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

Лабораторная работа №3

По дисциплине: «Современные платформы программирования»

Выполнил:
студент 3 курса
группы ПО-8
Сорока В.С.

Проверил:
Крощенко А.А.

Брест, 2024

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

Задание 1 (Вариант 2)
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.

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()

Равносторонний 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 инициализацией.
Реализовать метод equals, выполняющий сравнение объектов данного типа.

Спецификации ввода-вывода программы:
- Создание класса EquilateralTriangle для представления равностороннего треугольника.
- Класс EquilateralTriangle имеет приватное поле sideLength для хранения длины стороны треугольника.
- Конструктор класса EquilateralTriangle принимает аргумент sideLength и инициализирует поле sideLength переданным значением.
- Геттер getSideLength позволяет получить значение длины стороны треугольника.
- Сеттер setSideLength позволяет изменить значение длины стороны треугольника.
- Метод calculateArea вычисляет площадь равностороннего треугольника с помощью формулы (корень из 3 / 4) * sideLength * sideLength и возвращает полученное значение.
- Метод calculatePerimeter вычисляет периметр равностороннего треугольника, умножая длину стороны на 3, и возвращает полученное значение.

- Метод exists возвращает логическое значение true, если длина стороны треугольника больше 0 (треугольник существует), и false в противном случае.
- Переопределение метода toString предоставляет строковое представление объекта EquilateralTriangle, включающее значение длины стороны.
- Переопределение метода equals позволяет сравнивать два объекта типа EquilateralTriangle на равенство по значению длины стороны.
- В блоке кода case 1 создается объект triangle1 равностороннего треугольника с длиной стороны 7.
- Вызов методов calculateArea и calculatePerimeter позволяет вычислить площадь и периметр треугольника соответственно.
- Метод exists проверяет существование треугольника.
- Выводится информация о площади, периметре и существовании треугольника.
- Создается второй объект triangle2 равностороннего треугольника с длиной стороны 5.
- Метод equals сравнивает два треугольника на равенство длины стороны.
- Результат сравнения выводится на экран.

Текст программы

```
class EquilateralTriangle
{
    // длина стороны
    private double sideLength;

    // Конструктор с начальной инициализацией
    public EquilateralTriangle(double sideLength) {
        this.sideLength = sideLength;
    }

    // Геттер для получения длины стороны
    public double getSideLength() {
        return sideLength;
    }
    // Сеттер для изменения длины стороны
    public void setSideLength(double sideLength) {
        this.sideLength = sideLength;
    }

    // Метод для определения площади треугольника
    public double calculateArea() {
        return (Math.sqrt(3) / 4) * sideLength * sideLength;
    }

    // Метод для определения периметра треугольника
    public double calculatePerimeter() {
        return 3 * sideLength;
    }

    // Логический метод для определения существования треугольника
    public boolean exists() {
        return sideLength > 0;
    }

    // Переопределение метода toString()
```

```java
    @Override
    public String toString() {
        return "EquilateralTriangle{" +
                "sideLength=" + sideLength +
                '}';
    }

    // Переопределение метода equals()
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        EquilateralTriangle that = (EquilateralTriangle) o;
        return Double.compare(that.sideLength, sideLength) == 0;
    }
}
```

```java
case 1:
    // Создание объекта равностороннего треугольника с длиной стороны 7
    EquilateralTriangle triangle1 = new EquilateralTriangle(7);

    // Вызов методов для расчета площади и периметра треугольника
    double area = triangle1.calculateArea();
    double perimeter = triangle1.calculatePerimeter();

    // Проверка существования треугольника
    boolean exists = triangle1.exists();

    // Вывод результатов
    System.out.println("Площадь треугольника: " + area);
    System.out.println("Периметр треугольника: " + perimeter);
    System.out.println("Треугольник существует: " + exists);

    // Создание второго равностороннего треугольника с длиной стороны 5
    EquilateralTriangle triangle2 = new EquilateralTriangle(5);

    // Проверка равенства двух объектов
    boolean isEqual = triangle1.equals(triangle2);
    System.out.println("Два треугольника равны: " + isEqual);
    break;
```

Пример работы программы

```
1. Задание 1
2. Задание 2
3. Выход

Выберите необходимое:  1

========================================================================
Площадь треугольника: 21.217622392718745
Периметр треугольника: 21.0
Треугольник существует: true
Два треугольника равны: false
========================================================================
```

Задание 2 (Вариант 2)
Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса.
Реализовать требуемые функции обработки данных.

