# **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. Выполнить преобразование класса в кол-лекцию. Создать пользовательское меню. Организовать добавление объектов в кол-лекцию и вывод отсортированных объектов коллекции на экран с помощью меню.

**Код программы:**

package Pr6;

import java.time.LocalDate;

import java.util.\*;

import java.util.Scanner;

class Automobile {

protected String brand;

protected int year;

protected double price;

protected String configuration;

protected String countryOfOrigin;

protected LocalDate saleDate;

protected String buyerName;

public Automobile(String brand, int year, double price, String configuration,

String countryOfOrigin, LocalDate saleDate, String buyerName) {

this.brand = brand;

this.year = year;

this.price = price;

this.configuration = configuration;

this.countryOfOrigin = countryOfOrigin;

this.saleDate = saleDate;

this.buyerName = buyerName;

}

@Override

public String toString() {

return String.format("Марка: %s\nГод выпуска: %d\nЦена: %.2f\nКомплектация: %s\nСтрана производитель: %s\nДата продажи: %s\nПокупатель: %s",

brand, year, price, configuration, countryOfOrigin, saleDate, buyerName);

}

}

class CarSales {

private List<Automobile> soldCars = new ArrayList<>();

public void addCar(Automobile car) {

soldCars.add(car);

}

public void sortByBrand() {

soldCars.sort(Comparator.comparing(car -> car.brand));

}

public void sortByPrice() {

soldCars.sort(Comparator.comparingDouble(car -> car.price));

}

public void printSales() {

for (Automobile car : soldCars) {

System.out.println(car);

System.out.println("----------------------");

}

}

}

public class Main {

private static final Scanner scanner = new Scanner(System.in);

private static final CarSales sales = new CarSales();

public static void main(String[] args) {

while (true) {

System.out.println("Меню:");

System.out.println("1. Добавить автомобиль");

System.out.println("2. Вывести список автомобилей (сортировка по марке)");

System.out.println("3. Вывести список автомобилей (сортировка по цене)");

System.out.println("4. Выход");

System.out.print("Выберите опцию: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

addCar();

break;

case 2:

sales.sortByBrand();

sales.printSales();

break;

case 3:

sales.sortByPrice();

sales.printSales();

break;

case 4:

System.out.println("Выход из программы.");

return;

default:

System.out.println("Неверный ввод, попробуйте снова.");

}

}

}

private static void addCar() {

System.out.print("Введите марку автомобиля: ");

String brand = scanner.nextLine();

System.out.print("Введите год выпуска: ");

int year = scanner.nextInt();

System.out.print("Введите цену: ");

double price = scanner.nextDouble();

scanner.nextLine();

System.out.print("Введите комплектацию: ");

String configuration = scanner.nextLine();

System.out.print("Введите страну производитель: ");

String country = scanner.nextLine();

System.out.print("Введите дату продажи (ГГГГ-ММ-ДД): ");

LocalDate saleDate = LocalDate.parse(scanner.nextLine());

System.out.print("Введите ФИО покупателя: ");

String buyer = scanner.nextLine();

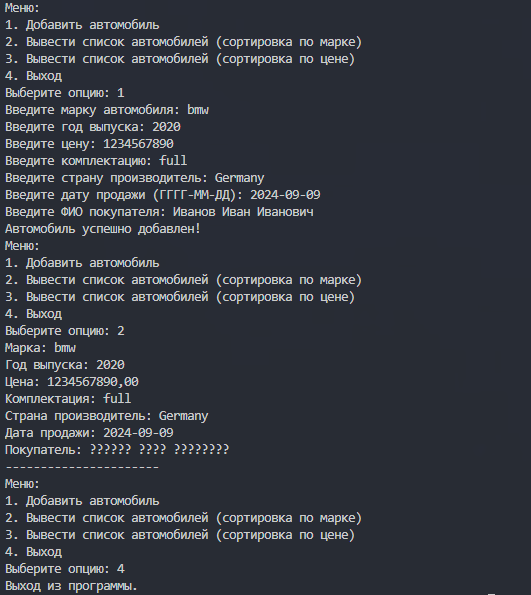
sales.addCar(new Automobile(brand, year, price, configuration, country, saleDate, buyer));

System.out.println("Автомобиль успешно добавлен!");

}

}

**Результат программы.**



**Вывод.**

**Сегодня я разобрал, как эффективно работать с вводом-выводом в Java, используя пакет java.io. Программа по учету продаж автомобилей продемонстрировала, как потоки и сериализация позволяют управлять данными, обеспечивая их сохранение, загрузку и обработку.**

1. Потоки как основа ввода-вывода

В программе используется Scanner для чтения данных с консоли и System.out для вывода. Это иллюстрирует ключевые возможности потоков:

* **Гибкость** – можно работать с разными источниками данных (консоль, файлы, сеть).
* **Универсальность** – одни и те же методы (например, readLine(), write()) применяются к различным устройствам.
* **Стандартизация** – классы java.io предоставляют готовые решения для чтения и записи (ObjectInputStream, Console и др.).

##### 2. Сериализация и работа с объектами

Программа использует принципы сериализации для сохранения состояния объектов:

* Класс Automobile можно легко преобразовать в байтовый поток и сохранить в файл.
* Интерфейс Serializable позволяет автоматически сериализовать объекты, а transient исключает ненужные поля.
* Если бы мы добавили сохранение данных в файл, это выглядело бы так: try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("cars.dat"))) {  
   oos.writeObject(soldCars); // сохранение списка автомобилей  
  }  
   Это показывает, как сериализация упрощает работу с данными.

3. Практическое применение

Программа наглядно демонстрирует:

* Как организовать интерактивный ввод через Scanner.
* Как структурировать данные в коллекции (ArrayList<Automobile>).
* Как сортировать объекты с помощью Comparator.
* Почему важно обрабатывать исключения (IOException, ClassNotFoundException).

**Вывод:**

Пакет java.io – это мощный инструмент для работы с данными, который в сочетании с ООП позволяет создавать гибкие и надежные приложения. Сериализация особенно полезна для сохранения состояния объектов, а потоки делают ввод-вывод универсальным независимо от источника данных.

Эта работа помогла мне глубже понять, как Java управляет вводом-выводом, и как применять эти знания в реальных проектах.