**2гМИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

### ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине: «Введение в разработку программного обеспечения»

на тему: **«**Развитие представлений о разработке программ: от спагетти-кода к методологии структурного программирования. Объектно-ориентированное программирование в *Java***»**

Выполнил: студент гр. ИТИ-21

Здановская А. С.

Принял преподаватель:

Карась О. В.

Гомель 2025

**Цель работы:** изучить технологии проектных решений в *Java.*

**Задание:**

Спроектировать проектное решение для приложения, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы *equals()*, *hashCode()*, *toString()*.

Вариант 5.

Создать объект класса Планета, используя класс Материк. Методы: вывести на консоль название материка, планеты, количество материков.

**Ход работы:**

Были созданы три класса: *Figure*, *Continent, Planet* и *PlanetDemo.*

Создана иерархия классов для компонентов компьютера. Абстрактный класс Component служит базовым классом для всех компонентов и содержит общее поле name (название) и абстрактный метод printInfo() для вывода информации.

Класс Winchester (винчестер) наследуется от Component и добавляет поля size (размер в ГБ) и hasViruses (наличие вирусов). Реализованы методы проверки на вирусы checkForViruses() (со случайным результатом) и очистки от вирусов cleanViruses().

Класс DiskDrive (дисковод) наследуется от Component и добавляет поле type (тип носителя). Класс RAM (оперативная память) наследуется от Component и добавляет поля capacity (объем в ГБ) и frequency (частота в МГц).

Класс Computer представляет компьютер и содержит поля: название, компоненты (винчестер, дисковод, ОЗУ) и статус питания isPoweredOn. Реализованы методы включения/выключения питания, проверки на вирусы, вывода размера винчестера и информации о всех компонентах.

В демонстрационном классе ComputerDemo показана работа системы: создаются компоненты и компьютер, демонстрируются основные методы, вывод информации о компонентах, работа методов toString(), equals() и hashCode(), работа с отдельными компонентами и управление питанием.

Для всех классов переопределены стандартные методы Object: toString() для строкового представления, equals() для сравнения объектов по значимым полям и hashCode() для генерации хэш-кодов на основе тех же полей.

На рисунке 1 показана структура папки, в которой и велось выполнение лабораторной работы.

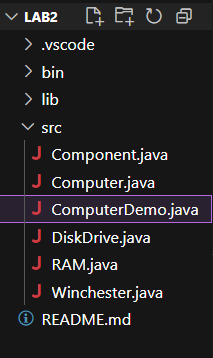


Рисунок 1 - Архитектура папки

Результат выполнения программы представлен на рисунке 2.

Листинг программы представлен в приложении А.

