Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
по дисциплине: **СПП**
Тема: Классы в Java

Выполнил
студент 3 курса
Корнасевич И. Д.

Проверил
Крощенко А. А.

Брест 2021

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

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.
- 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 так же логический метод, определяющий существует ли такой треугольник. Конструктор должен позволить создавать объекты с начальной инициали- зацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Triangle.java

```
1  package main3.task1;
2
3  import java.util.Objects;
4
5
6  public class Triangle {
7
8      private final double a;
9
10     private final double b;
11
12     private final double c;
13
14     public Triangle(double a, double b, double c) {
15         this.a = a;
16         this.b = b;
17         this.c = c;
18     }
19
20     public double area() {
21         var semiPerimeter = perimeter() / 2;
22         return Math.pow(semiPerimeter * (semiPerimeter - a) * (semiPerimeter - b) * (semiPerimeter -
       c), 0.5);
23     }
24
25     public double perimeter() {
26         return a + b + c;
27     }
28
29     @Override
30     public int hashCode() {
31         return Objects.hash(a, b, c);
32     }
33
34     @Override
35     public boolean equals(Object o) {
```

```
36          if (this == o) return true;
37          if (o == null || getClass() != o.getClass()) return false;
38          Triangle triangle = (Triangle) o;
39          return Double.compare(triangle.a, a) == 0 && Double.compare(triangle.b, b) == 0 && Double.
        compare(triangle.c, c) == 0;
40      }
41
42      @Override
43      public String toString() {
44          return "Triangle{" +
45                  "a=" + a +
46                  ", b=" + b +
47                  ", c=" + c +
48                  '}';
49      }
50
51      public boolean isExist() {
52          return a + b < c && b + c < a && a + c < b;
53      }
54  }
```

Задание 2 Разработать автоматизированную систему на основе некоторой структуры дан- ных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных

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.

App2.java

```java
package main3.task2;

import java.util.List;

import static java.lang.System.out;


public class App2 {

    public static void main(String[] args) {
        var park = new Park(
                List.of(
                        new Bus(new Driver("a", "b"), 1, 2000, "A", 50000, 1),
                        new Bus(new Driver("c", "d"), 2, 2010, "B", 30000, 2),
                        new Bus(new Driver("e", "f"), 3, 2015, "B", 10000),
                        new Bus(new Driver("g", "h"), 4, 2019, "C", 5000),
                        new Bus(new Driver("m", "n"), 5, 1992, "A", 100000, 3)
                )
        );
        // All buses
        out.println(park.find(x -> true));
        // All out of the park
        out.println(park.find(x -> x.route().isEmpty()));
        // All in the park
        out.println(park.find(x -> x.route().isPresent()));
        // Find by route
        out.println(park.find(x -> x.route().isPresent() && x.route().get() == 1));
        // Find by using
        out.println(park.find(x -> x.creationYear() < 2011));
        // Find by mileage
        out.println(park.find(x -> x.mileage() >= 100000));
    }
}
```

Bus.java

```java
package main3.task2;

import java.util.Objects;
import java.util.Optional;


public class Bus {

    private final Driver driver;

    private final int number;

    private final int creationYear;

    private final String brand;

    private final double mileage;

    private boolean isAssigned;

    private int route;

    public Bus(Driver driver, int number, int creationYear, String brand, double mileage) {
        this.driver = driver;
        this.number = number;
        this.creationYear = creationYear;
        this.brand = brand;
        this.mileage = mileage;
    }

    public Bus(Driver driver, int number, int creationYear, String brand, double mileage, int route) {
        this(driver, number, creationYear, brand, mileage);
        this.route = route;
        isAssigned = true;
    }

    @Override
    public int hashCode() {
        return Objects.hash(number, creationYear, brand, mileage, driver, isAssigned, route);
    }

    @Override
    public boolean equals(Object o) {
```

```
44          if (this == o) return true;
45          if (o == null || getClass() != o.getClass()) return false;
46          Bus bus = (Bus) o;
47          return number == bus.number && creationYear == bus.creationYear && Double.compare(bus.
        mileage, mileage) == 0 && isAssigned == bus.isAssigned && route == bus.route && Objects.equals(
        brand, bus.brand) && Objects.equals(driver, bus.driver);
48      }
49
50      @Override
51      public String toString() {
52          return "Bus{" +
53                  "driver=" + driver +
54                  ", number=" + number +
55                  ", creationYear=" + creationYear +
56                  ", brand='" + brand + '\'' +
57                  ", mileage=" + mileage +
58                  ", isAssigned=" + isAssigned +
59                  ", route=" + route +
60                  '}';
61      }
62
63      public String brand() {
64          return brand;
65      }
66
67      public double mileage() {
68          return mileage;
69      }
70
71      public int number() {
72          return number;
73      }
74
75      public int creationYear() {
76          return creationYear;
77      }
78
79      public void assign(int route) {
80          isAssigned = true;
81          this.route = route;
82      }
83
84      public void unassign() {
85          isAssigned = false;
86      }
87
88      public Optional<Integer> route() {
89          return isAssigned ? Optional.of(route) : Optional.empty();
90      }
91  }
```

Driver.java

```
1   package main3.task2;
2
3   import java.util.Objects;
4
5
6   public class Driver {
7
8       private final String name;
9
10      private final String surname;
11
12      public Driver(String name, String surname) {
13          this.name = name;
14          this.surname = surname;
15      }
16
17      @Override
18      public int hashCode() {
19          return Objects.hash(name, surname);
20      }
21
22      @Override
23      public boolean equals(Object o) {
24          if (this == o) return true;
25          if (o == null || getClass() != o.getClass()) return false;
26          Driver driver = (Driver) o;
27          return Objects.equals(name, driver.name) && Objects.equals(surname, driver.surname);
28      }
```

```
29
30      @Override
31      public String toString() {
32          return "Driver{" +
33                  "name='" + name + '\'' +
34                  ", surname='" + surname + '\'' +
35                  '}';
36      }
37  }
```

Park.java

```
1   package main3.task2;
2
3   import java.util.Collection;
4   import java.util.function.Predicate;
5   import java.util.stream.Collectors;
6
7
8   public class Park {
9
10      private final Collection<Bus> buses;
11
12      public Park(Collection<Bus> buses) {
13          this.buses = buses;
14      }
15
16      public Collection<Bus> find(Predicate<Bus> predicate) {
17          return buses.stream().filter(predicate).collect(Collectors.toList());
18      }
19  }
```

Вывод Я написал две системы классов, переопределив в во всех классах equals и hashcode.