****

**JAVA进阶编程第一次实验报告**



**学 院智能与计算学部**

**专 业 软件工程**

**年 级 2017**

**姓 名 刘兴宇**

**2019年 3 月 7 日**

# JAVA进阶编程第一次实验报告

1. 需求分析（描述具体需求）

某计算机组装公司主要销售各类组装计算机，计算机一般由CPU、内存、主板、硬盘等组件构成。具体组件信息如下：

|  |  |  |
| --- | --- | --- |
| 组件名 | 组件品牌 | 组件属性 |
| CPU | Intel、AMD | Name，coreNum，price |
| 内存 | Samsung, Kingston | Name, volume, price |
| 硬盘 | Seagate, WestDigitals | Name, volume, price |
| 主板 | Asus、Gigabyte | Name，speed, price |

对于所有的组件以及不同品牌，分别设计一个类来进行表示。每个组件拥有工作方法，简单地表示为“组件名称 + work”

设计一个Computer计算机类，包括名称、总价格、CPU、内存、硬盘、主办属性。并拥有展示方法，可以展示计算机各配件的参数。拥有工作方法，简单的表示为所有组件进行工作。

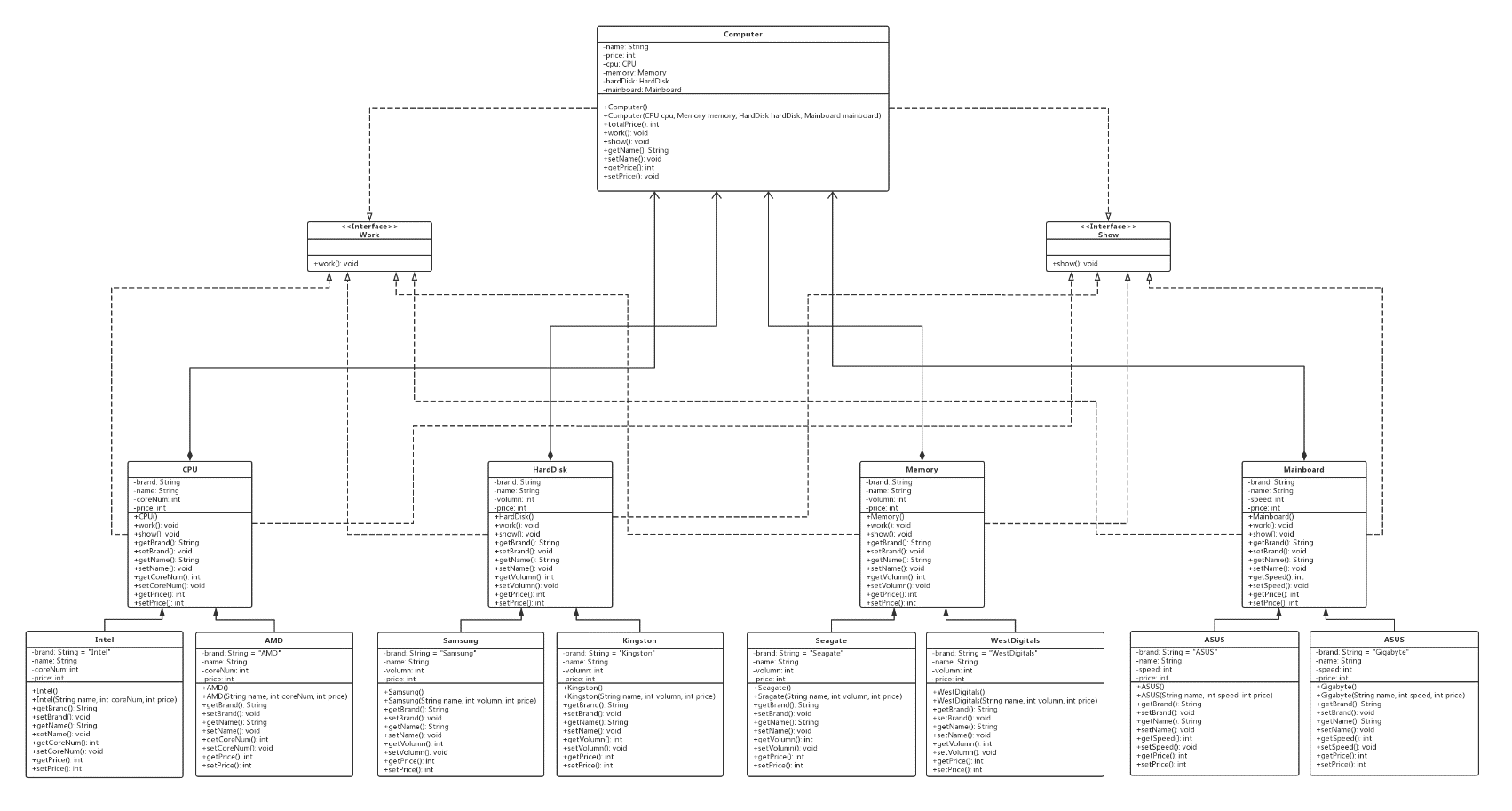
设计一个ComputerStore主类，包含三个Computer实例。在主方法内可以对每台计算机进行展示或工作。