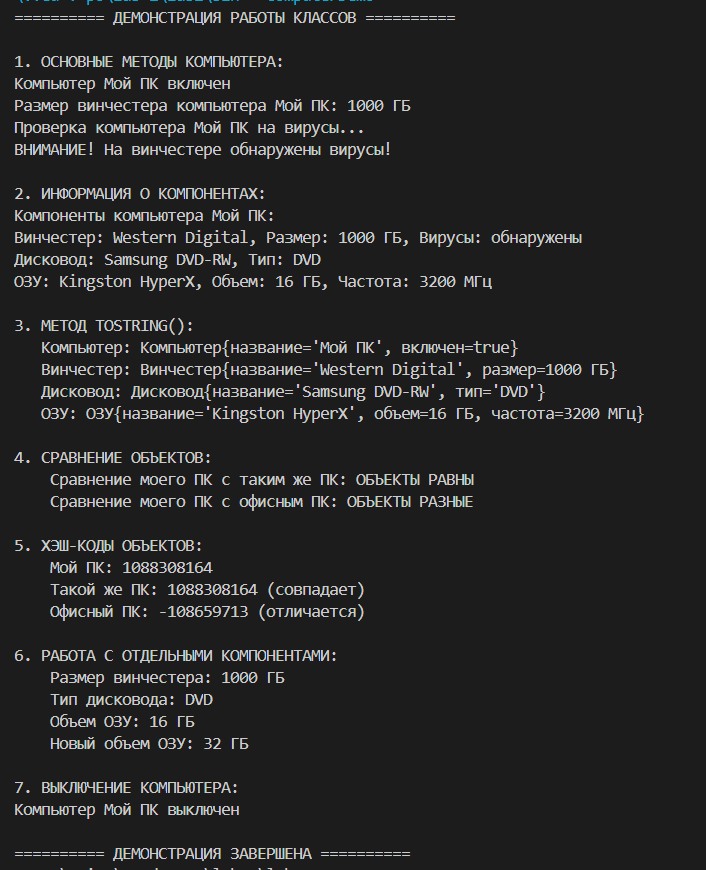


Рисунок 1 – Результат выполнения программы

**Вывод**: В ходе выполнения лабораторной работы был успешно спроектирован и реализован программный комплекс для работы с объектами "Компьютер" и его компонентами. Разработанная система демонстрирует практическое применение принципов объектно-ориентированного программирования, в частности композиции объектов, когда класс Computer агрегирует в себе объекты классов Winchester, DiskDrive и RAM.

Была создана иерархия из пяти классов: абстрактный класс Component, классы Winchester, DiskDrive и RAM (наследники Component), класс Computer и демонстрационный класс ComputerDemo. Каждый класс выполняет строго определенные функции в соответствии с принципом единственной ответственности.

**ПРИЛОЖЕНИЕ А**

/\*\*

 \* Абстрактный класс Компонент компьютера

 \*/

public abstract class Component {

    protected String name;

    public Component(String name) {

        this.name = name;

    }

    public abstract void printInfo();

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

}

/\*\*

 \* Класс Компьютер

 \*/

public class Computer {

    private String name;

    private Winchester winchester;

    private DiskDrive diskDrive;

    private RAM ram;

    private boolean isPoweredOn;

    public Computer(String name, Winchester winchester, DiskDrive diskDrive, RAM ram) {

        this.name = name;

        this.winchester = winchester;

        this.diskDrive = diskDrive;

        this.ram = ram;

        this.isPoweredOn = false;

    }

    /\*\*

     \* Включить компьютер

     \*/

    public void powerOn() {

        if (!isPoweredOn) {

            isPoweredOn = true;

            System.out.println("Компьютер " + name + " включен");

        } else {

            System.out.println("Компьютер " + name + " уже включен");

        }

    }

    /\*\*

     \* Выключить компьютер

     \*/

    public void powerOff() {

        if (isPoweredOn) {

            isPoweredOn = false;

            System.out.println("Компьютер " + name + " выключен");

        } else {

            System.out.println("Компьютер " + name + " уже выключен");

        }

    }

    /\*\*

     \* Проверить на вирусы

     \*/

    public void checkForViruses() {

        if (!isPoweredOn) {

            System.out.println("Сначала включите компьютер!");

            return;

        }

        System.out.println("Проверка компьютера " + name + " на вирусы...");

        boolean hasViruses = winchester.checkForViruses();

        if (hasViruses) {

            System.out.println("ВНИМАНИЕ! На винчестере обнаружены вирусы!");

        } else {

            System.out.println("Вирусы не обнаружены. Компьютер чист.");

        }

    }

    /\*\*

     \* Вывести размер винчестера на консоль

     \*/

    public void printWinchesterSize() {

        System.out.println("Размер винчестера компьютера " + name + ": " +

                          winchester.getSize() + " ГБ");

    }

    /\*\*

     \* Вывести информацию о всех компонентах

     \*/

    public void printAllComponentsInfo() {

        System.out.println("Компоненты компьютера " + name + ":");

        winchester.printInfo();

        diskDrive.printInfo();

        ram.printInfo();

