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

Отчёт

по лабораторной работе №3

Выполнила:
студентка группы ПО-9
Матюшик Е.П.

Проверил:
Крощенко А. А.

Брест 2024

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

Вариант 3

Задание 1.

• 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.
• Переопределить методы toString() и equals().
Прямоугольный 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:

```
Введите длину стороны A:
10
Введите длину стороны B:
9
Введите длину стороны C:
12
```

Выходные данные:

```
Площадь треугольника: 45.0
Периметр треугольника: 31.0
Треугольник существует? - true
Длины сторон треугольника: {сторона A=10.0, сторона B=9.0, сторона C=12.0}
Треугольники равны? - true
```

Код программы:

```
import java.util.Scanner;

public class Task1 {
    private double sideA;
    private double sideB;
    private double sideC;
```

```
// Конструктор без параметров
public Task1() {
   this.sideA = 0;
   this.sideB = 0;
   this.sideC = 0;
}

// Конструктор с начальной инициализацией
public Task1(double sideA, double sideB, double sideC) {
   this.sideA = sideA;
   this.sideB = sideB;
   this.sideC = sideC;
}

// Метод определения площади
public double calculateArea() {
   return 0.5 * sideA * sideB;
}

// Метод определения периметра
public double calculatePerimeter() {
   return sideA + sideB + sideC;
}

// Логический метод, определяющий существует ли такой треугольник
public boolean exists() {
   return (sideA + sideB > sideC) && (sideA + sideC > sideB) && (sideB + sideC > sideA);
}

// Переопределение метода toString()
@Override
public String toString() {
   return "{сторона A=" + sideA + ", сторона B=" + sideB + ", сторона C=" + sideC + "}";
}

// Переопределение метода equals()
@Override
public boolean equals(Object obj) {
   if (this == obj) {
      return true;
   }
   if (obj == null || getClass() != obj.getClass()) {
      return false;
```

```
    }
    Task1 other = (Task1) obj;
    return Double.compare(this.sideA, other.sideA) == 0 &&
           Double.compare(this.sideB, other.sideB) == 0 &&
           Double.compare(this.sideC, other.sideC) == 0;
}

// Метод main для примеров использования
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();

    scanner.close();

    // Создание объекта треугольника
    Task1 triangle = new Task1(sideA, sideB, sideC);

    // Вычисление и вывод площади треугольника
    double area = triangle.calculateArea();
    System.out.println("Площадь треугольника: " + area);

    // Вычисление и вывод периметра треугольника
    double perimeter = triangle.calculatePerimeter();
    System.out.println("Периметр треугольника: " + perimeter);

    // Проверка существует ли треугольник
    boolean exists = triangle.exists();
    System.out.println("Треугольник существует? - " + exists);

    // Вывод информации о треугольнике с помощью toString()
    System.out.println("Длины сторон треугольника: " + triangle);

    // Создание другого треугольника для сравнения
    Task1 anotherTriangle = new Task1(sideA, sideB, sideC);
```

```
        // Сравнение двух треугольников
        boolean isEqual = triangle.equals(anotherTriangle);
        System.out.println("Треугольники равны? - " + isEqual);
    }
}
```

Вариант 3

Задание 2.

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

Данные для программы загружаются из файла (формат произволен). Файл создать и написать вручную.

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.

