**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\_\_郑开\_\_\_\_\_\_\_\_\_\_**

**学 号\_\_3018216157\_\_**

**年 级\_\_18级\_\_\_\_\_\_\_\_\_**

**班 级\_\_3 班\_\_\_\_\_\_\_\_\_**

1. 实验要求

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”。

2. 实现功能：

具体要求：

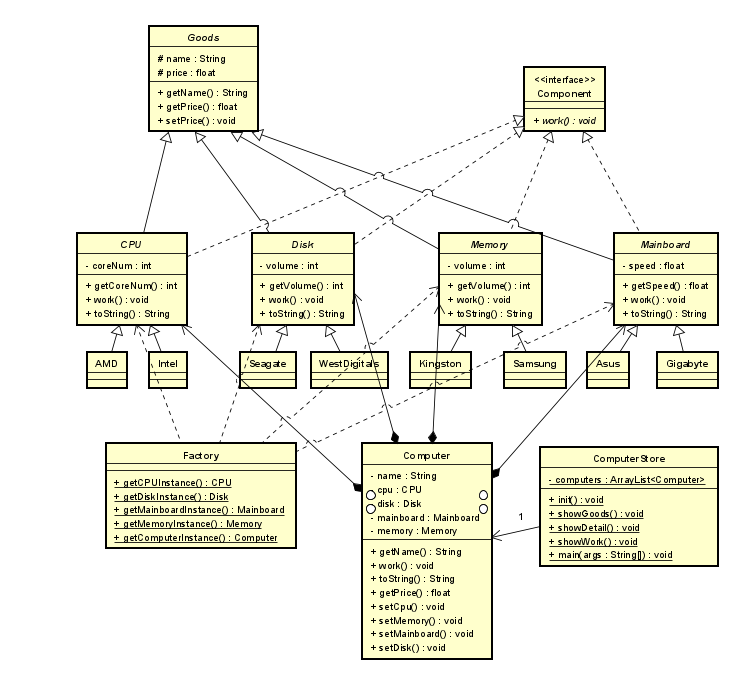
1) 针对每个组件的每个品牌，设计一个类，并画成整体的类图

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。

4) 设计时基于抽象类和接口，要尽可能的实现高内聚、低耦合。

1. 设计思路



首先构建组件。因为CPU、硬盘、内存、主板都可抽象为组件和商品，所以构建了抽象类商品Goods以及接口组件Component。Goods实现一个商品应有的属性功能，商品名称name以及商品价格price。而组件Component提供组件应有的工作函数，具体的工作方式由子类实现。四大组件都继承了Goods并实现了Component，并且将这四大组件设置为抽象类，具体的组件品牌只要继承组件抽象类即可。这样的话，如果有新型的组件，只需要再加一个类即可，并且组件自身的特性可以体现在自身的函数。

关于Computer类。计算机由四个组件构成，所以可以通过组合的方式，让Computer类持有四个类的对象，并设置自身的name属性，实现自身描述的函数和返回价格的函数即可。并且Computer可以更换组件，查看组件信息。

关于ComputerStore类。该类持有一个Computer的列表，在init函数中固定初始化了三台电脑。使用showGoods()函数展示商品一览，showDetail()函数展示电脑详细信息，showWork()函数展示电脑工作信息。

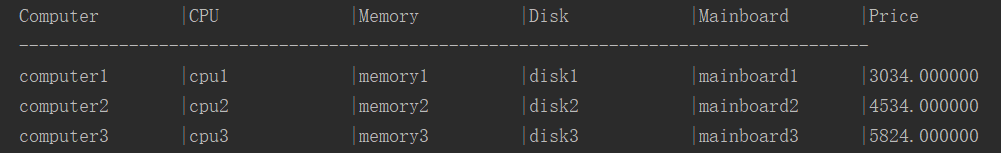
关于Factory类。该类提供组件以及电脑的创建函数，通过反射产生实例。

1. 源代码

package component;   
   
public interface Component {   
 void work();   
}   
   
package component;   
   
public abstract class Goods {   
 protected String name;   
 protected float price;   
   
 public Goods(String name, float price) {   
 this.name = name;   
 if (price <= 0) {   
 this.price = 0;   
 } else {   
 this.price = price;   
 }   
 }   
   
 public String getName() {   
 return name;   
 }   
   
 public float getPrice() {   
 return price;   
 }   
   
 public void setPrice(float price) {   
 if (price <= 0) {   
 this.price = 0;   
 } else {   
 this.price = price;   
 }   
 }   
}   
   
package component;   
   
   
public abstract class CPU extends Goods implements Component {   
 protected int coreNum;   
   
 public CPU(String name, float price, int coreNum) {   
 super(name, price);   
 this.coreNum = coreNum;   
 }   
   
