

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

Лабораторная работа №3

По дисциплине: «Современные платформы программирования»

Тема: «Классы в Java»

Выполнил:

Студент 3 курса

Группы ПО-8

Бувин Д.А.

Проверил:

А. А. Крощенко

Брест, 2024

Лабораторная работа №3

Вариант 3

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

Задание №1:

- Реализовать пользовательский класс по варианту.
- Создать другой класс с методом main, в котором будут находиться 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
- Переопределить методы toString() и equals()

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

Код программы:

```
import java.util.Scanner;
```

```
public class Task_1 {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.println("Введите длину стороны A: ");  
        double sideA = scanner.nextDouble();  
  
        System.out.println("Введите длину стороны B: ");  
        double sideB = scanner.nextDouble();  
    }  
}
```

```

System.out.println("Введите длину стороны C: ");
double sideC = scanner.nextDouble();

RightTriangle triangle = new RightTriangle(sideA, sideB, sideC);

System.out.println("Длины сторон исходного треугольника:");
System.out.println("Сторона A: " + triangle.getSideA());
System.out.println("Сторона B: " + triangle.getSideB());
System.out.println("Сторона C: " + triangle.getSideC());

System.out.println("Треугольник существует: " + triangle.isRightTriangle());

System.out.println("Площадь треугольника: " + triangle.calculateArea());

System.out.println("Периметр треугольника: " + triangle.calculatePerimeter());

RightTriangle anotherTriangle = new RightTriangle(3, 4, 5);

System.out.println("Длины сторон второго треугольника:");
System.out.println("Сторона A: " + anotherTriangle.getSideA());
System.out.println("Сторона B: " + anotherTriangle.getSideB());
System.out.println("Сторона C: " + anotherTriangle.getSideC());
    System.out.println("Треугольники исходный и второй равны: " +
triangle.equals(anotherTriangle));

RightTriangle anotherTriangle1 = new RightTriangle(4, 3, 5);

System.out.println("Длины сторон третьего треугольника:");
System.out.println("Сторона A: " + anotherTriangle1.getSideA());
System.out.println("Сторона B: " + anotherTriangle1.getSideB());
System.out.println("Сторона C: " + anotherTriangle1.getSideC());
    System.out.println("Треугольники исходный и третий равны: " +
triangle.equals(anotherTriangle1));
}
}
class RightTriangle {
    private double sideA;
    private double sideB;
    private double sideC;

    public RightTriangle(double sideA, double sideB, double sideC) {
        this.sideA = sideA;
        this.sideB = sideB;

```

```

        this.sideC = sideC;
    }

    public double getSideA() {
        return sideA;
    }

    public void setSideA(double sideA) {
        this.sideA = sideA;
    }

    public double getSideB() {
        return sideB;
    }

    public void setSideB(double sideB) {
        this.sideB = sideB;
    }

    public double getSideC() {
        return sideC;
    }

    public void setSideC(double sideC) {
        this.sideC = sideC;
    }

    public boolean isRightTriangle() {
        return sideA * sideA + sideB * sideB == sideC * sideC ||
            sideA * sideA + sideC * sideC == sideB * sideB ||
            sideB * sideB + sideC * sideC == sideA * sideA;
    }

    public double calculateArea() {
        return 0.5 * sideA * sideB;
    }

    public double calculatePerimeter() {
        return sideA + sideB + sideC;
    }

    @Override
    public String toString() {
        return "RightTriangle{" +

```

```

        "sideA=" + sideA +
        ", sideB=" + sideB +
        ", sideC=" + sideC +
        '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        RightTriangle that = (RightTriangle) o;

        if (Double.compare(that.sideA, sideA) != 0) return false;
        if (Double.compare(that.sideB, sideB) != 0) return false;
        return Double.compare(that.sideC, sideC) == 0;
    }

    @Override
    public int hashCode() {
        int result;
        long temp;
        temp = Double.doubleToLongBits(sideA);
        result = (int) (temp ^ (temp >>> 32));
        temp = Double.doubleToLongBits(sideB);
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        temp = Double.doubleToLongBits(sideC);
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        return result;
    }
}

```

Результат работы:

```
Введите длину стороны A:
3
Введите длину стороны B:
4
Введите длину стороны C:
5
Длины сторон исходного треугольника:
Сторона A: 3.0
Сторона B: 4.0
Сторона C: 5.0
Треугольник существует: true
Площадь треугольника: 6.0
Периметр треугольника: 12.0
Длины сторон второго треугольника:
Сторона A: 3.0
Сторона B: 4.0
Сторона C: 5.0
Треугольники исходный и второй равны: true
Длины сторон третьего треугольника:
Сторона A: 4.0
Сторона B: 3.0
Сторона C: 5.0
Треугольники исходный и третий равны: false

Process finished with exit code 0
```

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).

Программа должна обеспечивать:

- Формирование данных обо всех автобусах в виде списка;
- Формирование списка автобусов выехавших из парка;
- Формирование списка автобусов оставшихся в парке;
- Список автобусов для заданного номера маршрута;
- Список автобусов, которые эксплуатируются больше 10 лет;
- Список автобусов, пробег у которых больше 100000 км.
- Вывод сведений об автобусах, находящихся на маршруте и об автобусах, оставшихся в парке.