Результат программы:

```
Меню:
1. Просмотреть все автобусы
2. Просмотреть автобусы выехавшие из парка
3. Просмотреть автобусы оставшиеся в парке
4. Просмотреть автобусы по номеру маршрута
5. Просмотреть автобусы эксплуатируемые более 10 лет
6. Просмотреть автобусы с пробегом более 100000 км
7. Выйти
Выберите пункт меню: 1
```

```
Все автобусы:
Bus:ФИО водителя='Иванов И.И.', Номер автобуса='A123BC', Номер маршрута='101', Марка='Mercedes', Год начала эксплуатации=2015, Пробег=80000, Местонахождение='маршрут'
Bus:ФИО водителя='Петров П.П.', Номер автобуса='B456DE', Номер маршрута='102', Марка='Volvo', Год начала эксплуатации=2010, Пробег=120000, Местонахождение='парк'
Bus:ФИО водителя='Сидоров С.С.', Номер автобуса='C789FG', Номер маршрута='101', Марка='MAN', Год начала эксплуатации=2012, Пробег=95000, Местонахождение='маршрут'
Bus:ФИО водителя='Козлов К.К.', Номер автобуса='D012HI', Номер маршрута='103', Марка='Scania', Год начала эксплуатации=2013, Пробег=105000, Местонахождение='парк'
```

```
Выберите пункт меню: 2

Автобусы, выехавшие из парка:
Bus:ФИО водителя='Иванов И.И.', Номер автобуса='A123BC', Номер маршрута='101', Марка='Mercedes', Год начала эксплуатации=2015, Пробег=80000, Местонахождение='маршрут'
Bus:ФИО водителя='Сидоров С.С.', Номер автобуса='C789FG', Номер маршрута='101', Марка='MAN', Год начала эксплуатации=2012, Пробег=95000, Местонахождение='маршрут'
```

```
Выберите пункт меню: 3

Автобусы, оставшиеся в парке:
Bus:ФИО водителя='Петров П.П.', Номер автобуса='B456DE', Номер маршрута='102', Марка='Volvo', Год начала эксплуатации=2010, Пробег=120000, Местонахождение='парк'
Bus:ФИО водителя='Козлов К.К.', Номер автобуса='D012HI', Номер маршрута='103', Марка='Scania', Год начала эксплуатации=2013, Пробег=105000, Местонахождение='парк'
```

```
Выберите пункт меню: 4
Введите номер маршрута: 101

Автобусы на маршруте 101:
Bus:ФИО водителя='Иванов И.И.', Номер автобуса='A123BC', Номер маршрута='101', Марка='Mercedes', Год начала эксплуатации=2015, Пробег=80000, Местонахождение='маршрут'
Bus:ФИО водителя='Сидоров С.С.', Номер автобуса='C789FG', Номер маршрута='101', Марка='MAN', Год начала эксплуатации=2012, Пробег=95000, Местонахождение='маршрут'
```

```
Выберите пункт меню: 5

Автобусы, эксплуатируемые более 10 лет:
Bus:ФИО водителя='Петров П.П.', Номер автобуса='B456DE', Номер маршрута='102', Марка='Volvo', Год начала эксплуатации=2010, Пробег=120000, Местонахождение='парк'
Bus:ФИО водителя='Сидоров С.С.', Номер автобуса='C789FG', Номер маршрута='101', Марка='MAN', Год начала эксплуатации=2012, Пробег=95000, Местонахождение='маршрут'
Bus:ФИО водителя='Козлов К.К.', Номер автобуса='D012HI', Номер маршрута='103', Марка='Scania', Год начала эксплуатации=2013, Пробег=105000, Местонахождение='парк'
```

```
Выберите пункт меню: 6

Автобусы с пробегом более 100000 км:
Bus:ФИО водителя='Петров П.П.', Номер автобуса='B456DE', Номер маршрута='102', Марка='Volvo', Год начала эксплуатации=2010, Пробег=120000, Местонахождение='парк'
Bus:ФИО водителя='Козлов К.К.', Номер автобуса='D012HI', Номер маршрута='103', Марка='Scania', Год начала эксплуатации=2013, Пробег=105000, Местонахождение='парк'
```

```
Выберите пункт меню: 7
Выход из программы.


Process finished with exit code 0
```

Код программы:

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

public class Task2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BusPark busPark = new BusPark();
        busPark.loadBusesFromFile("BusInfo.txt");

        while (true) {
            // Выводим меню
            System.out.println("\nМеню:");
            System.out.println("1. Просмотреть все автобусы");
```

```
System.out.println("2. Просмотреть автобусы выехавшие из парка");
System.out.println("3. Просмотреть автобусы оставшиеся в парке");
System.out.println("4. Просмотреть автобусы по номеру маршрута");
System.out.println("5. Просмотреть автобусы эксплуатируемые более 10 лет");
System.out.println("6. Просмотреть автобусы с пробегом более 100000 км");
System.out.println("7. Выйти");

// Получаем выбор пользователя
System.out.print("Выберите пункт меню: ");
int choice = scanner.nextInt();
scanner.nextLine(); // считываем лишний перевод строк

switch (choice) {
   case 1:
      // Выводим все автобусы
      System.out.println("\nВсе автобусы:");
      busPark.printBuses(busPark.getAllBuses());
      break;
   case 2:
      // Выводим автобусы, выехавшие из парка
      System.out.println("\nАвтобусы, выехавшие из парка:");
      busPark.printBuses(busPark.getBusesLeftPark());
      break;
   case 3:
      // Выводим автобусы, оставшиеся в парке
      System.out.println("\nАвтобусы, оставшиеся в парке:");
      busPark.printBuses(busPark.getBusesInPark());
      break;
   case 4:
      // Запрашиваем номер маршрута у пользователя и выводим эти автобусы
      System.out.print("Введите номер маршрута: ");
      String routeNumber = scanner.nextLine();
      System.out.println("\nАвтобусы на маршруте " + routeNumber + ":");
      busPark.printBuses(busPark.getBusesByRouteNumber(routeNumber));
      break;
   case 5:
      // Выводим автобусы, эксплуатируемые более 10 лет
      System.out.println("\nАвтобусы, эксплуатируемые более 10 лет:");
      busPark.printBuses(busPark.getBusesOverTenYears());
      break;
   case 6:
      // Выводим автобусы с пробегом более 100000 км
      System.out.println("\nАвтобусы с пробегом более 100000 км:");
```

