**JAVA编程进阶上机报告**

****

第一次上机作业

**学 院\_智能与计算学部\_**

**专 业\_\_ 软件工程\_\_\_\_\_**

**姓 名\_\_\_ 李济孚\_\_ \_ \_**

**学 号\_\_ 3018216260\_\_\_**

**年 级\_\_\_\_ 2018\_\_\_\_\_\_**

**班 级\_\_\_\_\_\_ 5\_\_\_\_\_\_\_\_**

# 一、实验要求

**Lab 1：计算机销售系统的设计**

1. 需求描述：

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 |

每个组件都有自己的工作方式，简单起见，每个组件的工作内容为打印“组件名+work”。

1. 实现功能：

具体要求：

1. 针对每个组件的每个品牌，设计一个类，并画成整体的类图
2. 设计计算机类（Computer.java），由上述四类组件组装而成，包括计算机的名称、计算机的描述（包括各个组件名）以及总价格等
3. 设计计算机销售主类（ComputerStore.java），包括3个由不同组件组装在一起的计算机实例，可实现计算机商品一览表，可展示每台计算机的描述、价格、工作等。

设计时基于抽象类和接口，要尽可能的实现高内聚、低耦合

# 二、源代码

**CPU：**

**public** **interface** CPU {

**public** String ***name*** = "CPU";

**public** **int** ***coreNum*** = 0;

**public** **int** ***price*** = 0;

**default** **void** CPU\_work()

{

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

}

}

# 三、运行结果

**public** **class** WestDigitals **implements** disk{

**public** String name = "WestDigitals";

**public** **int** volume = 950;

**public** **int** price = 430;

}

**public** **class** Seagate **implements** disk{

**public** String name = "Seagate";

**public** **int** volume = 900;

**public** **int** price = 400;

# }

**Disk**

**public** **interface** disk {

**public** String ***name*** = "disk";

**public** **int** ***speed*** = 0;

**public** **int** ***price*** = 0;

**default** **void** disk\_work()

{

System.***out***.println("disk work");

}

}

Memory

**public** **interface** memory {

**public** String ***name*** = "memory";

**public** **int** ***volume*** = 0;

**public** **int** ***price*** = 0;

**default** **void** mem\_work()

{

System.***out***.println("memory work");

}

}

Mainboard

**public** **interface** mainboard {

**public** String ***name*** = "mainboard";

**public** **int** ***speed*** = 0;

**public** **int** ***price*** = 0;

**default** **void** board\_work()

{

System.***out***.println("mainborad work");

}

}

**public** **class** Intel **implements** CPU{

**public** String name = "Intel";

**public** **int** coreNum = 16;

**public** **int** price = 2000;

# }

**public** **class** AMD **implements** CPU{

**public** String name = "AMD";

**public** **int** coreNum = 16;

**public** **int** price = 1600;

}

**public** **class** Samsung **implements** memory{

**public** String name = "Samsung";

**public** **int** volume = 90;

**public** **int** price = 40;

}

**public** **class** Kingston **implements** memory{

**public** String name = "Kingston";

**public** **int** volume = 80;

**public** **int** price = 30;

}

**public** **class** Asus **implements** mainboard{

**public** String name = "Asus";

**public** **int** speed = 100;

**public** **int** price = 730;

}

**public** **class** Gigabyte **implements** mainboard{

**public** String name = "Gigabyte";

**public** **int** speed = 110;

**public** **int** price = 750;

}

**public** **class** Computer **implements** CPU, mainboard, memory, disk{

**private** String name = "";

**private** String cpuname = "";

**private** String diskname = "";

**private** String memoryname = "";

**private** String mainboardname = "";

**private** **int** totalprice = 0;

**public** Computer(String name, String CPU, String mainboard, String memory, String disk)

{

**this**.name = name;

**if**(CPU.equals("Intel"))

{

Intel cpu = **new** Intel();

**this**.cpuname = cpu.name;

**this**.totalprice += cpu.price;

}

**else**

{

AMD cpu = **new** AMD();

**this**.cpuname = cpu.name;

**this**.totalprice += cpu.price;

}

**if**(mainboard.equals("Asus"))

{

Asus mb = **new** Asus();

**this**.mainboardname = mb.name;

**this**.totalprice += mb.price;

}

**else**

{

Gigabyte gb = **new** Gigabyte();

**this**.mainboardname = gb.name;

**this**.totalprice += gb.price;

}

**if**(disk.equals("Intel"))

{

WestDigitals di = **new** WestDigitals();

**this**.diskname = di.name;

**this**.totalprice += di.price;

}

**else**

{

Seagate di = **new** Seagate();