 @Override   
 public String toString() {   
 return "CPU{" +   
 "coreNum=" + coreNum + " cores"+   
 ", name='" + name + '\'' +   
 ", price=" + price + " RMB"+   
 '}';   
 }   
   
 public int getCoreNum() {   
 return coreNum;   
 }   
   
 @Override   
 public void work() {   
 System.out.println(name + " work");   
 }   
}   
   
package component;   
   
public abstract class Disk extends Goods implements Component {   
 protected int volume;   
   
 public Disk(String name, float price, int volume) {   
 super(name, price);   
 this.volume = volume;   
 }   
   
 @Override   
 public String toString() {   
 return "Disk{" +   
 "volume=" + volume +" GB"+   
 ", name='" + name + '\'' +   
 ", price=" + price + "RMB"+   
 '}';   
 }   
   
 public int getVolume() {   
 return volume;   
 }   
   
 @Override   
 public void work() {   
 System.out.println(name + " work");   
 }   
}   
   
package component;   
   
public abstract class Mainboard extends Goods implements Component {   
 protected float speed;   
   
 public Mainboard(String name, float price, float speed) {   
 super(name, price);   
 this.speed = speed;   
 }   
   
 @Override   
 public String toString() {   
 return "Mainboard{" +   
 "speed=" + speed +" GHz"+   
 ", name='" + name + '\'' +   
 ", price=" + price + " RMB" +   
 '}';   
 }   
   
 public float getSpeed() {   
 return speed;   
 }   
   
 @Override   
 public void work() {   
 System.out.println(name + " work");   
 }   
}   
   
package component;   
   
   
public abstract class Memory extends Goods implements Component {   
 protected int volume;   
   
 public Memory(String name, float price, int volume) {   
 super(name, price);   
 this.volume = volume;   
 }   
   
 @Override   
 public String toString() {   
 return "Memory{" +   
 "volume=" + volume +" GB"+   
 ", name='" + name + '\'' +   
 ", price=" + price + " RMB" +   
 '}';   
 }   
   
 public int getVolume() {   
 return volume;   
 }   
   
 @Override   
 public void work() {   
 System.out.println(name + " work");   
 }   
}

package component.cpu;   
   
import component.CPU;   
   
public class AMD extends CPU {   
 public AMD(String name, float price, int coreNum) {   
 super(name, price, coreNum);   
 }   
}   
   
package component.cpu;   
   
import component.CPU;   
   
public class Intel extends CPU {   
 public Intel(String name, float price, int coreNum) {   
 super(name, price, coreNum);   
 }   
}   
   
package component.disk;   
   
import component.Disk;   
   
public class Seagate extends Disk {   
 public Seagate(String name, float price, int volume) {   
 super(name, price, volume);   
 }   
}   
   
package component.disk;   
   
import component.Disk;   
   
public class WestDigitals extends Disk {   
 public WestDigitals(String name, float price, int volume) {   
 super(name, price, volume);   
 }   
}   
   
package component.mainboard;   
   
import component.Mainboard;   
   
public class Asus extends Mainboard {   
 public Asus(String name, float price, float speed) {   
 super(name, price, speed);   
 }   
}   
   
package component.mainboard;   
   
import component.Mainboard;   
   
public class Gigabyte extends Mainboard {   
 public Gigabyte(String name, float price, float speed) {   
 super(name, price, speed);   
 }   
}   
   
package component.memory;   
   
import component.Memory;   
   
public class Kingston extends Memory {   
 public Kingston(String name, float price, int volume) {   
 super(name, price, volume);   
 }   
}   
   
package component.memory;   
   
import component.Memory;   
   
public class Samsung extends Memory {   
 public Samsung(String name, float price, int volume) {   
 super(name, price, volume);   
 }   
}   
   
  
package computer;   
   
import component.\*;   
   