    }

    // Геттеры и сеттеры

    public String getName() { return name; }

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

    public Winchester getWinchester() { return winchester; }

    public DiskDrive getDiskDrive() { return diskDrive; }

    public RAM getRam() { return ram; }

    public boolean isPoweredOn() { return isPoweredOn; }

    // Переопределение стандартных методов

    @Override

    public String toString() {

        return "Компьютер{название='" + name + "', включен=" + isPoweredOn + "}";

    }

    @Override

    public boolean equals(Object obj) {

        if (this == obj) return true;

        if (obj == null || getClass() != obj.getClass()) return false;

        Computer computer = (Computer) obj;

        return name.equals(computer.name);

    }

    @Override

    public int hashCode() {

        return name.hashCode();

    }

}

/\*\*

 \* Демонстрационный класс

 \*/

public class ComputerDemo {

    public static void main(String args[]) {

        // Создание компонентов компьютера

        Winchester hdd = new Winchester("Western Digital", 1000);

        DiskDrive dvd = new DiskDrive("Samsung DVD-RW", "DVD");

        RAM memory = new RAM("Kingston HyperX", 16, 3200);

        // Создание компьютера

        Computer myComputer = new Computer("Мой ПК", hdd, dvd, memory);

        System.out.println("========== ДЕМОНСТРАЦИЯ РАБОТЫ КЛАССОВ ==========\n");

        // 1. Основные методы

        System.out.println("1. ОСНОВНЫЕ МЕТОДЫ КОМПЬЮТЕРА:");

        myComputer.powerOn();

        myComputer.printWinchesterSize();

        myComputer.checkForViruses();

        // 2. Демонстрация информации о компонентах

        System.out.println("\n2. ИНФОРМАЦИЯ О КОМПОНЕНТАХ:");

        myComputer.printAllComponentsInfo();

        // 3. Демонстрация toString()

        System.out.println("\n3. МЕТОД TOSTRING():");

        System.out.println("   Компьютер: " + myComputer.toString());

        System.out.println("   Винчестер: " + hdd.toString());

        System.out.println("   Дисковод: " + dvd.toString());

        System.out.println("   ОЗУ: " + memory.toString());

        // 4. Демонстрация equals() и hashCode()

        System.out.println("\n4. СРАВНЕНИЕ ОБЪЕКТОВ:");

        Computer sameComputer = new Computer("Мой ПК",

            new Winchester("Seagate", 500),

            new DiskDrive("LG", "CD"),

            new RAM("Corsair", 8, 2666));

        Computer otherComputer = new Computer("Офисный ПК", hdd, dvd, memory);

        System.out.println("    Сравнение моего ПК с таким же ПК: " +

                          (myComputer.equals(sameComputer) ? "ОБЪЕКТЫ РАВНЫ" : "объекты разные"));

        System.out.println("    Сравнение моего ПК с офисным ПК: " +

                          (myComputer.equals(otherComputer) ? "объекты равны" : "ОБЪЕКТЫ РАЗНЫЕ"));

        System.out.println("\n5. ХЭШ-КОДЫ ОБЪЕКТОВ:");

        System.out.println("    Мой ПК: " + myComputer.hashCode());

        System.out.println("    Такой же ПК: " + sameComputer.hashCode() + " (совпадает)");

        System.out.println("    Офисный ПК: " + otherComputer.hashCode() + " (отличается)");

        // 6. Работа с отдельными компонентами

        System.out.println("\n6. РАБОТА С ОТДЕЛЬНЫМИ КОМПОНЕНТАМИ:");

        System.out.println("    Размер винчестера: " + myComputer.getWinchester().getSize() + " ГБ");

        System.out.println("    Тип дисковода: " + myComputer.getDiskDrive().getType());

        System.out.println("    Объем ОЗУ: " + myComputer.getRam().getCapacity() + " ГБ");

        // Изменим объем ОЗУ

        myComputer.getRam().setCapacity(32);

        System.out.println("    Новый объем ОЗУ: " + myComputer.getRam().getCapacity() + " ГБ");

        // 7. Выключение компьютера

        System.out.println("\n7. ВЫКЛЮЧЕНИЕ КОМПЬЮТЕРА:");

        myComputer.powerOff();

