**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 8.

Выполнил: студент группы БПИ2401

Голуб Дмитрий Сергеевич

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 (классыдолжны 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

8. Базовый класс: Мебель. Дочерние классы: Стол, Стул, Кровать.

Шаг 1. Создаем абстрактный класс Furniture, 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 Table, наследуясь от класса Furniture. Здесь мы описываем конструкторы, переопределяем методы суперкласса, пишем геттеры и сеттеры.

class Table extends Furniture {  
 private String shape;  
 private int seats;  
 private boolean extendable;  
  
 public Table() {  
 this("Обеденный стол", "дерево", "орех", *18.0*, *15990.0*,  
 "прямоугольный", *6*, *true*);  
 }  
  
 public Table(String name, String material, String color, double weightKg, double price,  
 String shape, int seats, boolean extendable) {  
 super(name, material, color, weightKg, price);  
 this.shape = shape;  
 this.seats = seats;  
 this.extendable = extendable;  
 }  
  
 public String getShape() { return shape; }  
 public void setShape(String shape) { this.shape = shape; }  
  
 public int getSeats() { return seats; }  
 public void setSeats(int seats) { this.seats = seats; }  
  
 public boolean isExtendable() { return extendable; }  
 public void setExtendable(boolean extendable) { this.extendable = extendable; }  
  
 public void extend() {  
 if (extendable) {  
 seats += *2*;  
 System.out.println(name + ": стол разложен, теперь мест: " + seats);  
 } else {  
 System.out.println(name + ": стол не раскладывается.");  
 }  
 }  
  
 @Override  
 public void use() {  
 System.out.println(name + ": сервируем и принимаем пищу.");  
 }  
  
 @Override  
 public void showInfo() {  
 System.out.println("=== Стол ===");  
 System.out.printf(Locale.US,  
 "Название: %s%nМатериал: %s%nЦвет: %s%nВес: %.1f кг%nЦена: %.2f%n",  
 name, material, color, weightKg, price);  
 System.out.println("Форма: " + shape);  
 System.out.println("Количество мест: " + seats);  
 System.out.println("Раскладной: " + (extendable ? "да" : "нет"));  
 System.out.println();  
 }  
}

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

class Chair extends Furniture {  
 private boolean hasArmrests;  
 private int heightCm;  
 private int maxLoadKg;  
  
 public Chair() {  
 this("Офисный стул", "пластик/ткань", "чёрный", *9.0*, *7990.0*,  
 *true*, *45*, *120*);  
 }  
  
 public Chair(String name, String material, String color, double weightKg, double price,  
 boolean hasArmrests, int heightCm, int maxLoadKg) {  
 super(name, material, color, weightKg, price);  
 this.hasArmrests = hasArmrests;  
 this.heightCm = heightCm;  
 this.maxLoadKg = maxLoadKg;  
 }  
  
 public boolean isHasArmrests() { return hasArmrests; }  
 public void setHasArmrests(boolean hasArmrests) { this.hasArmrests = hasArmrests; }  
  
 public int getHeightCm() { return heightCm; }  
 public void setHeightCm(int heightCm) { this.heightCm = heightCm; }  
  
 public int getMaxLoadKg() { return maxLoadKg; }  
 public void setMaxLoadKg(int maxLoadKg) { this.maxLoadKg = maxLoadKg; }  
  
 public void adjustHeight(int delta) {  
 heightCm = Math.max(*35*, Math.min(*60*, heightCm + delta));  
 System.out.println(name + ": высота отрегулирована до " + heightCm + " см.");  
 }  
  
 @Override  
 public void use() {  
 System.out.println(name + ": садимся за работу или отдых.");  
 }  
  
 @Override  
 public void showInfo() {  
 System.out.println("=== Стул ===");  
 System.out.printf(Locale.US,  
 "Название: %s%nМатериал: %s%nЦвет: %s%nВес: %.1f кг%nЦена: %.2f%n",  
 name, material, color, weightKg, price);  
 System.out.println("Подлокотники: " + (hasArmrests ? "есть" : "нет"));  
 System.out.println("Высота сиденья: " + heightCm + " см");  
 System.out.println("Макс. нагрузка: " + maxLoadKg + " кг");  
 System.out.println();  
 }  
}

Шаг 4. Аналогичным образом создаем класс Bed

class Bed extends Furniture {  
 private String size;  
 private String mattress;  
 private boolean hasStorage;  
  
 public Bed() {  
 this("Кровать «Соната»", "массив", "белый", *35.0*, *24990.0*,  
 "двуспальная", "ортопедический", *true*);  
 }  
  
 public Bed(String name, String material, String color, double weightKg, double price,  
 String size, String mattress, boolean hasStorage) {  
 super(name, material, color, weightKg, price);  
 this.size = size;  
 this.mattress = mattress;  
 this.hasStorage = hasStorage;  
 }  
  
 public String getSize() { return size; }  
 public void setSize(String size) { this.size = size; }  
  
 public String getMattress() { return mattress; }  
 public void setMattress(String mattress) { this.mattress = mattress; }  
  
 public boolean isHasStorage() { return hasStorage; }  
 public void setHasStorage(boolean hasStorage) { this.hasStorage = hasStorage; }  
  
 public void makeBed() {  
 System.out.println(name + ": кровать застелена, всё готово ко сну.");  
 }  
  
