天津大学

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　　陈沛圻
2019年 3 月 7 日

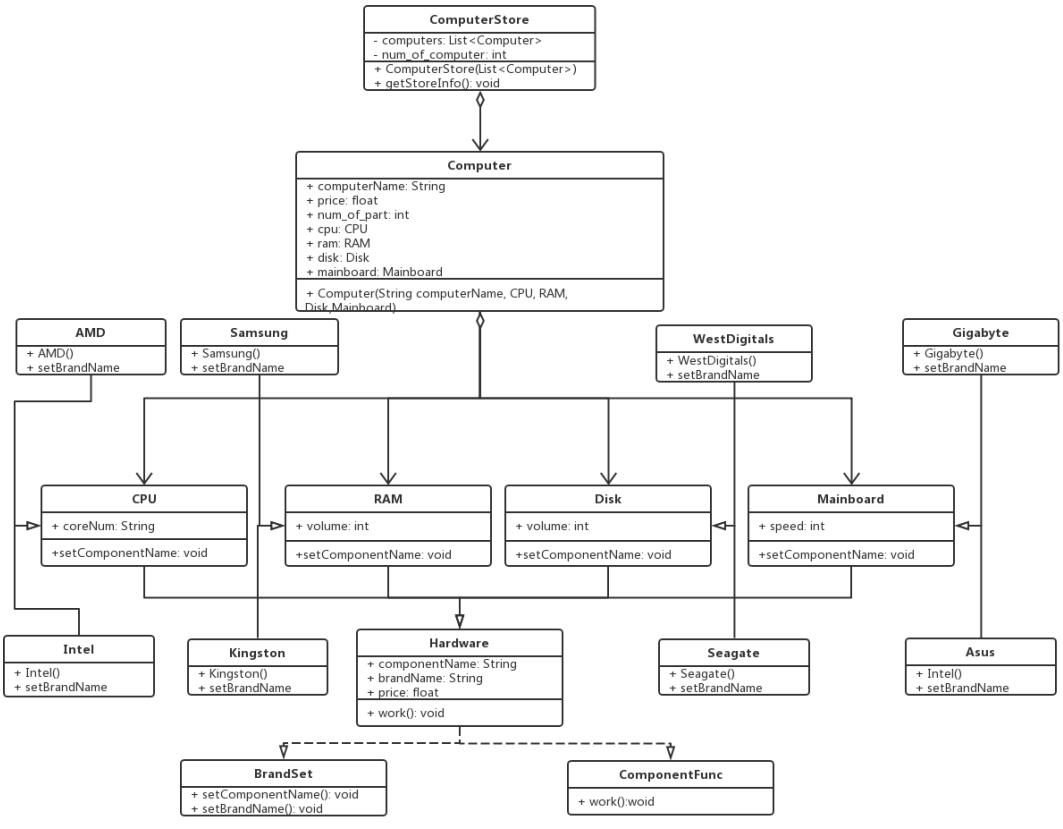
JAVA进阶编程第一次实验报告

一. 需求分析（描述具体需求）

1) 每个组件有若干牌子，针对每个组件的每个品牌，设计一个类。
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。

二. 概要设计（简单描述设计思路，配合UML图）

CPU，AMD，Disk，Mainboard是三个抽象类，继承自抽象类Hardware；
抽象类Hardware实现两个接口BrandSet和ComponentFunc；
Computer类由CPU，AMD，Disk，Mainboard中的至少三个聚合而成；
ComputerStore由类Computer聚合而成，有一个Computer类的List；
UML类图见下：



三. 详细设计（详细描述具体如何实现，附代码及说明）

两个接口：

BrandSet实现每个部件设置部件名字和品牌名字

```
public interface BrandSet {
    void setComponentName();
    void setBrandName();
}
```

ComponentFunc将各部件公共的work方法抽出

```
public interface ComponentFunc {
    void work();
}
```

抽象类：

Hardware：抽象类，有四种零部件的公共属性，实现了work方法

```
abstract public class Hardware implements ComponentFunc,BrandSet{
    String componentName;
    String brandName;
    float price;
    @Override
    public void work(){
        System.out.println(this.componentName+"work");
    }
}
```

CPU，AMD，Disk，Mainboard：都为抽象类，继承类Hardware。除了Hardware中的公共属性，有各自的特殊属性，并各自实现接口BrandSet中setComponentName的方法，以CPU为例

```
abstract public class CPU extends Hardware{
        String coreNum;
        CPU(){
            this.setComponentName();
        }
    @Override
    public void setComponentName(){
        this.componentName = "CPU";
    }
```

品牌类：

每个品牌为一个类，继承相应的部件名字的类，（如类Intel继承类CPU），并各自实现接口BrandSet中setBrandName的方法，以继承CPU类的Intel类为例

```
class Intel extends CPU{
    Intel(){
        this.setBrandName();
    }
    @Override
    public void setBrandName(){
        this.brandName = "Intel";
    }
}
```

