**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Лабораторная работа №2

**Объектно-ориентированное программирование**  
**по дисциплине  
«Информационные технологии и программирование»**

Выполнил: студент гр. БВТ2403

Лашков Ю.Е.

Руководитель:

Москва, 2025 г.

**Цель работы:**изучить основные принципы объектно-ориентированного программирования в языке Java и реализовать иерархию классов с применением инкапсуляции, наследования, полиморфизма и абстракции.

**Индивидуальное задание:**Разработать иерархию классов по теме «Оружие». Базовый класс — Weapon. Дочерние классы — Sword, Bow и MagicWand. Предусмотреть наличие конструкторов, геттеров и сеттеров, а также продемонстрировать применение принципов ООП при работе с объектами.

**Основная часть:**  
Объектно-ориентированное программирование (ООП) — это подход к разработке программ, основанный на использовании объектов и классов. В Java данная парадигма лежит в основе языка и обеспечивает его гибкость и структурированность. Основными принципами ООП являются инкапсуляция, наследование, полиморфизм и абстракция.

Инкапсуляция заключается в сокрытии внутренней реализации объекта и предоставлении доступа к данным только через методы класса. Это повышает надёжность программы и защищает данные от несанкционированного изменения. В Java для этого используются модификаторы доступа, которые определяют область видимости полей и методов.

Наследование позволяет создавать новые классы на основе уже существующих, что обеспечивает повторное использование кода и упрощает его сопровождение. Класс-наследник получает все свойства и методы родительского класса, при необходимости добавляя собственные.

Полиморфизм обеспечивает возможность использования одного интерфейса для работы с объектами разных типов. В Java он проявляется в перегрузке и переопределении методов, что делает программы более гибкими.

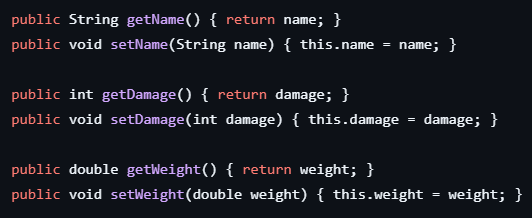
Абстракция позволяет выделить ключевые характеристики объектов, скрывая детали реализации. В языке Java для этого применяются абстрактные классы и интерфейсы, задающие общие шаблоны поведения для наследников.

**Реализация задания лабораторной работы:**

В начале работы был создан абстрактный класс **Weapon**, который служит базой для всех видов оружия. В нём определены общие поля — название, урон и вес, а также конструкторы для их инициализации. Кроме того, реализован метод **showInfo()**, выводящий информацию об объекте, и абстрактный метод **attack()**, который должен быть переопределён в каждом подклассе.



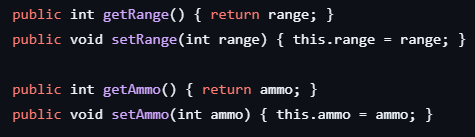
Для обеспечения инкапсуляции были добавлены геттеры и сеттеры для всех полей.