public class Computer {   
 private String name;   
 private CPU cpu;   
 private Disk disk;   
 private Mainboard mainboard;   
 private Memory memory;   
 public Computer(String name,CPU cpu, Disk disk, Mainboard mainboard, Memory memory){   
 this.name = name;   
 this.cpu = cpu;   
 this.disk = disk;   
 this.mainboard = mainboard;   
 this.memory = memory;   
 }   
 public String getCPUName(){   
 return cpu.getName();   
 }   
 public String getDiskName(){   
 return disk.getName();   
 }   
 public String getMainboardName(){   
 return mainboard.getName();   
 }   
 public String getMemoryName(){   
 return memory.getName();   
 }   
   
 public void setCpu(CPU cpu) {   
 this.cpu = cpu;   
 }   
   
 public void setDisk(Disk disk) {   
 this.disk = disk;   
 }   
   
 public void setMainboard(Mainboard mainboard) {   
 this.mainboard = mainboard;   
 }   
   
 public void setMemory(Memory memory) {   
 this.memory = memory;   
 }   
   
 public void work(){   
 cpu.work();   
 disk.work();   
 mainboard.work();   
 memory.work();   
 }   
 @Override   
 public String toString() {   
 return "Computer{" +   
 "name='" + name + '\'' +   
 ", cpu=" + cpu +   
 ", disk=" + disk +   
 ", mainboard=" + mainboard +   
 ", memory=" + memory +   
 '}';   
 }   
   
 public String getName() {   
 return name;   
 }   
 public float getPrice(){   
 float price = 0;   
 if(cpu != null){   
 price += cpu.getPrice();   
 }   
 if(disk != null){   
 price += disk.getPrice();   
 }   
 if(mainboard != null){   
 price += mainboard.getPrice();   
 }   
 if(memory != null){   
 price += memory.getPrice();   
 }   
 return price;   
 }   
}   
   
package factory;   
   
   
import component.CPU;   
import component.Disk;   
import component.Mainboard;   
import component.Memory;   
import computer.Computer;   
import org.jetbrains.annotations.NotNull;   
   
import java.lang.reflect.InvocationTargetException;   
   
public class Factory {   
 @NotNull   
 public static CPU getCPUInstance(String type, String name, float price, int coreNum) throws ClassNotFoundException, NoSuchMethodException, IllegalAccessException, InvocationTargetException, InstantiationException {   
 return (CPU) Class.forName("component.cpu." + type)   
 .getConstructor(String.class, float.class, int.class)   
 .newInstance(name, price, coreNum);   
 }   
   
 @NotNull   
 public static Disk getDiskInstance(String type, String name, float price, int volume) throws ClassNotFoundException, NoSuchMethodException, IllegalAccessException, InvocationTargetException, InstantiationException {   
 return (Disk) Class.forName("component.disk." + type)   
 .getConstructor(String.class, float.class, int.class)   
 .newInstance(name, price, volume);   
 }   
   
 @NotNull   
 public static Mainboard getMainboardInstance(String type, String name, float price, float speed) throws ClassNotFoundException, NoSuchMethodException, IllegalAccessException, InvocationTargetException, InstantiationException {   
 return (Mainboard) Class.forName("component.mainboard." + type)   
 .getConstructor(String.class, float.class, float.class)   
 .newInstance(name, price, speed);   
 }   
   
 @NotNull   
 public static Memory getMemoryInstance(String type, String name, float price, int volume) throws ClassNotFoundException, NoSuchMethodException, IllegalAccessException, InvocationTargetException, InstantiationException {   
 return (Memory) Class.forName("component.memory." + type)   
 .getConstructor(String.class, float.class, int.class)   
 .newInstance(name, price, volume);   
 }   
   
 @NotNull   
 public static Computer getComputerInstance(String name, CPU cpu, Disk disk, Mainboard mainboard, Memory memory) throws ClassNotFoundException, NoSuchMethodException, IllegalAccessException, InvocationTargetException, InstantiationException {   
 return (Computer) Class.forName("computer.Computer")   
 .getConstructor(String.class, CPU.class,Disk.class,Mainboard.class,Memory.class)   
 .newInstance(name, cpu, disk,mainboard,memory);   
 }   
}   
   
package main;   
   
import component.CPU;   
import component.Disk;   
import component.Mainboard;   
import component.Memory;   
import computer.Computer;   
import factory.Factory;   
   
import java.lang.reflect.InvocationTargetException;   
import java.util.ArrayList;   
import java.util.List;   
   
public class ComputerStore {   
 private static List<Computer> computers = new ArrayList<>();   
   
 public static void main(String[] args) throws ClassNotFoundException, NoSuchMethodException, InvocationTargetException, InstantiationException, IllegalAccessException {   
 init();   
 showGoods();   
 showDetail();   
 showWork();   
   