Код программы:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

class Bus {
    private String driverName;
    private String busNumber;
    private int routeNumber;
    private String brand;
    private int yearOfOperation;
    private double mileage;
    private String location;

    public Bus(String driverName, String busNumber, int routeNumber, String brand, int
yearOfOperation, double mileage, String location) {
        this.driverName = driverName;
        this.busNumber = busNumber;
        this.routeNumber = routeNumber;
        this.brand = brand;
        this.yearOfOperation = yearOfOperation;
        this.mileage = mileage;
        this.location = location;
    }
}
```

```
@Override
public String toString() {
    return "Bus{" +
        "driverName=\"" + driverName + "\" +
        ", busNumber=" + busNumber +
        ", routeNumber=" + routeNumber +
        ", brand=\"" + brand + "\" +
        ", yearOfOperation=" + yearOfOperation +
        ", mileage=" + mileage +
        ", location=\"" + location + "\" +
        '}'";
}

public int getRouteNumber() {
    return routeNumber;
}

public int getYearOfOperation() {
    return yearOfOperation;
}

public double getMileage() {
    return mileage;
}

public String getLocation() {
    return location;
}

public String getDriverName() {
    return driverName;
}

public void setDriverName(String driverName) {
    this.driverName = driverName;
}

public String getBusNumber() {
    return busNumber;
}

public void setBusNumber(String busNumber) {
    this.busNumber = busNumber;
}
```



```

    }

    public String getBrand() {
        return brand;
    }

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

public class Task_2 {
    private List<Bus> buses;

    public Task_2() {
        this.buses = new ArrayList<>();
    }

    public void readFromFile(String fileName) {
        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] data = line.split(";");
                buses.add(parseBusData(data));
            }
        } catch (IOException e) {
            System.err.println("Ошибка чтения файла: " + e.getMessage());
        }
    }

    private Bus parseBusData(String[] data) {
        String driverName = data[0];
        String busNumber = data[1];
        int routeNumber = Integer.parseInt(data[2]);
        String brand = data[3];
        int yearOfOperation = Integer.parseInt(data[4]);
        double mileage = Double.parseDouble(data[5]);
        String location = data[6];
        return new Bus(driverName, busNumber, routeNumber, brand, yearOfOperation,
mileage, location);
    }

    public List<Bus> getBuses() {
        return buses;
    }
}

```

```

    }

    public List<Bus> getBusesOnRoute() {
        return getBusesByLocation("маршрут");
    }

    public List<Bus> getBusesInPark() {
        return getBusesByLocation("парк");
    }

    private List<Bus> getBusesByLocation(String location) {
        List<Bus> filteredBuses = new ArrayList<>();
        for (Bus bus : buses) {
            if (bus.getLocation().equalsIgnoreCase(location)) {
                filteredBuses.add(bus);
            }
        }
        return filteredBuses;
    }

    public List<Bus> getBusesByRouteNumber(int routeNumber) {
        List<Bus> filteredBuses = new ArrayList<>();
        for (Bus bus : buses) {
            if (bus.getRouteNumber() == routeNumber) {
                filteredBuses.add(bus);
            }
        }
        return filteredBuses;
    }

    public List<Bus> getBusesMoreThanTenYearsInOperation() {
        List<Bus> filteredBuses = new ArrayList<>();
        for (Bus bus : buses) {
            if (2024 - bus.getYearOfOperation() > 10) {
                filteredBuses.add(bus);
            }
        }
        return filteredBuses;
    }

    public List<Bus> getBusesWithMileageMoreThanHundredThousand() {
        List<Bus> filteredBuses = new ArrayList<>();
        for (Bus bus : buses) {
            if (bus.getMileage() > 100000) {

```

```


        filteredBuses.add(bus);
    }
}
return filteredBuses;
}

public static void main(String[] args) {
    Task_2 busPark = new Task_2();
    busPark.readFromFile("buses.txt");

    System.out.println("Список всех автобусов:");
    for (Bus bus : busPark.getBuses()) {
        System.out.println(bus);
        System.out.println("Driver Name: " + bus.getDriverName());
        System.out.println("Bus Number: " + bus.getBusNumber());
        System.out.println("Route Number: " + bus.getRouteNumber());
        System.out.println("Brand: " + bus.getBrand());
        System.out.println("Year of Operation: " + bus.getYearOfOperation());
        System.out.println("Mileage: " + bus.getMileage());
        System.out.println("Location: " + bus.getLocation());
        System.out.println();
    }
}
}

```

Текстовый файл:

 buses – Блокнот

Файл Правка Формат Вид Справка

```

John Doe;123;456;Mercedes;2018;500.00;City A
John Doe;123;456;Mercedes;2018;500.00;City B
John Doe;123;456;Mercedes;2018;500.00;City C

```

Результат работы:

```
Список всех автобусов:
Bus{driverName='John Doe', busNumber=123, routeNumber=456, brand='Mercedes', yearOfOperation=2018, mileage=500.0, location='City A'}
Driver Name: John Doe
Bus Number: 123
Route Number: 456
Brand: Mercedes
Year of Operation: 2018
Mileage: 500.0
Location: City A

Bus{driverName='John Doe', busNumber=123, routeNumber=456, brand='Mercedes', yearOfOperation=2018, mileage=500.0, location='City B'}
Driver Name: John Doe
Bus Number: 123
Route Number: 456
Brand: Mercedes
Year of Operation: 2018
Mileage: 500.0
Location: City B

Bus{driverName='John Doe', busNumber=123, routeNumber=456, brand='Mercedes', yearOfOperation=2018, mileage=500.0, location='City C'}
Driver Name: John Doe
Bus Number: 123
Route Number: 456
Brand: Mercedes
Year of Operation: 2018
Mileage: 500.0
Location: City C
```

Вывод: По итогу выполнения лабораторной работы, я научился создавать и использовать классы в программах на языке программирования Java.