**this**.diskname = di.name;

**this**.totalprice += di.price;

}

**if**(memory.equals("Kingston"))

{

Kingston mem = **new** Kingston();

**this**.memoryname = mem.name;

**this**.totalprice += mem.price;

}

**else**

{

Samsung mem = **new** Samsung();

**this**.memoryname = mem.name;

**this**.totalprice += mem.price;

}

}

**public** **void** showinfo()

{

CPU\_work();

board\_work();

disk\_work();

mem\_work();

System.***out***.println("This computer named: " + name);

System.***out***.println("It contains: CPU-" + cpuname);

System.***out***.println(" disk-" + diskname);

System.***out***.println(" memory-" + memoryname);

System.***out***.println(" mainborad-" + mainboardname);

System.***out***.println("And the price is $" + totalprice);

System.***out***.println();

}

}

**public** **class** ComputerStore {

**public** **static** **void** main(String[] args)

{

Computer a = **new** Computer("A", "Intel", "Asus", "Samsung", "Seagate");

Computer b = **new** Computer("B", "AMD", "Gigabyte", "Samsung", "WestDigital");

Computer c = **new** Computer("C", "Intel", "Gigabyte", "Kingston", "Seagate");

a.showinfo();

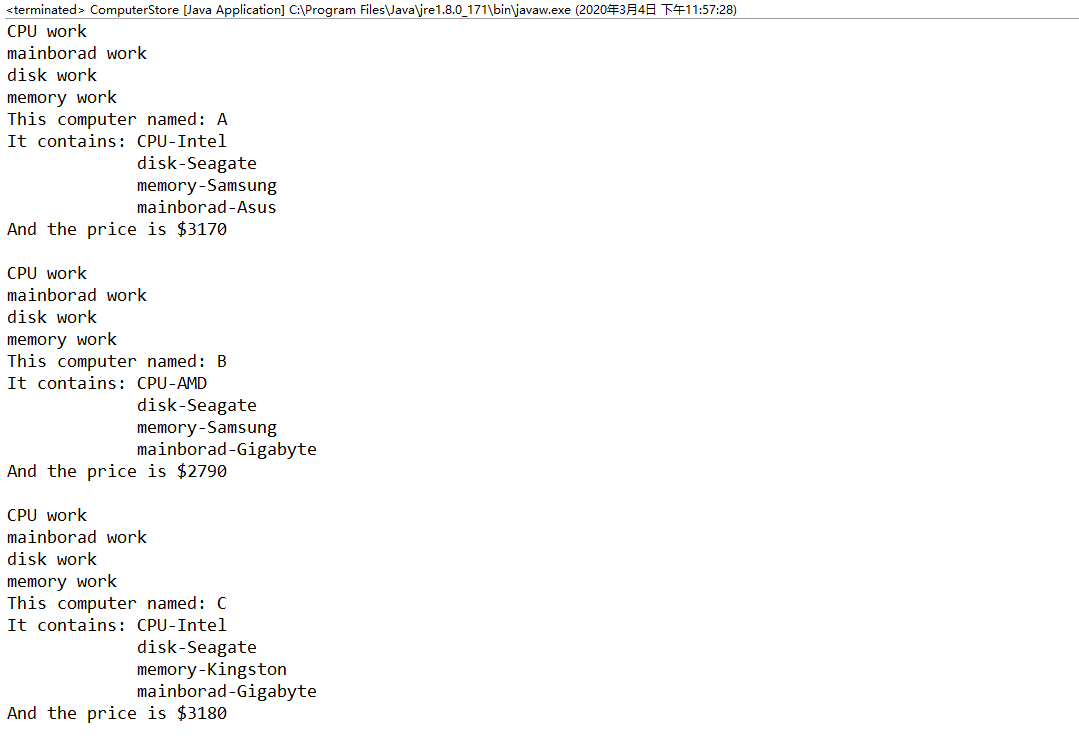
b.showinfo();

c.showinfo();

}

}

# 三、运行结果



# 

# 四、设计思路和类图

设计思路：由于每个组件都可以作为对应品牌的抽象，所以看可以将四个组件分别设置为四个接口，而对应的品牌则是实现对应接口的类，这样就可以保证品牌之间又有彼此之间的共性也有特性。而对于工作内容这个函数可以在接口中设置default方法，这样每个品牌就都会继承这个方法。对于computer类来说，因为要在computerStore类里实例化出3个带有不同组件的电脑对象，所以对于computer类的设计就要采用组合+代理的办法，以此来实现用一个computer类来实例化出不同品牌的组件的电脑实例。最后，computerStore作为主类所以带有main函数，而暂时不需要其他方法和成员变量。

下面是类图：