 with price more than $15000:
id=1, brand='Toyota', model='Camry', year=2018, color='Black', price=20000.0, registrationNumber='ABC123456', plateNumber='RUS1234', renterName='John Doe', renterPassport='1234567890'
id=7, brand='Honda', model='Civic', year=2018, color='Black', price=16000.0, registrationNumber='PQR987654', plateNumber='RUS3579'

Rented cars:
id=1, brand='Toyota', model='Camry', year=2018, color='Black', price=20000.0, registrationNumber='ABC123456', plateNumber='RUS1234', renterName='John Doe', renterPassport='1234567890'
id=2, brand='Honda', model='Accord', year=2017, color='White', price=18000.0, registrationNumber='XYZ987654', plateNumber='RUS5678', renterName='Alice Smith', renterPassport='0987654321'
id=4, brand='Toyota', model='Corolla', year=2015, color='Red', price=17000.0, registrationNumber='GHI987654', plateNumber='RUS8765', renterName='Robert Johnson', renterPassport='1357924680'
id=6, brand='Toyota', model='Camry', year=2017, color='Gray', price=19000.0, registrationNumber='MNO654321', plateNumber='RUS1357', renterName='Sarah Williams', renterPassport='2468135790'

id=8, brand='Ford', model='Fusion', year=2014, color='White', price=14000.0, registrationNumber='STU456789', plateNumber='RUS7890', renterName='Michael Brown', renterPassport='9876543210'
id=9, brand='BMW', model='3 Series', year=2019, color='Blue', price=28000.0, registrationNumber='VWX123456', plateNumber='RUS5793', renterName='Emily Davis', renterPassport='6543210987'

Rented cars with renter information:
id=1, brand='Toyota', model='Camry', year=2018, color='Black', price=20000.0, registrationNumber='ABC123456', plateNumber='RUS1234', renterName='John Doe', renterPassport='1234567890'
id=2, brand='Honda', model='Accord', year=2017, color='White', price=18000.0, registrationNumber='XYZ987654', plateNumber='RUS5678', renterName='Alice Smith', renterPassport='0987654321'
id=4, brand='Toyota', model='Corolla', year=2015, color='Red', price=17000.0, registrationNumber='GHI987654', plateNumber='RUS8765', renterName='Robert Johnson', renterPassport='1357924680'
id=6, brand='Toyota', model='Camry', year=2017, color='Gray', price=19000.0, registrationNumber='MNO654321', plateNumber='RUS1357', renterName='Sarah Williams', renterPassport='2468135790'
id=8, brand='Ford', model='Fusion', year=2014, color='White', price=14000.0, registrationNumber='STU456789', plateNumber='RUS7890', renterName='Michael Brown', renterPassport='9876543210'
id=9, brand='BMW', model='3 Series', year=2019, color='Blue', price=28000.0, registrationNumber='VWX123456', plateNumber='RUS5793', renterName='Emily Davis', renterPassport='6543210987'

Код программы:

Car.java

```
class Car {
    private final int id;
    private final String brand;
    private final String model;
    private final int year;
    private final String color;
    private final double price;
    private final String registrationNumber;
    private final String plateNumber;
    private final String renterName;
    private final String renterPassport;

    public Car(int id, String brand, String model, int year, String color,
double price, String registrationNumber, String plateNumber, String
renterName, String renterPassport) {
        this.id = id;
        this.brand = brand;
        this.model = model;
        this.year = year;
```

```
        this.color = color;
        this.price = price;
        this.registrationNumber = registrationNumber;
        this.plateNumber = plateNumber;
        this.renterName = renterName;
        this.renterPassport = renterPassport;
    }


    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }


    public double getPrice() {
        return price;
    }


    public String getRenterName() {
        return renterName;
    }



    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("id=").append(id)
                .append(", brand='").append(brand).append('\'')
                .append(", model='").append(model).append('\'')
                .append(", year=").append(year)
                .append(", color='").append(color).append('\'')
                .append(", price=").append(price)
                .append(",
registrationNumber='").append(registrationNumber).append('\'')
                .append(", plateNumber='").append(plateNumber).append('\'');

        if (!renterName.isEmpty()) {
            builder.append(", renterName='").append(renterName).append('\'');
        }
        if (!renterPassport.isEmpty()) {
            builder.append(",
renterPassport='").append(renterPassport).append('\'');
        }

        return builder.toString();
    }

}
```

Test.java

```
public class Test {
    public static void main(String[] args) {
        RentalSystem rentalSystem = new RentalSystem();
        System.out.println("All cars:");
```

```
        rentalSystem.printAllCars();
        System.out.println("\nCars by brand 'Toyota':");
        rentalSystem.printCarsByBrand("Toyota");
        System.out.println("\nCars by model 'Camry' and more than 5 years
old:");
        rentalSystem.printCarsByModelAndAge("Camry", 5);
        System.out.println("\nCars of 2018 with price more than $15000:");
        rentalSystem.printCarsByYearAndPrice(2018, 15000);
        System.out.println("\nRented cars:");
        rentalSystem.printRentedCars();
        System.out.println("\nRented cars with renter inf:");
        rentalSystem.printRentedCarsWithRenterInfo();
    }
}
```

RentalSystem.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class RentalSystem {
    private List<Car> cars;

    public RentalSystem() {
        cars = new ArrayList<>();
        String fileName =
"C:\\Users\\user\\IdeaProjects\\untitled2\\src\\inputcars.txt";
        loadCarsFromFile(fileName);
    }

    private void loadCarsFromFile(String fileName) {
        try {
            File file = new File(fileName);
            Scanner scanner = new Scanner(file);
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] parts = line.split(",");
                int id = Integer.parseInt(parts[0]);
                String brand = parts[1];
                String model = parts[2];
                int year = Integer.parseInt(parts[3]);
                String color = parts[4];
                double price = Double.parseDouble(parts[5]);
                String registrationNumber = parts[6];
                String plateNumber = parts[7];
                String renterName = "";
                String renterPassport = "";
                if (parts.length > 8) {
                    renterName = parts[8];
                    renterPassport = parts[9];
                }
                cars.add(new Car(id, brand, model, year, color, price,
registrationNumber, plateNumber, renterName, renterPassport));
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.err.println("File not found: " + fileName);
            e.printStackTrace();
        }
    }
```

```
    public void printAllCars() {
        for (Car car : cars) {
            System.out.println(car);
        }
    }

    public void printCarsByBrand(String brand) {
        for (Car car : cars) {
            if (car.getBrand().equalsIgnoreCase(brand)) {
                System.out.println(car);
            }
        }
    }

    public void printCarsByModelAndAge(String model, int age) {
        int currentYear = 2024;
        for (Car car : cars) {
            if (car.getModel().equalsIgnoreCase(model) && (currentYear -
car.getYear()) > age) {
                System.out.println(car);
            }
        }
    }

    public void printCarsByYearAndPrice(int year, double minPrice) {
        for (Car car : cars) {
            if (car.getYear() == year && car.getPrice() > minPrice) {
                System.out.println(car);
            }
        }
    }

    public void printRentedCars() {
        for (Car car : cars) {
            if (!car.getRenterName().isEmpty()) {
                System.out.println(car);
            }
        }
    }

    public void printRentedCarsWithRenterInfo() {
        for (Car car : cars) {
            if (!car.getRenterName().isEmpty()) {
                System.out.println(car);
            }
        }
    }
}
```

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