        System.out.println("\n========== ДЕМОНСТРАЦИЯ ЗАВЕРШЕНА ==========");

    }

}

/\*\*

 \* Класс Дисковод

 \*/

public class DiskDrive extends Component {

    private String type; // CD, DVD, Blu-ray

    public DiskDrive(String name, String type) {

        super(name);

        this.type = type;

    }

    @Override

    public void printInfo() {

        System.out.println("Дисковод: " + name + ", Тип: " + type);

    }

    // Геттеры и сеттеры

    public String getType() { return type; }

    public void setType(String type) { this.type = type; }

    // Переопределение стандартных методов

    @Override

    public String toString() {

        return "Дисковод{название='" + name + "', тип='" + type + "'}";

    }

    @Override

    public boolean equals(Object obj) {

        if (this == obj) return true;

        if (obj == null || getClass() != obj.getClass()) return false;

        DiskDrive diskDrive = (DiskDrive) obj;

        return name.equals(diskDrive.name) && type.equals(diskDrive.type);

    }

    @Override

    public int hashCode() {

        return name.hashCode() + type.hashCode();

    }

}

/\*\*

 \* Класс ОЗУ (оперативная память)

 \*/

public class RAM extends Component {

    private int capacity; // объем в ГБ

    private int frequency; // частота в МГц

    public RAM(String name, int capacity, int frequency) {

        super(name);

        this.capacity = capacity;

        this.frequency = frequency;

    }

    @Override

    public void printInfo() {

        System.out.println("ОЗУ: " + name + ", Объем: " + capacity + " ГБ, Частота: " + frequency + " МГц");

    }

    // Геттеры и сеттеры

    public int getCapacity() { return capacity; }

    public void setCapacity(int capacity) { this.capacity = capacity; }

    public int getFrequency() { return frequency; }

    public void setFrequency(int frequency) { this.frequency = frequency; }

    // Переопределение стандартных методов

    @Override

    public String toString() {

        return "ОЗУ{название='" + name + "', объем=" + capacity + " ГБ, частота=" + frequency + " МГц}";

    }

    @Override

    public boolean equals(Object obj) {

        if (this == obj) return true;

        if (obj == null || getClass() != obj.getClass()) return false;

        RAM ram = (RAM) obj;

        return capacity == ram.capacity &&

               frequency == ram.frequency &&

               name.equals(ram.name);

    }

    @Override

    public int hashCode() {

        return name.hashCode() + Integer.hashCode(capacity) + Integer.hashCode(frequency);

    }

}

/\*\*

 \* Класс Винчестер (жесткий диск)

 \*/

public class Winchester extends Component {

    private int size; // размер в ГБ

    private boolean hasViruses;

    public Winchester(String name, int size) {

        super(name);

        this.size = size;

        this.hasViruses = false;

    }

    @Override

    public void printInfo() {

        System.out.println("Винчестер: " + name + ", Размер: " + size + " ГБ, Вирусы: " +

                          (hasViruses ? "обнаружены" : "не обнаружены"));

    }

    // Проверка на вирусы

    public boolean checkForViruses() {

        // Имитация случайного обнаружения вирусов

        this.hasViruses = Math.random() > 0.7;

        return hasViruses;

    }

    public void cleanViruses() {

        this.hasViruses = false;

    }

    // Геттеры и сеттеры

    public int getSize() { return size; }

    public void setSize(int size) { this.size = size; }

    public boolean hasViruses() { return hasViruses; }

    // Переопределение стандартных методов

    @Override

    public String toString() {

        return "Винчестер{название='" + name + "', размер=" + size + " ГБ}";

    }

    @Override

    public boolean equals(Object obj) {

        if (this == obj) return true;

        if (obj == null || getClass() != obj.getClass()) return false;

        Winchester winchester = (Winchester) obj;

        return size == winchester.size && name.equals(winchester.name);

    }

    @Override

    public int hashCode() {

        return name.hashCode() + Integer.hashCode(size);

    }

}