МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»

Факультет управления и информатики в технологических системах
Кафедра Информационной безопасности
Направление подготовки
(специальность) 10.05.03 Информационная безопасность автоматизированных систем

Отчет

по практике по технологиям и методам программирования
наименование (вид) практики

Выполнил студент гр. УБ-42
Крылов Никита Романович
(Ф.И.О.)

_____
(подпись)

Проверили:

Маслов А.А.
(Ф.И.О.)

_____ _____
(оценка) *(подпись)*

_____
(дата)

Воронеж - 2025

Задание 15.

Продажа автомобилей. Реализовать сортировку по марке автомо-билей и по цене.

Код программы:

```
package Pr5

import java.time.LocalDate;
import java.util.*;

class Car {
private String brand;
private int year;
private double price;
private String комплектация;
private String country;
private LocalDate saleDate;
private String customerFullName;

public Car(String brand, int year, double price, String комплектация, String country, LocalDate
saleDate, String customerFullName) {
this.brand = brand;
this.year = year;
this.price = price;
this.комплектация = комплектация;
this.country = country;
this.saleDate = saleDate;
this.customerFullName = customerFullName;
}
public String getBrand() {
return brand;
}

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

public int getYear() {
return year;
}

public void setYear(int year) {
this.year = year;
}

public double getPrice() {
return price;
}

public void setPrice(double price) {
this.price = price;
}
```

```
public String getКомплектация() {
return комплектация;
}

public void setКомплектация(String комплектация) {
this.комплектация = комплектация;
}

public String getCountry() {
return country;
}

public void setCountry(String country) {
this.country = country;
}

public LocalDate getSaleDate() {
return saleDate;
}

public void setSaleDate(LocalDate saleDate) {
this.saleDate = saleDate;
}

public String getCustomerFullName() {
return customerFullName;
}

public void setCustomerFullName(String customerFullName) {
this.customerFullName = customerFullName;
}
@Override
public String toString() {
return "Car{" +
"brand='" + brand + '\'' +
", year=" + year +
", price=" + price +
", комплектация='" + комплектация + '\'' +
", country='" + country + '\'' +
", saleDate=" + saleDate +
", customerFullName='" + customerFullName + '\'' +
'}';
}
@Override
public boolean equals(Object o) {
if (this == o) return true;
if (o == null || getClass() != o.getClass()) return false;
Car car = (Car) o;
return year == car.year &&
Double.compare(car.price, price) == 0 &&
Objects.equals(brand, car.brand) &&
Objects.equals(комплектация, car.комплектация) &&
Objects.equals(country, car.country) &&
Objects.equals(saleDate, car.saleDate) &&
Objects.equals(customerFullName, car.customerFullName);
}
```

```
@Override
public int hashCode() {
return Objects.hash(brand, year, price, комплектация, country, saleDate, customerFullName);
}
}
class UsedCar extends Car {
private String condition;
private String previousOwnerFullName;
private int mileage;

public UsedCar(String brand, int year, double price, String комплектация, String country, LocalDate
saleDate, String customerFullName, String condition, String previousOwnerFullName, int mileage) {
super(brand, year, price, комплектация, country, saleDate, customerFullName);
this.condition = condition;
this.previousOwnerFullName = previousOwnerFullName;
this.mileage = mileage;
}
public String getCondition() {
return condition;
}

public void setCondition(String condition) {
this.condition = condition;
}

public String getPreviousOwnerFullName() {
return previousOwnerFullName;
}

public void setPreviousOwnerFullName(String previousOwnerFullName) {
this.previousOwnerFullName = previousOwnerFullName;
}

public int getMileage() {
return mileage;
}

public void setMileage(int mileage) {
this.mileage = mileage;
}
@Override
public String toString() {
return "UsedCar{" +
"condition='" + condition + '\'' +
", previousOwnerFullName='" + previousOwnerFullName + '\'' +
", mileage=" + mileage +
"} " + super.toString();
}
@Override
public boolean equals(Object o) {
if (this == o) return true;
if (o == null || getClass() != o.getClass()) return false;
if (!super.equals(o)) return false;
UsedCar usedCar = (UsedCar) o;
return mileage == usedCar.mileage && Objects.equals(condition, usedCar.condition) &&
Objects.equals(previousOwnerFullName, usedCar.previousOwnerFullName);
```

```
}

@Override
public int hashCode() {
return Objects.hash(super.hashCode(), condition, previousOwnerFullName, mileage);
}
}

class SportsCar extends Car {
private double accelerationTime;
private double engineCapacity;
private int horsepower;