四. 调试分析（在实验过程中遇到的问题以及如何解决）

逻辑问题：

初始的类之间逻辑混乱。设置了类Brand作为各个品牌的父类，各个部件品牌又为一类。如类CPUBrand继承类Brand，但同时又设置了类CPU作为具体部件。

觉得为每个具体品牌设置一个类太过于繁琐，想要用枚举来实现品牌与部件名字的联系，但是只能用具体的名字才能调用查看部件名字，不能动态的查看品牌名与部件名字的关系。

开始使用数组作为store中computer的存储结构，后考虑到Computer可以动态增删，于是改用list作为容器。

没有使用接口，没有将公共方法抽离出来。

后来删去一些类，改变逻辑架构，如上述。

代码运行问题：

输出品牌名字的时候输出为指向品牌类的指针，输出时忘记加上属性名。

Iterator遍历store中computer的list总是跳过一些computer，后发现是在输出时使用过it.next(),输出后又写了一遍，故一次while循环跳过了一个computer。

Computer由CPU，AMD，Disk，Mainboard中的至少三个聚合而成，考虑到三个类都是抽象类无法实例化，初始将四个类初始为null，在构造函数传参中进行赋值。但在store遍历时，只传三个部件的Computer就会出现NullPointerException。后加写了四个三部件的构造函数，但由于部件类如类CPU是抽象类，还需在三个组件的构造方法中重写setBrandName方法，由于此时Computer没有CPU，故暂将brandName设置为空字符。

未解决：

想要实现从键盘读取电脑及零部件等数据，建立Computerstore，但是创建部

件类的时候遇到了Object无法转换为CPU类的问题。
函数如下：

```
public static Object getclass(String className)
{
    Class clz = Class.forName(className);
    Constructor constructor = clz.getDeclaredConstructor(String.class);
    Object instance = constructor.newInstance();
    return instance;
}
```

main中的调用代码如下：

```
List<Computer> comList = new ArrayList<>();
ComputerStore store = new ComputerStore(comList);

Scanner sc = new Scanner(System.in);
System.out.println("Input Computer numbers:");
int num = sc.nextInt();

for(int i=0; i< num; i++){

    System.out.println("Input Computername:");
    String computerName = sc.nextLine();
    System.out.println("This computer has 4 or 3 parts: ");
    int part = sc.nextInt();
    System.out.println("Input the brand of every part in this order: CPU
RAM Disk Mainboard:\n");

    String p1 = sc.nextLine();
    String p2 = sc.nextLine();
    String p3 = sc.nextLine();

    Object o1,o2,o3;

    o1 = getclass(p1);
    o2 = getclass(p2);
    o3 = getclass(p3);

    if(part==4){
        String p4 = sc.nextLine();
        Object o4;
        o4  =getclass(p4);
        comList.add(new Computer(computerName,o1,o2,o3,o4));
    }else{
        comList.add(new Computer(computerName,o1,o2,o3));
    }
}
```

而且一次只能创建一个store，逻辑问题未解决。
根据所得字符串动态创建类的过程有许多需要抛出的异常，来保证代码不出现warning，较为复杂，但IDEA可以帮你全部做好。
部分应该设为ptivate的数据没有考虑周全。

五. 测试结果（描述输入和输出）

创建各组件品牌类实例，创建若干Computer实例，创建ComputerStore实例

```
CPU intel = new Intel();
RAM samsung = new Samsung();
Disk seagate = new Seagate();
Mainboard asus = new Asus();

Computer cm1 = new Computer("com1",intel,samsung,seagate,asus);
Computer cm2 = new Computer("com2",intel,seagate,asus);
Computer cm3 = new Computer("com3",intel,samsung,seagate,asus);
Computer cm4 = new Computer("com4",intel,samsung,seagate,asus);

List<Computer> comList = new ArrayList<>();
comList.add(cm1);
comList.add(cm2);
comList.add(cm3);
comList.add(cm4);
ComputerStore store1 = new ComputerStore(comList);
```

测试各组件work功能，测试store中的各个computer信息

```
intel.work();
samsung.work();
seagate.work();
asus.work();

store1.getStoreInfo();
```

输出如下：

```
CPUwork
RAMwork
diskwork
mainboardwork
ComputerName is :com1;
Price is: 0.0;
Has Intel Samsung Seagate Asus;


ComputerName is :com2;
Price is: 0.0;
Has Intel null Seagate Asus;


ComputerName is :com3;
```

```
Price is: 0.0;
Has Intel Samsung Seagate Asus;


ComputerName is :com4;
Price is: 0.0;
Has Intel Samsung Seagate Asus;



Process finished with exit code 0
```

六. 总结

代码实现部分写的较简单，花了很多时间缕清类与类关系，后来重写了逻辑结构；
在属性设置上，尽力保证同样的属性不会出现在两个地方。
做了一些工作，使代码语义性更强：
专门将相同的函数提出，写成接口，并写了两个接口；
组件类如CPU类的构造函数调用了公共的setComponentName函数，而不是直接String name=“CPU”，因为组件四个类的构造方法类似，可以抽象出一个统一的方法。