 @Override  
 public void use() {  
 System.out.println(name + ": спим и отдыхаем.");  
 }  
  
 @Override  
 public void showInfo() {  
 System.out.println("=== Кровать ===");  
 System.out.printf(Locale.US,  
 "Название: %s%nМатериал: %s%nЦвет: %s%nВес: %.1f кг%nЦена: %.2f%n",  
 name, material, color, weightKg, price);  
 System.out.println("Размер: " + size);  
 System.out.println("Матрас: " + mattress);  
 System.out.println("Ящики для хранения: " + (hasStorage ? "есть" : "нет"));  
 System.out.println();  
 }  
}

Шаг 5. Тестирование и демонстрация

Напишем Main.java, в котором создадим по 2 объекта каждого класса (Table, Chair, Bed. Проверим работу методов showInfo(), use(), проверим геттеры и сеттеры. Вывод программы представлен на рисунках 1 и 2.

public class Main {  
 public static void main(String[] args) {  
 Table t1 = new Table();  
 Chair c1 = new Chair();  
 Bed b1 = new Bed();  
  
 Table t2 = new Table("Журнальный столик", "стекло/металл", "прозрачный",  
 *8.0*, *10990.0*, "круглый", *4*, *false*);  
  
 Chair c2 = new Chair("Стул «Олимп»", "бук", "орех", *6.5*, *4990.0*,  
 *false*, *43*, *140*);  
  
 Bed b2 = new Bed("Кровать «Норд»", "ЛДСП", "дуб сонома", *29.0*, *18990.0*,  
 "полуторная", "пружинный", *false*);  
  
 t1.showInfo(); t2.showInfo();  
 c1.showInfo(); c2.showInfo();  
 b1.showInfo(); b2.showInfo();  
  
 Furniture[] items = { t1, t2, c1, c2, b1, b2 };  
 for (Furniture f : items) f.use();  
  
 t1.extend();  
 c1.adjustHeight(+*5*);  
 b1.makeBed();  
  
 c2.setHasArmrests(*true*);  
 System.out.println(c2.getName() + ": подлокотники теперь " + (c2.isHasArmrests() ? "есть" : "нет"));  
 t2.setSeats(*2*);  
 System.out.println(t2.getName() + ": пересчитаны места = " + t2.getSeats());  
 b2.setMattress("латексный");  
 System.out.println(b2.getName() + ": матрас -> " + b2.getMattress());  
  
 Furniture.showCount();  
 }  
}

**Вывод программы:**

dmitrygolub@MacBook-Pro-Dmitry lab2 % java Main

=== Стол ===

Название: Обеденный стол

Материал: дерево

Цвет: орех

Вес: 18.0 кг

Цена: 15990.00

Форма: прямоугольный

Количество мест: 6

Раскладной: да

=== Стол ===

Название: Журнальный столик

Материал: стекло/металл

Цвет: прозрачный

Вес: 8.0 кг

Цена: 10990.00

Форма: круглый

Количество мест: 4

Раскладной: нет

=== Стул ===

Название: Офисный стул

Материал: пластик/ткань

Цвет: чёрный

Вес: 9.0 кг

Цена: 7990.00

Подлокотники: есть

Высота сиденья: 45 см

Макс. нагрузка: 120 кг

=== Стул ===

Название: Стул «Олимп»

Материал: бук

Цвет: орех

Вес: 6.5 кг

Цена: 4990.00

Подлокотники: нет

Высота сиденья: 43 см

Макс. нагрузка: 140 кг

=== Кровать ===

Название: Кровать «Соната»

Материал: массив

Цвет: белый

Вес: 35.0 кг

Цена: 24990.00

Размер: двуспальная

Матрас: ортопедический

Ящики для хранения: есть

=== Кровать ===

Название: Кровать «Норд»

Материал: ЛДСП

Цвет: дуб сонома

Вес: 29.0 кг

Цена: 18990.00

Размер: полуторная

Матрас: пружинный

Ящики для хранения: нет

Обеденный стол: сервируем и принимаем пищу.

Журнальный столик: сервируем и принимаем пищу.

Офисный стул: садимся за работу или отдых.

Стул «Олимп»: садимся за работу или отдых.

Кровать «Соната»: спим и отдыхаем.

Кровать «Норд»: спим и отдыхаем.

Обеденный стол: стол разложен, теперь мест: 8

Офисный стул: высота отрегулирована до 50 см.

Кровать «Соната»: кровать застелена, всё готово ко сну.

Стул «Олимп»: подлокотники теперь есть

Журнальный столик: пересчитаны места = 2

Кровать «Норд»: матрас -> латексный

Создано объектов мебели: 6

**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 онj 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 Furniture, Table, Chair и Bed, продемонстрированы их взаимодействие и работа методов. Работа размещена на GitHub - <https://github.com/DmitryGolub/itip>