```
                busPark.printBuses(busPark.getBusesWithMileageOver100000());
                break;
            case 7:
                // Завершаем программу
                System.out.println("Выход из программы.");
                return;
            default:
                // Сообщаем о неверном выборе
                System.out.println("Неверный выбор. Попробуйте снова.");
        }
      }
   }
}

class BusPark {
   private List<Bus> buses;

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

   // Метод для загрузки данных об автобусах из файла
   public void loadBusesFromFile(String filename) {
      try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
         String line;
         while ((line = reader.readLine()) != null) {
            String[] parts = line.split(";");
            if (parts.length == 7) {
               String driverName = parts[0];
               String busNumber = parts[1];
               String routeNumber = parts[2];
               String brand = parts[3];
               int startYear = Integer.parseInt(parts[4]);
               int mileage = Integer.parseInt(parts[5]);
               String location = parts[6];
               Bus bus = new Bus(driverName, busNumber, routeNumber, brand, startYear,
mileage, location);
               buses.add(bus);
            } else {
               // Выводим сообщение об ошибке, если формат строки неверный
               System.err.println("Неверный формат строки: " + line);
            }
         }
```

```
    } catch (IOException e) {
        // Выводим сообщение об ошибке, если произошла ошибка чтения файла
        System.err.println("Ошибка чтения файла: " + e.getMessage());
    }
}

// Метод для получения всех автобусов
public List<Bus> getAllBuses() {
    return new ArrayList<>(buses);
}

// Метод для получения списка автобусов, выехавших из парка
public List<Bus> getBusesLeftPark() {
    List<Bus> leftPark = new ArrayList<>();
    for (Bus bus : buses) {
        if (bus.getLocation().equalsIgnoreCase("маршрут")) {
            leftPark.add(bus);
        }
    }
    return leftPark;
}

// Метод для получения списка автобусов, оставшихся в парке
public List<Bus> getBusesInPark() {
    List<Bus> inPark = new ArrayList<>();
    for (Bus bus : buses) {
        if (bus.getLocation().equalsIgnoreCase("парк")) {
            inPark.add(bus);
        }
    }
    return inPark;
}

// Метод для получения списка автобусов по номеру маршрута
public List<Bus> getBusesByRouteNumber(String routeNumber) {
    List<Bus> byRouteNumber = new ArrayList<>();
    for (Bus bus : buses) {
        if (bus.getRouteNumber().equalsIgnoreCase(routeNumber)) {
            byRouteNumber.add(bus);
        }
    }
    return byRouteNumber;
}
```

```
    // Метод для получения списка автобусов, эксплуатируемых более 10 лет
    public List<Bus> getBusesOverTenYears() {
        List<Bus> overTenYears = new ArrayList<>();
        int currentYear = 2024; // Предположим, что текущий год 2024
        for (Bus bus : buses) {
            if (currentYear - bus.getStartYear() > 10) {
                overTenYears.add(bus);
            }
        }
        return overTenYears;
    }

    // Метод для получения списка автобусов с пробегом более 100000 км
    public List<Bus> getBusesWithMileageOver100000() {
        List<Bus> over100000 = new ArrayList<>();
        for (Bus bus : buses) {
            if (bus.getMileage() > 100000) {
                over100000.add(bus);
            }
        }
        return over100000;
    }

    // Метод для вывода списка автобусов
    public void printBuses(List<Bus> buses) {
        for (Bus bus : buses) {
            System.out.println(bus);
        }
    }
}

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

    // Конструктор класса
    public Bus(String driverName, String busNumber, String routeNumber, String brand, int
```

```
startYear, int mileage, String location) {
        this.driverName = driverName;
        this.busNumber = busNumber;
        this.routeNumber = routeNumber;
        this.brand = brand;
        this.startYear = startYear;
        this.mileage = mileage;
        this.location = location;
    }

    // Геттеры и сеттеры для полей класса
    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 getRouteNumber() {
        return routeNumber;
    }

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

    public String getBrand() {
        return brand;
    }

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

```
    public int getStartYear() {
        return startYear;
    }

    public void setStartYear(int startYear) {
        this.startYear = startYear;
    }

    public int getMileage() {
        return mileage;
    }

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

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    // Переопределение метода toString для класса Bus
    @Override
    public String toString() {
        return "Bus:" +
                "ФИО водителя='" + driverName + '\'' +
                ", Номер автобуса='" + busNumber + '\'' +
                ", Номер маршрута='" + routeNumber + '\'' +
                ", Марка='" + brand + '\'' +
                ", Год начала эксплуатации=" + startYear +
                ", Пробег=" + mileage +
                ", Местонахождение='" + location + '\'';
    }
}
```

Вывод: в результате выполнения лабораторной были приобретены практические навыки создания и использования классов в программах на языке Java.