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.
Создать программу на языке Java для определения класса в некоторой предметной области. Описать свойства, конструктор, методы геттеры/сеттеры, перекрыть метод toString() для вывода полной информации об объекте в отформатированном виде:

Вариант 15). Продажа автомобилей

Car:

Свойства:

- марка автомобиля;
- Год выпуска;
- Цена автомобиля;
- Комплетация;
- Страна производитель;
- Дата продажи;
- ФИО покупателя;

} Конструктор

Код программы:

```
import java.util.Date;

public class Car{

private String brand;
private int year;
private double price;
private String equipment;
private String country;
private Date saleDate;
private String buyerName;

public Car(String brand, int year, double price, String equipment, String country,
Date saleDate, String buyerName) {
this.brand = brand;
this.year = year;
this.price = price;
this.equipment = equipment;
this.country = country;
this.saleDate = saleDate;
this.buyerName = buyerName;
```

```
}

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 getEquipment() {
return equipment;
}

public void setEquipment(String equipment) {
this.equipment = equipment;
}

public String getCountry() {
```

```
return country;
}

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

public Date getSaleDate() {
return saleDate;
}

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

public String getBuyerName() {
return buyerName;
}

public void setBuyerName(String buyerName) {
this.buyerName = buyerName;
}

@Override
public String toString() {
return "Car{" +
"brand='" + brand + '\'' +
", year=" + year +
", price=" + price +
", equipment='" + equipment + '\'' +
", country='" + country + '\'' +
", saleDate=" + saleDate +
", buyerName='" + buyerName + '\'' +
'}';
}
```

```
public static void main(String[] args) {
Car car = new Car("Toyota", 2022, 25000.0, "Full", "Japan", new Date(), "John Doe");
System.out.println(car);
}
}
```

Вывод.

Сегодня я изучил и научился работать с основными методами ООП на примере данной программы . Этот простой, но показательный пример позволил мне закрепить понимание фундаментальных концепций, таких как классы, объекты, инкапсуляция, конструкторы и полиморфизм (через переопределение метода toString()).

Изучив этот код , я осознал, как класс Car служит чертежом или шаблоном для создания объектов, каждый из которых представляет собой конкретный автомобиль. Внутри класса определены поля (переменные экземпляра), такие как brand, year, price и другие, которые хранят информацию о состоянии объекта Car. Использование модификатора private для этих полей подчеркивает важность инкапсуляции – принципа сокрытия внутренних деталей реализации класса и предоставления доступа к ним только через определенные методы (геттеры и сеттеры). Это позволяет контролировать изменение состояния объекта и предотвращает возможность случайного повреждения данных.

Конструктор класса Car играет ключевую роль в создании новых объектов. Он инициализирует поля объекта с заданными значениями, обеспечивая корректное начальное состояние. Я понял, что конструктор – это особый метод, который вызывается автоматически при создании объекта с использованием ключевого слова new. Конструктор гарантирует, что каждый объект Car будет создан с допустимыми и согласованными значениями атрибутов. Отсутствие явного конструктора привело бы к использованию конструктора по умолчанию, который, возможно, не инициализировал бы поля класса желаемым образом.

Особое внимание было уделено методу toString(). Я научился, что этот метод по умолчанию наследуется от класса Object и возвращает строковое представление объекта. Однако, переопределив метод toString() в классе Car, мы можем создать более информативное и понятное строковое представление объекта, включающее значения всех его полей. Это особенно

полезно при отладке и логировании, когда необходимо быстро и легко получить информацию о состоянии объекта. Вывод, генерируемый toString(), четко показывает значения марки, года выпуска, цены и других характеристик автомобиля. Без переопределения toString(), вывод был бы просто Car@xxxxxxx, что не несет никакой полезной информации.

В заключение, Car.java – это ценный пример, который помог мне закрепить понимание основных концепций ООП в Java. Я научился создавать классы, определять поля и методы, использовать конструкторы для инициализации объектов, переопределять метод toString() для информативного представления объектов и контролировать доступ к полям через геттеры и сеттеры. Эти знания будут полезны при разработке более сложных и функциональных Java-приложений. Дальнейшее изучение и расширение программы Car.java позволит мне углубить свои навыки в ООП и научиться применять эти концепции на практике.