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

Отчет по лабораторной работе № 2.

Вариант 13.

Выполнил: студент группы БПИ2401

Костиль Антон Валерьевич

Проверил: Харрасов Камиль Раисович

Москва, 2025

Цель работы: ознакомление с принципами объектно-ориентированного программирования (ООП) в Java и их практическое применение

Ход работы:

Задание 1. Создайте иерархию классов в соответствии с вариантом. Ваша иерархия должна содержать:

• абстрактный класс;

• два уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведение объекта);

• демонстрацию реализации всех принципов ООП; • наличие конструкторов (в том числе по умолчанию);

• наличие геттеров и сеттеров;

• ввод/вывод информации о создаваемых объектах; • предусмотрите в одном из классов создание счетчика созданных объектов с использованием статической переменной, продемонстрируйте работу

Вариант 13. Базовый класс: Оружие. Дочерние классы: Меч, Лук, Волшебная палочка.

Шаг 1. Создаем абстрактный класс Weapon, от которого будут наследоваться все последующие

```
public abstract class Weapon {
    protected String name;
    protected int damage;
    protected double weight;
    protected static int count;

    public Weapon() {
        count++;
    }

    public Weapon(String name, int damage, double weight) {
        this.name = name;
        this.damage = damage;
```

```
        this.weight = weight;
        count++;
    }

    public String getName() { return name; }
    public void setName(String name) {
        this.name = name;
    }

    public int getDamage() {
        return damage;
    }
    public void setDamage(int damage) {
        this.damage = damage;
    }

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

    public abstract void attack();
    public abstract void showInfo();

    public static void showCount() {
        System.out.println("weapons created: " + count);
    }
}
```

Шаг 2. Создаем класс Sword, наследуясь от класса Weapon. Здесь мы описываем конструкторы, переопределяем методы суперкласса, пишем геттеры и сеттеры.

```
public class Sword extends Weapon {
    private String bladeMaterial;
    private boolean twoHanded;

    public Sword() {
        super();
        this.name = "unnamed sword";
        this.damage = 10;
        this.weight = 2.5;
        this.bladeMaterial = "steel";
        this.twoHanded = false;
    }
```

```
    public Sword(String name, int damage, double weight, String bladeMaterial,
boolean twoHanded) {
        super(name, damage, weight);
        this.bladeMaterial = bladeMaterial;
        this.twoHanded = twoHanded;
    }

    @Override
    public void attack() {
        System.out.println("sword strike, damage: " + damage);
    }

    @Override
    public void showInfo() {
        System.out.println("name: " + name);
        System.out.println("damage: " + damage);
        System.out.println("weight: " + weight + " kg");
        System.out.println("blade material : " + bladeMaterial);
        System.out.println("two-handed: " + (twoHanded ? "yes" : "no"));
    }

    public String getBladeMaterial() {
        return bladeMaterial;
    }

    public void setBladeMaterial(String bladeMaterial) {
        this.bladeMaterial = bladeMaterial;
    }

    public boolean isTwoHanded() {
        return twoHanded;
    }

    public void setTwoHanded(boolean twoHanded) {
        this.twoHanded = twoHanded;
    }
}
```

Шаг 3. Создаем класс Bow, наследуясь от класса Weapon. Также, как и ранее, описываем конструкторы, переопределяем методы суперкласса, пишем геттеры и сеттеры.

```
public class Bow extends Weapon {
    private int range;
    private int arrowsCount;

    public Bow() {
        super();
        this.name = "unnamed bow";
```

```
        this.damage = 15;
        this.weight = 3.0;
        this.range = 30;
        this.arrowsCount = 10;
    }

    public Bow(String name, int damage, double weight, int range, int
arrowsCount) {
        super(name, damage, weight);
        this.range = range;
        this.arrowsCount = arrowsCount;
    }

    @Override
    public void attack() {
        if (arrowsCount > 0) {
            System.out.println(name + " fired, damage: " + damage);
            arrowsCount--;
        } else {
            System.out.println(name + " can`t atack, no arrows");
        }
    }

    @Override
    public void showInfo() {
        System.out.println("name: " + name);
        System.out.println("damage: " + damage);
        System.out.println("qeight: " + weight + " kg");
        System.out.println("range: " + range + " m");
        System.out.println("arrows: " + arrowsCount);
    }

    public int getRange() {
        return range;
    }

    public void setRange(int range) {
        this.range = range;
    }

    public int getArrowsCount() {
        return arrowsCount;
    }

    public void setArrowsCount(int arrowsCount) {
        this.arrowsCount = arrowsCount;
    }
}
```