public SportsCar(String brand, int year, double price, String комплектация, String country, LocalDate
saleDate, String customerFullName, double accelerationTime, double engineCapacity, int horsepower) {
super(brand, year, price, комплектация, country, saleDate, customerFullName);
this.accelerationTime = accelerationTime;
this.engineCapacity = engineCapacity;
this.horsepower = horsepower;
}

public double getAccelerationTime() {
return accelerationTime;
}

public void setAccelerationTime(double accelerationTime) {
this.accelerationTime = accelerationTime;
}

public double getEngineCapacity() {
return engineCapacity;
}

public void setEngineCapacity(double engineCapacity) {
this.engineCapacity = engineCapacity;
}

public int getHorsepower() {
return horsepower;
}

public void setHorsepower(int horsepower) {
this.horsepower = horsepower;
}

@Override
public String toString() {
return "SportsCar{" +
"accelerationTime=" + accelerationTime +
", engineCapacity=" + engineCapacity +
", horsepower=" + horsepower +
"} " + super.toString();
}
@Override
public boolean equals(Object o) {
if (this == o) return true;
if (o == null || getClass() != o.getClass()) return false;
```

```
if (!super.equals(o)) return false;
SportsCar sportsCar = (SportsCar) o;
return Double.compare(sportsCar.accelerationTime, accelerationTime) == 0 &&
Double.compare(sportsCar.engineCapacity, engineCapacity) == 0 && horsepower ==
sportsCar.horsepower;
}

@Override
public int hashCode() {
return Objects.hash(super.hashCode(), accelerationTime, engineCapacity, horsepower);
}
}

class SpecialEquipment extends Car {
private String type;
private double weight;
private String dimensions;

public SpecialEquipment(String brand, int year, double price, String комплектация, String country,
LocalDate saleDate, String customerFullName, String type, double weight, String dimensions) {
super(brand, year, price, комплектация, country, saleDate, customerFullName);
this.type = type;
this.weight = weight;
this.dimensions = dimensions;
}

public String getType() {
return type;
}

public void setType(String type) {
this.type = type;
}

public double getWeight() {
return weight;
}

public void setWeight(double weight) {
this.weight = weight;
}

public String getDimensions() {
return dimensions;
}

public void setDimensions(String dimensions) {
this.dimensions = dimensions;
}

@Override
public String toString() {
return "SpecialEquipment{" +
"type='" + type + '\'' +
", weight=" + weight +
", dimensions='" + dimensions + '\'' +
"} " + super.toString();
```

```
}

@Override
public boolean equals(Object o) {
if (this == o) return true;
if (o == null || getClass() != o.getClass()) return false;
if (!super.equals(o)) return false;
SpecialEquipment that = (SpecialEquipment) o;
return Double.compare(that.weight, weight) == 0 && Objects.equals(type, that.type) &&
Objects.equals(dimensions, that.dimensions);
}

@Override
public int hashCode() {
return Objects.hash(super.hashCode(), type, weight, dimensions);
}
}

class CarSalesList {
private List<Car> cars;

public CarSalesList() {
this.cars = new ArrayList<>();
}

public void addCar(Car car) {
this.cars.add(car);
}

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

public void sortByBrand() {
Collections.sort(this.cars, Comparator.comparing(Car::getBrand));
}
public void sortByPrice() {
Collections.sort(this.cars, Comparator.comparing(Car::getPrice));
}
}

public class CarSaleExample {
public static void main(String[] args) {
CarSalesList salesList = new CarSalesList();

LocalDate saleDate1 = LocalDate.of(2023, 11, 15);
Car car1 = new Car("Lada", 2020, 10000.0, "Base", "Russia", saleDate1, "Петров Петр Петрович");
salesList.addCar(car1);

LocalDate saleDate2 = LocalDate.of(2023, 11, 20);
UsedCar usedCar1 = new UsedCar("Toyota", 2015, 15000.0, "Comfort", "Japan", saleDate2, "Сидоров
Иван Иванович", "Excellent", "Иванов Иван", 100000);
salesList.addCar(usedCar1);

LocalDate saleDate3 = LocalDate.of(2023, 11, 25);
```

```
SportsCar sportsCar1 = new SportsCar("Ferrari", 2022, 300000.0, "Full", "Italy", saleDate3, "Смирнов Алексей", 3.5, 4.0, 600);
salesList.addCar(sportsCar1);

LocalDate saleDate4 = LocalDate.of(2023, 12, 01);
SpecialEquipment specialEquipment1 = new SpecialEquipment("Caterpillar", 2018, 250000.0, "N/A", "USA", saleDate4, "Кузнецов Сергей", "Excavator", 20000.0, "8.5x3x3.5 m");
salesList.addCar(specialEquipment1);

System.out.println("Список проданных автомобилей (изначальный):");
salesList.printCarList();

System.out.println("\nСписок проданных автомобилей (сортировка по марке):");
salesList.sortByBrand();
salesList.printCarList();

System.out.println("\nСписок проданных автомобилей (сортировка по цене):");
salesList.sortByPrice();
salesList.printCarList();
}
}
```