 }   
   
 private static void init() throws ClassNotFoundException, NoSuchMethodException, InvocationTargetException, InstantiationException, IllegalAccessException {   
 CPU cpu1 = Factory.getCPUInstance("AMD", "cpu1", 1000, 4);   
 CPU cpu2 = Factory.getCPUInstance("AMD", "cpu2", 1200, 8);   
 CPU cpu3 = Factory.getCPUInstance("Intel", "cpu3", 1300, 8);   
   
 Disk disk1 = Factory.getDiskInstance("Seagate", "disk1", 300, 512);   
 Disk disk2 = Factory.getDiskInstance("Seagate", "disk2", 400, 1024);   
 Disk disk3 = Factory.getDiskInstance("WestDigitals", "disk3", 500, 1024);   
   
 Mainboard mainboard1 = Factory.getMainboardInstance("Asus", "mainboard1", 1234, 2.34f);   
 Mainboard mainboard2 = Factory.getMainboardInstance("Asus", "mainboard2", 2134, 3.45f);   
 Mainboard mainboard3 = Factory.getMainboardInstance("Gigabyte", "mainboard3", 3124, 3.45f);   
   
 Memory memory1 = Factory.getMemoryInstance("Kingston", "memory1", 500, 4);   
 Memory memory2 = Factory.getMemoryInstance("Kingston", "memory2", 800, 8);   
 Memory memory3 = Factory.getMemoryInstance("Samsung", "memory3", 900, 8);   
   
 computers.add(Factory.getComputerInstance("computer1", cpu1, disk1, mainboard1, memory1));   
 computers.add(Factory.getComputerInstance("computer2", cpu2, disk2, mainboard2, memory2));   
 computers.add(Factory.getComputerInstance("computer3", cpu3, disk3, mainboard3, memory3));   
 }   
   
 public static void showGoods() {   
 System.out.format("%-15s |%-15s |%-15s |%-15s |%-15s |%-15s\n", "Computer", "CPU", "Memory", "Disk", "Mainboard", "Price");   
 for (int i = 0; i < 17 \* 5; i++) {   
 System.out.print("-");   
 }   
 System.out.println();   
 for (Computer com : computers) {   
 System.out.format("%-15s |%-15s |%-15s |%-15s |%-15s |%-15f\n", com.getName(), com.getCPUName(), com.getMemoryName(), com.getDiskName(), com.getMainboardName(), com.getPrice());   
 }   
 }   
   
 public static void showDetail() {   
 for (Computer com : computers) {   
 System.out.println(com);   
 }   
 }   
   
 public static void showWork() {   
 for (int i = 0; i < computers.size(); i++) {   
 System.out.println("=========computer"+i+" work=========");   
 computers.get(i).work();   
 }   
 }   
}

1. 实验结果
2. 一览信息价格



1. 详细信息

Computer{name='computer1', cpu=CPU{coreNum=4 cores, name='cpu1', price=1000.0 RMB}, disk=Disk{volume=512 GB, name='disk1', price=300.0RMB}, mainboard=Mainboard{speed=2.34 GHz, name='mainboard1', price=1234.0 RMB}, memory=Memory{volume=4 GB, name='memory1', price=500.0 RMB}}

Computer{name='computer2', cpu=CPU{coreNum=8 cores, name='cpu2', price=1200.0 RMB}, disk=Disk{volume=1024 GB, name='disk2', price=400.0RMB}, mainboard=Mainboard{speed=3.45 GHz, name='mainboard2', price=2134.0 RMB}, memory=Memory{volume=8 GB, name='memory2', price=800.0 RMB}}

Computer{name='computer3', cpu=CPU{coreNum=8 cores, name='cpu3', price=1300.0 RMB}, disk=Disk{volume=1024 GB, name='disk3', price=500.0RMB}, mainboard=Mainboard{speed=3.45 GHz, name='mainboard3', price=3124.0 RMB}, memory=Memory{volume=8 GB, name='memory3', price=900.0 RMB}}

1. 工作情况

=========computer0 work=========

cpu1 work

disk1 work

mainboard1 work

memory1 work

=========computer1 work=========

cpu2 work

disk2 work

mainboard2 work

memory2 work

=========computer2 work=========

cpu3 work

disk3 work

mainboard3 work

memory3 work