Шаг 4. Аналогичным образом создаем класс MagicWand

```
public class MagicWand extends Weapon {
    private int mana;
    private String spellType;

    public MagicWand() {
        super();
        this.name = "unnamed magic wand";
        this.damage = 12;
        this.weight = 1.0;
        this.mana = 50;
        this.spellType = "fire";
    }

    public MagicWand(String name, int damage, double weight, int mana, String
spellType) {
        super(name, damage, weight);
        this.mana = mana;
        this.spellType = spellType;
    }

    @Override
    public void attack() {
        System.out.println(name + " attacks with a spell (" + spellType + "),
damage: " + damage);
    }

    @Override
    public void showInfo() {
        System.out.println("name: " + name);
        System.out.println("damage: " + damage);
        System.out.println("weight: " + weight + " kg");
        System.out.println("spell type: " + spellType);
        System.out.println("mana: " + mana);
    }

    public int getMana() {
        return mana;
    }

    public void setMana(int mana) {
        this.mana = mana;
    }

    public String getSpellType() {
        return spellType;
    }

    public void setSpellType(String spellType) {
        this.spellType = spellType;
```

```
    }
}
```

Шаг 5. Тестирование и демонстрация

Напишем Main.java, в котором создадим по 2 объекта каждого класса (Sword, Bow, MagicWand. Проверим работу методов showInfo(), attack(), проверим геттеры и сеттеры. Вывод программы представлен на рисунках 1 и 2.

```
public class Main {
    public static void main(String[] args) {
        Sword sword1 = new Sword();
        Sword sword2 = new Sword("paladin sword", 25, 4.0, "metal", true);

        Bow bow1 = new Bow();
        Bow bow2 = new Bow("Bow", 20, 3.5, 50, 15);

        MagicWand wand1 = new MagicWand();
        MagicWand wand2 = new MagicWand("supercharged", 30, 0.2, 100,
"lightning");

        sword1.showInfo();
        sword2.showInfo();

        bow1.showInfo();
        bow2.showInfo();

        wand1.showInfo();
        wand2.showInfo();

        //полиморфизм
        Weapon[] weapons = {sword1, sword2, bow1, bow2, wand1, wand2};

        for (Weapon w : weapons) {
            w.attack();
        }

        sword1.setTwoHanded(true);
        bow1.setArrowsCount(0);
        wand1.setMana(0);

        //после изменений
        sword1.showInfo();
        bow1.showInfo();
        wand1.showInfo();

        System.out.println("blade material is: " + sword2.getBladeMaterial());
        System.out.println("range " + bow2.getRange());
```

```
        System.out.println("spell type is: " + wand2.getSpellType());

        Weapon.showCount();
    }
}
```

```
seled@DESKTOP-EL2FK86 MINGW64 /d/учеба/ИТиП/lab2 (main)
$ java Main
sword information
name: unnamed sword
damage: 10
weight: 2.5 kg
blade material : steel
two-handed: no
sword information
name: paladin sword
damage: 25
weight: 4.0 kg
blade material : metal
two-handed: yes
bow information
name: unnamed bow
damage: 15
qeight: 3.0 kg
range: 30 m
arrows: 10
bow information
name: Bow
damage: 20
qeight: 3.5 kg
range: 50 m
arrows: 15
magic wand information
name: unnamed magic wand
damage: 12
weight: 1.0 kg
spell type: fire
mana: 50
```