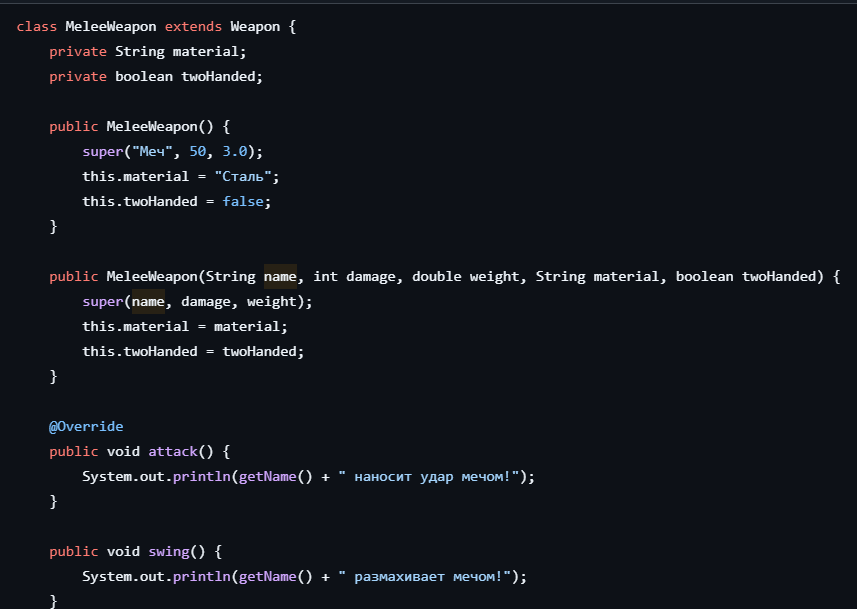
Следующим шагом я создал класс **RangedWeapon**, который наследуется от абстрактного класса Weapon и описывает оружие дальнего боя. В этом классе добавлены новые поля — дальность стрельбы и количество стрел. Метод **attack()** переопределён таким образом, чтобы при каждом выстреле уменьшалось количество стрел, а при их отсутствии выводилось сообщение о невозможности атаки. Кроме того, реализован метод **reload()**, позволяющий пополнить запас стрел.



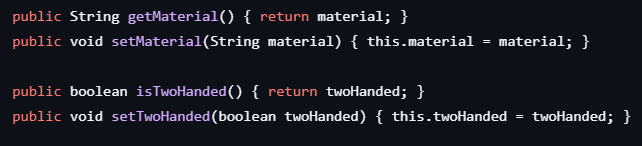
Для удобства работы с полями добавлены геттеры и сеттеры.



Далее был создан класс **MeleeWeapon**, наследующий абстрактный класс Weapon и описывающий оружие ближнего боя. В этом классе добавлены поля **material** (материал оружия) и **twoHanded** (дополнение о том, что оружие двуручное). Метод **attack()** переопределён для демонстрации удара мечом, а дополнительный метод **swing()** показывает размахивание оружием.

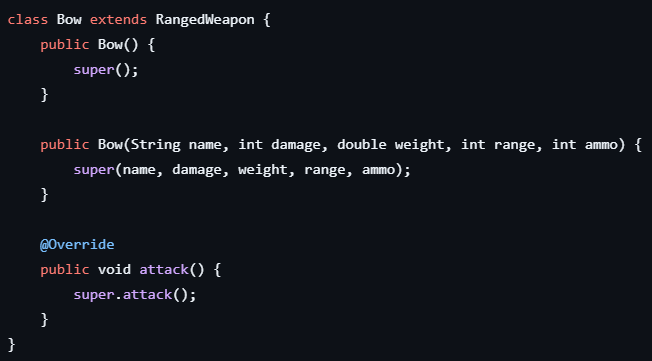


Для работы с полями реализованы геттеры и сеттеры.



После этого был создан класс **Sword**, наследующий **MeleeWeapon**. В классе реализован счётчик созданных объектов **swordCount**, который увеличивается при каждом создании экземпляра. Метод **attack()** переопределён для демонстрации удара мечом в бою. Также добавлен статический метод **getSwordCount()**, позволяющий узнать количество созданных объектов Sword.



Также был создан класс **Bow**, наследующий **RangedWeapon** и реализующий оружие дальнего боя. В классе переопределён метод **attack()**, который использует функциональность родительского класса для стрельбы и уменьшения количества стрел. Конструкторы позволяют создавать объекты с настройками по умолчанию или с указанием конкретных параметров оружия. 

В конце работы был создан класс **Main**, в котором продемонстрировано создание и использование объектов всех разработанных классов. Были созданы экземпляры мечей (**Sword**) и луков (**Bow**), после чего выполнен вывод информации о каждом объекте с помощью метода **showInfo()**, а также проверена работа методов **attack()** и **reload()**. Дополнительно продемонстрирован счётчик созданных объектов мечей с использованием статического метода **getSwordCount()**.

**Вывод:**

В ходе работы я научился создавать и использовать классы и объекты, применять наследование и полиморфизм, работать с конструкторами, методами и статическими переменными.

<https://github.com/deylon999/ITAP/tree/master/2lab>