Требования к выполнению
• Задание посвящено написанию классов, решающих определенную задачу автоматизации;
• Данные для программы загружаются из файла (формат произволен). Файл создать и написать вручную.

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.

Спецификации ввода-вывода программы:

класс Car:
- Класс Car имеет приватные поля, которые хранят информацию о различных характеристиках автомобиля, такие как идентификатор (id), бренд, модель, год выпуска, цвет кузова, цена, регистрационный номер, идентификационный номер (VIN), ФИО лица-арендатора и номер паспорта лица-арендатора.
- У всех полей определены публичные геттеры, которые возвращают значения соответствующих полей.

- Конструктор класса Car принимает аргументы для всех полей и инициализирует их значениями заданными аргументами.
- Класс Car переопределяет метод toString(), который возвращает строковое представление объекта Car, содержащее информацию обо всех его полях. Если поля renterName и renterPassportNumber не равны null, то возвращаемая строка также включает информацию о ФИО лица-арендатора и номере паспорта лица-арендатора.
- Вывод объекта Car осуществляется с помощью метода toString().

класс CarRentalSystem:
- Класс CarRentalSystem содержит приватное поле cars, которое представляет собой список объектов класса Car.
- Конструктор класса CarRentalSystem инициализирует список cars пустым ArrayList.
- Метод addCar добавляет переданный объект car в список cars.
- Метод getAllCars возвращает весь список cars.
- Метод getCarsByBrand принимает строковый аргумент brand и возвращает список автомобилей, у которых значение поля brand соответствует переданному аргументу.
- Метод getCarsByModelAndYearsUsedMoreThan принимает целочисленный аргумент years и возвращает список автомобилей, у которых значение поля model соответствует переданному аргументу и количество лет, прошедших с года выпуска автомобиля, больше переданного аргумента years.
- Метод getCarsByYearAndPriceGreaterThan принимает целочисленные аргументы year и price и возвращает список автомобилей, у которых значение поля year равно переданному аргументу year, а значение поля price больше переданного аргумента price.
- Метод getRentedCars возвращает список арендованных автомобилей, у которых поля renterName и renterPassportNumber не равны null. Возвращаемый список содержит объекты типа Car, у которых значения полей renterName и renterPassportNumber заменены на null.
- Метод getRentedCarsWithRenterInfo возвращает список арендованных автомобилей, у которых поля renterName и renterPassportNumber не равны null. Возвращаемый список содержит оригинальные объекты типа Car, не изменяя значения полей renterName и renterPassportNumber.

Вывод
- Код начинается с создания объекта CarRentalSystem.
- Далее происходит чтение данных из файла "cars.json" с помощью FileReader.
- С использованием библиотеки Gson, данные в формате JSON преобразуются в список объектов Car.
- Каждый объект Car из списка добавляется в CarRentalSystem с помощью метода addCar.
- После добавления всех автомобилей, происходит вывод информации:

- Вывод всех автомобилей из CarRentalSystem с помощью метода getAllCars.
- Вывод автомобилей заданной марки из CarRentalSystem с помощью метода getCarsByBrand. Марки автомобилей заданы в списке brands.
- Вывод автомобилей заданной модели, которые эксплуатируются больше n лет с помощью метода getCarsByModelAndYearsUsedMoreThan.
- Вывод автомобилей заданного года выпуска, цена которых больше указанной, с помощью метода getCarsByYearAndPriceGreaterThan.
- Вывод автомобилей, взятых на прокат, из CarRentalSystem с помощью метода getRentedCars.
- Все выводы выполняются с использованием цикла for и метода println.