Результат программы.

```
> /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/n1kzet/.config/Code/User/workspaceStorage/fc612d967b764b6b5b43e845124743df/redhat.java/jdt_ws/Programming-technologies-and-methods_4672a479/bin Pr5.CarSaleExample
Список проданных автомобилей (изначальный):
Car{brand='Lada', year=2020, price=10000.0, комплектация='Base', country='Russia', saleDate=2023-11-15, customerFullName='Петров Петр Петрович'}
UsedCar{condition='Excellent', previousOwnerFullName='Иванов Иван', mileage=100000} Car{brand='Toyota', year=2015, price=15000.0, комплектация='Comfort', country='Japan', saleDate=2023-11-20, customerFullName='Сидоров Иван Иванович'}
SportsCar{accelerationTime=3.5, engineCapacity=4.0, horsepower=600} Car{brand='Ferrari', year=2022, price=300000.0, комплектация='Full', country='Italy', saleDate=2023-11-25, customerFullName='Смирнов Алексей'}
SpecialEquipment{type='Excavator', weight=20000.0, dimensions='8.5x3x3.5 m'} Car{brand='Caterpillar', year=2018, price=250000.0, комплектация='N/A', country='USA', saleDate=2023-12-01, customerFullName='Кузнецов Сергей'}

Список проданных автомобилей (сортировка по марке):
SpecialEquipment{type='Excavator', weight=20000.0, dimensions='8.5x3x3.5 m'} Car{brand='Caterpillar', year=2018, price=250000.0, комплектация='N/A', country='USA', saleDate=2023-12-01, customerFullName='Кузнецов Сергей'}
SportsCar{accelerationTime=3.5, engineCapacity=4.0, horsepower=600} Car{brand='Ferrari', year=2022, price=300000.0, комплектация='Full', country='Italy', saleDate=2023-11-25, customerFullName='Смирнов Алексей'}
Car{brand='Lada', year=2020, price=10000.0, комплектация='Base', country='Russia', saleDate=2023-11-15, customerFullName='Петров Петр Петрович'}
UsedCar{condition='Excellent', previousOwnerFullName='Иванов Иван', mileage=100000} Car{brand='Toyota', year=2015, price=15000.0, комплектация='Comfort', country='Japan', saleDate=2023-11-20, customerFullName='Сидоров Иван Иванович'}

Список проданных автомобилей (сортировка по цене):
Car{brand='Lada', year=2020, price=10000.0, комплектация='Base', country='Russia', saleDate=2023-11-15, customerFullName='Петров Петр Петрович'}
UsedCar{condition='Excellent', previousOwnerFullName='Иванов Иван', mileage=100000} Car{brand='Toyota', year=2015, price=15000.0, комплектация='Comfort', country='Japan', saleDate=2023-11-20, customerFullName='Сидоров Иван Иванович'}
SpecialEquipment{type='Excavator', weight=20000.0, dimensions='8.5x3x3.5 m'} Car{brand='Caterpillar', year=2018, price=250000.0, комплектация='N/A', country='USA', saleDate=2023-12-01, customerFullName='Кузнецов Сергей'}
SportsCar{accelerationTime=3.5, engineCapacity=4.0, horsepower=600} Car{brand='Ferrari', year=2022, price=300000.0, комплектация='Full', country='Italy', saleDate=2023-11-25, customerFullName='Смирнов Алексей'}
```

Вывод.

Сегодня я разобрал, как эффективно работать с коллекциями в Java, используя принципы ООП. Программа по учету продаж автомобилей показала, как коллекции позволяют управлять группами объектов, обеспечивая удобное хранение, сортировку и обработку данных.

1. Коллекции как основа хранения данных

В программе используется List<Car> – динамическая коллекция, хранящая все типы автомобилей (обычные, подержанные, спортивные и спецтехнику). Это демонстрирует ключевые преимущества коллекций:

- Гибкость – можно добавлять, удалять и модифицировать элементы.
- Универсальность– благодаря полиморфизму, коллекция List<Car> может содержать объекты любого подкласса (`UsedCar`, `SportsCar` и т. д.).
- Стандартные операции – сортировка, итерация, фильтрация (например, `sortByBrand()` и `sortByPrice()`).

2. Полиморфизм и коллекции

Коллекция `List<Car>` работает с разными типами автомобилей через общий интерфейс:

- При вызове `car.toString()` для каждого элемента выполняется переопределенная версия метода (из `UsedCar`, `SportsCar` и др.).
- Это позволяет обрабатывать разнородные объекты единообразно, не задумываясь об их конкретном типе.

Программа наглядно показала, что коллекции – это мощный инструмент, который в сочетании с ООП позволяет создавать структурированные и масштабируемые приложения.