1. 概要设计（简单描述设计思路，配合UML图）

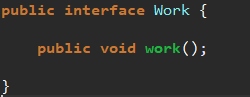
对于展示功能和工作功能，设计两个接口work、show用于实现，每个接口内只有对应的方法。

对于每种组件，设计一个父类（CPU\Memory\HardDisk\Mainboard）。父类中拥有表格对应的属性和一个品牌属性（以及get/set方法），并实现了work和show接口。对于每一种父类，分别设计对应品牌的子类继承于父类。

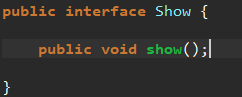
对于Computer类，拥有名称、总价格以及四个组件属性，拥有默认构造方法和带参数的构造方法，用于组装不同配置的计算机。拥有totalPrice方法，用于计算计算机的总价格。实现了work和show接口，分别调用各组件的work和show方法。

UML类图如下：（源文件/ex01/adv\_java\_ex01）

1. 详细设计（详细描述具体如何实现，附代码及说明）

1. Work接口：

2. Show接口：



3. 组件父类（CPU、Memory、HardDisk、Mainboard）：实现show、work接口

属性：对应表中的三个组件属性+品牌属性

方法：

无参构造方法

各属性对应的get、set方法

Show方法（override）：按格式输出字符串“组件名：品牌 名称 对应参数”

Work方法（override）：按格式输出字符串“组件名 work”

具体代码（以CPU类为例）

public abstract class CPU implements Work, Show{

private String brand;

private String name;

private int coreNum;

private int price;

public CPU() {

}

@Override

public void work() {

System.out.println("CPU work.");

}

@Override

public void show() {

System.out.println("CPU: " + this.getBrand() + " " + this.getName() + " coreNum: " + this.getCoreNum() + " price: " + this.getPrice());

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getCoreNum() {

return coreNum;

}

public void setCoreNum(int coreNum) {

this.coreNum = coreNum;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

}

4. 组件子类（Intel、AMD、Samsung、Kingston、Seagate、WestDigitals、ASUS、Gigabyte）：继承自对应的组件父类

属性：与父类相同，brand赋予初始值，为其父类类名。

方法：

无参构造方法

带三个参数的构造方法（对应除去brand的其它三个属性）

各属性的get和set方法

具体代码（以AMD为例）：

public class AMD extends CPU{

private String brand = "AMD";

private String name;

private int coreNum;;

private int price;;

public AMD() {

}

public AMD(String name, int coreNum, int price) {

this.name = name;

this.coreNum = coreNum;

this.price = price;

}

public String getName() {

return name;

}

public int getCoreNum() {

return coreNum;

}

public int getPrice() {

return price;

}

public void setName(String name) {

this.name = name;

}

public void setCoreNum(int coreNum) {

this.coreNum = coreNum;

}

public void setPrice(int price) {

this.price = price;

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

}

5. Computer类，计算机类，实现了work、show接口，由CPU、Memory、HardDisk、Mainboard类聚合而成

属性：计算机名称name、总价格price、CPU对象cpu、内存对象memory、硬盘对象hardDIsk、主板对象mainboard

方法：

无参构造方法

带四个参数的构造方法（CPU、Memory、HardDisk、Mainboard），使用子类组件的实例作为参数传入。

各属性的get和set方法

totalprice方法：计算总价格，返回四个组件的价格之和

Work方法（override）：分别调用四个组件的work方法

Show方法（override）：分别调用四个组件的show方法

具体代码：

public class Computer implements Work, Show{

private String name;

private int price;

private CPU cpu;

private Memory memory;

private HardDisk hardDisk;

private Mainboard mainboard;

public Computer() {

//TODO Auto-generated constructor stub.