Текст программы

```
class Car {
    private int id;
    public int getId() {
        return id;
    }
    private String brand;
    public String getBrand() {
        return brand;
    }
    private String model;
    public String getModel() {
        return model;
    }
    private int year;
    public int getYear() {
        return year;
    }
    private String color;
    public String getColor() {
        return color;
    }
    private double price;
    public double getPrice() {
        return price;
    }
    private String registrationNumber;
    public String getRegistrationNumber() {
        return registrationNumber;
    }
    private String licensePlateNumber;
    public String getLicensePlateNumber() {
        return licensePlateNumber;
    }
    private String renterName;
    public String getRenterName() {
        return renterName;
    }
    private String renterPassportNumber;
    public String getRenterPassportNumber() {
        return renterPassportNumber;
    }

    public Car(int id, String brand, String model, int year, String color,
double price, String registrationNumber, String licensePlateNumber, String
renterName, String renterPassportNumber) {
```

```
        this.id = id;
        this.brand = brand;
        this.model = model;
        this.year = year;
        this.color = color;
        this.price = price;
        this.registrationNumber = registrationNumber;
        this.licensePlateNumber = licensePlateNumber;
        this.renterName = renterName;
        this.renterPassportNumber = renterPassportNumber;
    }

    // Геттеры и сеттеры для всех полей
    @Override
    public String toString() {
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("\nID: " + id +
                "\nБренд: " + brand +
                "\nМодель: " + model +
                "\nГод выпуска: " + year +
                "\nЦвет кузова: " + color +
                "\nЦена при покупке: " + price + " $" +
                "\nРегистрационный номер: " + registrationNumber +
                "\nИдентификационный номер (VIN): " + licensePlateNumber);

        if (renterName != null && renterPassportNumber != null) {
            stringBuilder.append("\nФИО лица-арендатора: " + renterName +
                    "\nНомер паспорта лица-арендатора: " +
renterPassportNumber);
        }
        return stringBuilder.toString();
    }
}

class CarRentalSystem {
    private List<Car> cars;
    public CarRentalSystem() {
        this.cars = new ArrayList<>();
    }
    public void addCar(Car car) {
        cars.add(car);
    }
    public List<Car> getAllCars() {
        return cars;
    }
    public List<Car> getCarsByBrand(String brand) {
        List<Car> carsByBrand = new ArrayList<>();
        for (Car car : cars) {
            if (car.getBrand().equals(brand)) {
                carsByBrand.add(car);
            }
        }
        return carsByBrand;
    }
    public List<Car> getCarsByModelAndYearsUsedMoreThan( int years) {
        List<Car> carsByModelAndYearsUsedMoreThan = new ArrayList<>();
        for (Car car : cars) {
            if (2024 - car.getYear() > years) {
                carsByModelAndYearsUsedMoreThan.add(car);
            }
        }
        return carsByModelAndYearsUsedMoreThan;
    }
    public List<Car> getCarsByYearAndPriceGreaterThan(int year, double price)
```

```
{
        List<Car> carsByYearAndPriceGreaterThan = new ArrayList<>();
        for (Car car : cars) {
            if (car.getYear() == year && car.getPrice() > price) {
                carsByYearAndPriceGreaterThan.add(car);
            }
        }
        return carsByYearAndPriceGreaterThan;
    }
    public List<Car> getRentedCars() {
        List<Car> rentedCars = new ArrayList<>();
        for (Car car : cars) {
            if (car.getRenterName() != null && car.getRenterPassportNumber()
!= null) {
                Car rentedCar = new Car(
                        car.getId(),
                        car.getBrand(),
                        car.getModel(),
                        car.getYear(),
                        car.getColor(),
                        car.getPrice(),
                        car.getLicensePlateNumber(),
