**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Введение в разработку программного обеспечения»

на тему: **«**Развитие представлений о разработке программ. Объектно-ориентированное программирование в *Java***»**

Выполнил: студент гр. ИТД-21

Цагельник Ю. А.

Проверил: преподаватель

Малиновский И. Л.

Гомель 2024

**Цель работы:**

1. Разработать UML-диаграмму иерархии классов, согласно варианта (таблица 1).

2. При наименовании компонентов руководствоваться соглашением о наименовании.

3. При описании иерархии использовать наследование и композицию.

4. На основе UML-диаграммы разработать иерархию классов на языке Java.

5. Весь код должен быть снабжен элементами документирования.

6. Разработанную иерархию поместить в .jar файл для дальнейшего использования в качестве библиотечных классов.

7. Создать консольное приложение для демонстрации работы созданных классов.

8. Составить отчет о проделанной работе.

**Ход работы**

Задание.

1.1 Создать объект класса Компьютер, используя классы Винчестер, Дисковод, ОЗУ. Методы: включить, выключить, проверить на вирусы, вывести на консоль размер винчестера.

1.2 Создать не менее 10 объектов.

1.3 Добавить возможность докупать компоненты.

1.4 Вывести всю информацию о компьютере.

1.5 Вывести компьютеры, которые собирались вручную.

1.6 Подсчитать самый выгодный компьютер.

Запуск программы представлен на рисунке 1.

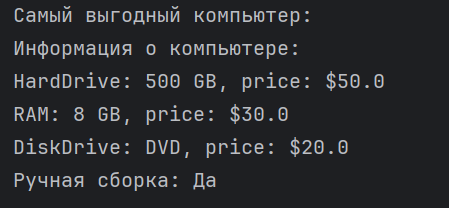


Рисунок 1 – Результат выполнения задания.

Листинг программы представлен в приложении А.

**Вывод**: В ходе выполнения лабораторной работы была разработана UML-диаграмма, описывающая структуру классов и их взаимодействие. На основе диаграммы реализована иерархия классов на языке Java с применением основных принципов объектно-ориентированного программирования, таких как наследование и композиция.

Была создана программа, демонстрирующая работу классов через консольное приложение, включающая такие функциональные возможности, как управление состоянием объектов, вывод информации о компонентах, и расчет стоимости. Все классы снабжены элементами документирования, что упрощает их дальнейшее использование.

Итогом работы является создание универсальной структуры кода, которая была упакована в .jar файл для дальнейшего использования как библиотечных**:** классов в других проектах.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программ**

«HardDrive.java»

public class HardDrive {

private int size; // Размер винчестера в ГБ

private double price; // Цена винчестера

public HardDrive(int size, double price) {

this.size = size;

this.price = price;

}

public int getSize() {

return size;

}

public double getPrice() {

return price;

}

@Override

public String toString() {

return "HardDrive: " + size + " GB, price: $" + price;

}

}

«Computer.java»

public class Computer {

private HardDrive hardDrive;

private RAM ram;

private DiskDrive diskDrive;

private boolean isManualAssembly; // Флаг для ручной сборки

private boolean isOn; // Состояние компьютера

public Computer(HardDrive hardDrive, RAM ram, DiskDrive diskDrive, boolean isManualAssembly) {

this.hardDrive = hardDrive;

this.ram = ram;

this.diskDrive = diskDrive;

this.isManualAssembly = isManualAssembly;

this.isOn = false;

}

public void turnOn() {

if (!isOn) {

isOn = true;

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

} else {

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

}

}

public void turnOff() {

if (isOn) {

isOn = false;

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

} else {

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

}

}

public void checkForViruses() {

if (isOn) {

System.out.println("Проверка на вирусы завершена. Вирусы не найдены.");

} else {

System.out.println("Компьютер выключен. Включите его для проверки на вирусы.");

}

}

public void printHardDriveSize() {

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

}

public void printInfo() {

System.out.println("Информация о компьютере:");

System.out.println(hardDrive);

System.out.println(ram);

System.out.println(diskDrive);

System.out.println("Ручная сборка: " + (isManualAssembly ? "Да" : "Нет"));

}

public double getTotalPrice() {

return hardDrive.getPrice() + ram.getPrice() + diskDrive.getPrice();

}

public boolean isManualAssembly() {

return isManualAssembly;

}

public void upgradeHardDrive(HardDrive newHardDrive) {

this.hardDrive = newHardDrive;

System.out.println("Жесткий диск был обновлен до: " + hardDrive);

}

public void upgradeRAM(RAM newRAM) {

this.ram = newRAM;

System.out.println("ОЗУ было обновлено до: " + ram);

}

public void upgradeDiskDrive(DiskDrive newDiskDrive) {

this.diskDrive = newDiskDrive;

System.out.println("Дисковод был обновлен до: " + diskDrive);

}

}

«RAM.java»

public class RAM {

private int size; // Размер ОЗУ в ГБ

private double price; // Цена ОЗУ

public RAM(int size, double price) {

this.size = size;

this.price = price;

}

public int getSize() {

return size;

}

public double getPrice() {

return price;

}

@Override

public String toString() {

return "RAM: " + size + " GB, price: $" + price;

}

}

«DiskDrive.java»

public class DiskDrive {

private String type; // Тип дисковода

private double price; // Цена дисковода

public DiskDrive(String type, double price) {

this.type = type;

this.price = price;

}

public String getType() {

return type;

}

public double getPrice() {

return price;

}

@Override

public String toString() {

return "DiskDrive: " + type + ", price: $" + price;

}

}

«Main.java»

public class Main {

public static void main(String[] args) {

// Создаем несколько компьютеров

Computer[] computers = new Computer[10];

computers[0] = new Computer(new HardDrive(500, 50), new RAM(8, 30), new DiskDrive("DVD", 20), true);

computers[1] = new Computer(new HardDrive(1000, 100), new RAM(16, 60), new DiskDrive("Blu-ray", 40), false);

computers[2] = new Computer(new HardDrive(2000, 200), new RAM(32, 120), new DiskDrive("DVD", 20), true);

computers[3] = new Computer(new HardDrive(500, 50), new RAM(8, 30), new DiskDrive("Blu-ray", 40), false);

computers[4] = new Computer(new HardDrive(750, 75), new RAM(12, 45), new DiskDrive("DVD", 20), true);

computers[5] = new Computer(new HardDrive(1500, 150), new RAM(24, 90), new DiskDrive("Blu-ray", 40), true);

computers[6] = new Computer(new HardDrive(1000, 100), new RAM(16, 60), new DiskDrive("DVD", 20), false);

computers[7] = new Computer(new HardDrive(2000, 200), new RAM(32, 120), new DiskDrive("Blu-ray", 40), true);

computers[8] = new Computer(new HardDrive(750, 75), new RAM(12, 45), new DiskDrive("DVD", 20), false);

computers[9] = new Computer(new HardDrive(1500, 150), new RAM(24, 90), new DiskDrive("Blu-ray", 40), true);

// Выводим информацию о каждом компьютере

for (Computer computer : computers) {

computer.printInfo();

System.out.println();

}

// Выводим компьютеры, которые были собраны вручную

System.out.println("Компьютеры, собранные вручную:");

for (Computer computer : computers) {

if (computer.isManualAssembly()) {

computer.printInfo();

}

}

// Находим самый выгодный компьютер

Computer bestValueComputer = computers[0];

for (Computer computer : computers) {

if (computer.getTotalPrice() < bestValueComputer.getTotalPrice()) {

bestValueComputer = computer;

}

}

System.out.println("Самый выгодный компьютер:");

bestValueComputer.printInfo();

}

}