Рисунок 1 – вывод программы

```
magic wand information
name: supercharged
damage: 30
weight: 0.2 kg
spell type: lightning
mana: 100
sword strike, damage: 10
sword strike, damage: 25
unnamed bow fired, damage: 15
Bow fired, damage: 20
unnamed magic wand attacks with a spell (fire), damage: 12
supercharged attacks with a spell (lightning), damage: 30
sword information
name: unnamed sword
damage: 10
weight: 2.5 kg
blade material : steel
two-handed: yes
bow information
name: unnamed bow
damage: 15
qeight: 3.0 kg
range: 30 m
arrows: 0
magic wand information
name: unnamed magic wand
damage: 12
weight: 1.0 kg
spell type: fire
mana: 0
blade material is: metal
range 50
spell type is: lightning
weapons created: 6
```

Рисунок 2 – вывод программы (продолжение)

Ответы на контрольные вопросы:

1. Что такое абстракция и как она реализуется в языке Java?

Абстракция — это принцип ООП, который позволяет выделять общие свойства и поведение объектов, скрывая детали реализации.
В Java реализуется через абстрактные классы и интерфейсы

2. Что такое инкапсуляция и как она реализуется в языке Java?

Инкапсуляция — это скрытие данных объекта от прямого доступа извне и предоставление доступа через методы. В Java реализуется объявлением полей как private, созданием геттеров и сеттеров

3. Что такое наследование и как onj реализуется в языке Java?

Наследование — это механизм, когда один класс (подкласс) наследует поля и методы другого класса (суперкласса)**.** В Java реализуется через ключевое слово extends

4. Что такое полимофизм и как он реализуется в языке Java?

Полиморфизм — это способность объектов разных классов реагировать на одинаковые вызовы методов по-своему.
В Java реализуется через переопределение методов (@Override), ссылки на суперкласс, которые могут хранить объекты подклассов

5. Что такое множественное наследование и есть ли оно в Java?

Множественное наследование — когда класс наследует сразу несколько классов. В Java множественного наследования классов нет

6. Для чего нужно ключевое слово final?

Для переменных — значение нельзя изменить после присвоения.

Для методов — метод нельзя переопределить в подклассах.

Для классов — класс нельзя наследовать.

7. Какие в Java есть модификаторы доступа?

Private – доступ только внутри класса

Default (нет модификатора) – доступ только внутри пакета

Protected – доступ в пакете и у подклассов

Public - доступ везде

8. Что такое конструктор? Какие типы конструкторов бывают в Java?

Конструктор — это метод, который вызывается при создании объекта, чтобы инициализировать его поля.
Типы:

- По умолчанию — без параметров, задаёт стандартные значения.
- С параметрами — для передачи значений при создании объекта.

9. Для чего нужно ключевое слово this в Java?

this— ссылка на текущий объект.

Используется:

- Для различения полей и параметров с одинаковыми именами
- Для передачи текущего объекта в метод

10. Для чего нужно ключевое слово super в Java?

super — ссылка на суперкласс.
Используется:

- Для вызова конструктора суперкласса: super(name, damage)
- Для вызова метода суперкласса, если он переопределён: super.showInfo()

11. Что такое геттеры и сеттеры? Зачем они нужны?

Геттер— возвращает значение поля. Сеттер— изменяет значение поля. Они нужны для инкапсуляции, чтобы управлять доступом к приватным полям объекта.

12. Что такое переопределение?

Переопределение (override) — это изменение поведения метода суперкласса в подклассе.

13. Что такое перегрузка?

Перегрузка методов (overloading) — это наличие нескольких методов с одним именем, но разными параметрами. Компилятор сам решает, какой метод вызвать в зависимости от переданных аргументов.

Вывод: в ходе выполнения работы были освоены основные принципы объектно-ориентированного программирования на языке Java: создание абстрактного класса, наследование, инкапсуляция, полиморфизм, перегрузка и переопределение методов. Были разработаны и реализованы классы Weapon, Sword, Bow и MagicWand, продемонстрированы их взаимодействие и работа методов. Работа размещена на GitHub - https://github.com/Antosha044/ITiP