                        car.getRegistrationNumber(),
                        null,
                        null
                );
                rentedCars.add(rentedCar);
            }
        }
        return rentedCars;
    }
    public List<Car> getRentedCarsWithRenterInfo() {
        List<Car> rentedCarsWithRenterInfo = new ArrayList<>();
        for (Car car : cars) {
            if (car.getRenterName() != null && car.getRenterPassportNumber()
!= null) {
                rentedCarsWithRenterInfo.add(car);
            }
        }
        return rentedCarsWithRenterInfo;
    }
}
```

```
case 2:
    CarRentalSystem carRentalSystem = new CarRentalSystem();

    try (FileReader reader = new FileReader("cars.json")) {
        Gson gson = new Gson();
        Type carListType = new TypeToken<List<Car>>() {}.getType();
        List<Car> cars = gson.fromJson(reader, carListType);

        for (Car car : cars) {
            carRentalSystem.addCar(car);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
    // Вывод всех автомобилей
    System.out.println("Все автомобили: ");
    List<Car> allCars = carRentalSystem.getAllCars();
    for (Car car : allCars) {
        System.out.println(car);
    }

    // Вывод автомобилей заданной марки
    List<String> brands = new ArrayList<>();
    brands.add("Ford");
    brands.add("Peugeot");
    String brandString = String.join(", ", brands);
    System.out.println("Автомобили заданных марок: " + brandString);
    for (String brand : brands) {
        List<Car> carsByBrand = carRentalSystem.getCarsByBrand(brand);
        for (Car car : carsByBrand) {
            System.out.println(car);
        }
    }

    // Вывод автомобилей заданной модели, котор. эксплуатируются больше n лет
    System.out.println("Автомобили, которые эксплуатируются больше 5 лет: ");
    List<Car> carsByModelAndYearsUsedMoreThan =
carRentalSystem.getCarsByModelAndYearsUsedMoreThan( 5);
    for (Car car : carsByModelAndYearsUsedMoreThan) {
        System.out.println(car);
    }

    // Вывод автомобилей заданного года выпуска, цена которых больше
указанной
    int year = 2004;
    double price = 5800;
    System.out.println("Автомобили заданного года выпуска, цена которых
больше указанной: \nгод - " + year + ", цена - от " + price + "$");
    List<Car> carsByYearAndPriceGreaterThan =
carRentalSystem.getCarsByYearAndPriceGreaterThan(year, price);
    for (Car car : carsByYearAndPriceGreaterThan) {
        System.out.println(car);
    }

    // Вывод автомобилей, взятых на прокат
    System.out.println("Автомобили, взятые на прокат: ");
    List<Car> rentedCars = carRentalSystem.getRentedCars();
    for (Car car : rentedCars) {
        System.out.println(car);
    }

    // Вывод автомобилей, взятых на прокат с выводом личной информации об
арендаторах
    System.out.println("Автомобили, взятые на прокат с личной информацией об
арендаторах: ");
    List<Car> rentedCarsWithRenterInfo =
carRentalSystem.getRentedCarsWithRenterInfo();
    for (Car car : rentedCarsWithRenterInfo) {
        System.out.println(car);
    }
    break;
```

Пример работы программы:

```
----------------------------------------------------------------------
Автомобили заданного года выпуска, цена которых больше указанной:
год - 2004, цена - от 5800.0$

ID: 2
Бренд: Peugeot
Модель: 307
Год выпуска: 2004
Цвет кузова: Серебристый металлик
Цена при покупке: 5900.0 $
Регистрационный номер: null
Идентификационный номер (VIN): 2222 BC-7
ФИО лица-арендатора: Сорока Вадим Сергеевич
Номер паспорта лица-арендатора: 389566PB410C1
----------------------------------------------------------------------
Автомобили, взятые на прокат:

ID: 1
Бренд: Volkswagen
Модель: Passat B3
Год выпуска: 1990
Цвет кузова: Чёрный металлик
Цена при покупке: 3300.0 $
Регистрационный номер: 2545 MH-1
Идентификационный номер (VIN): null

ID: 2
Бренд: Peugeot
Модель: 307
Год выпуска: 2004
Цвет кузова: Серебристый металлик
Цена при покупке: 5900.0 $
Регистрационный номер: 2222 BC-7
Идентификационный номер (VIN): null
----------------------------------------------------------------------
```

Вывод: научился создавать и использовать классы языка программирования Java при решении практических задач.