}

public Computer(String name, CPU cpu, Memory memory, HardDisk hardDisk, Mainboard mainboard) {

this.name = name;

this.cpu = cpu;

this.memory = memory;

this.hardDisk = hardDisk;

this.mainboard = mainboard;

this.price = this.totalprice();

}

public int totalprice() {

return cpu.getPrice() + memory.getPrice() + mainboard.getPrice() + hardDisk.getPrice();

}

public void show() {

System.out.println("Computer:" + this.getName());

System.out.println("Price:" + this.getPrice());

this.cpu.show();

this.memory.show();

this.hardDisk.show();

this.mainboard.show();

System.out.println("------------------");

}

public void work() {

System.out.println(this.getName() + " starts working.");

this.cpu.work();

this.memory.work();

this.hardDisk.work();

this.mainboard.work();

System.out.println("------------------");

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

}

6. ComputerStore类：主类

属性：三个computer对象

方法：主方法，用于展示和运行computer

具体代码：

public class ComputerStore {

private static Computer computer1

= new Computer("DELL", new Intel("i5", 4, 500), new Samsung("A1", 4, 100), new Seagate("S1", 512, 300), new Asus("T3", 128, 500));

private static Computer computer2

= new Computer("VSUS", new Intel("i7", 8, 1000), new Kingston("K3", 8, 120), new Seagate("S2", 512, 300), new Gigabyte("G4", 128, 450));

private static Computer computer3

= new Computer("LENOVE", new AMD("M300", 8, 900), new Samsung("A5", 8, 200), new WestDigitals("WD2", 256, 200), new Gigabyte("G7", 256, 600));

public ComputerStore() {

}

public static void main(String args[]) {

computer1.show();

computer2.show();

computer3.show();

computer1.work();

computer2.work();

computer3.work();

}

}

1. 调试分析（在实验过程中遇到的问题以及如何解决）

在实验过程中出现了一个bug。创建的三个computer对象在调用show方法时，虽然总价格计算并显示正确，但是所有的组件参数信息均为null或0.

解决方法：因为在创建Computer对象时使用的组件参数是组件子类的实例（例如：CPU cpu = new AMD（）），继承了父类中的show方法。而在组件父类中的show方法中，打印组件信息时使用的是this.属性名直接访问private属性（静态绑定于引用变量的类型），因此打印出的是父类的没有赋值的属性值，因此默认为null或0.子类中覆盖了父类的get方法（动态绑定于对象类型），调用get方法可以得到正确的子类对象的属性值。

1. 测试结果（描述输入和输出）

三台computer的输入为：

private static Computer computer1

= new Computer("DELL", new Intel("i5", 4, 500), new Samsung("A1", 4, 100), new Seagate("S1", 512, 300), new Asus("T3", 128, 500));

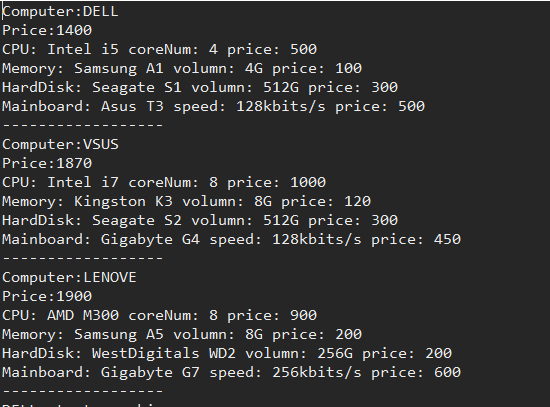
private static Computer computer2

= new Computer("VSUS", new Intel("i7", 8, 1000), new Kingston("K3", 8, 120), new Seagate("S2", 512, 300), new Gigabyte("G4", 128, 450));

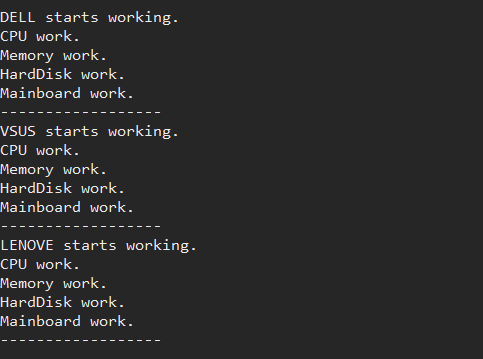
private static Computer computer3

= new Computer("LENOVE", new AMD("M300", 8, 900), new Samsung("A5", 8, 200), new WestDigitals("WD2", 256, 200), new Gigabyte("G7", 256, 600));

Show方法的输出为：



Work方法的输出为：



1. 总结

本实验的目的主要是为了练习面向对象编程，总体实现难度不大。但是想要更好地实现高内聚低耦合的优化目标也并不容易。可以使用反射机制来优化代码。