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	张世琦
学　号	3018216185
年　级	大二
班　级	4

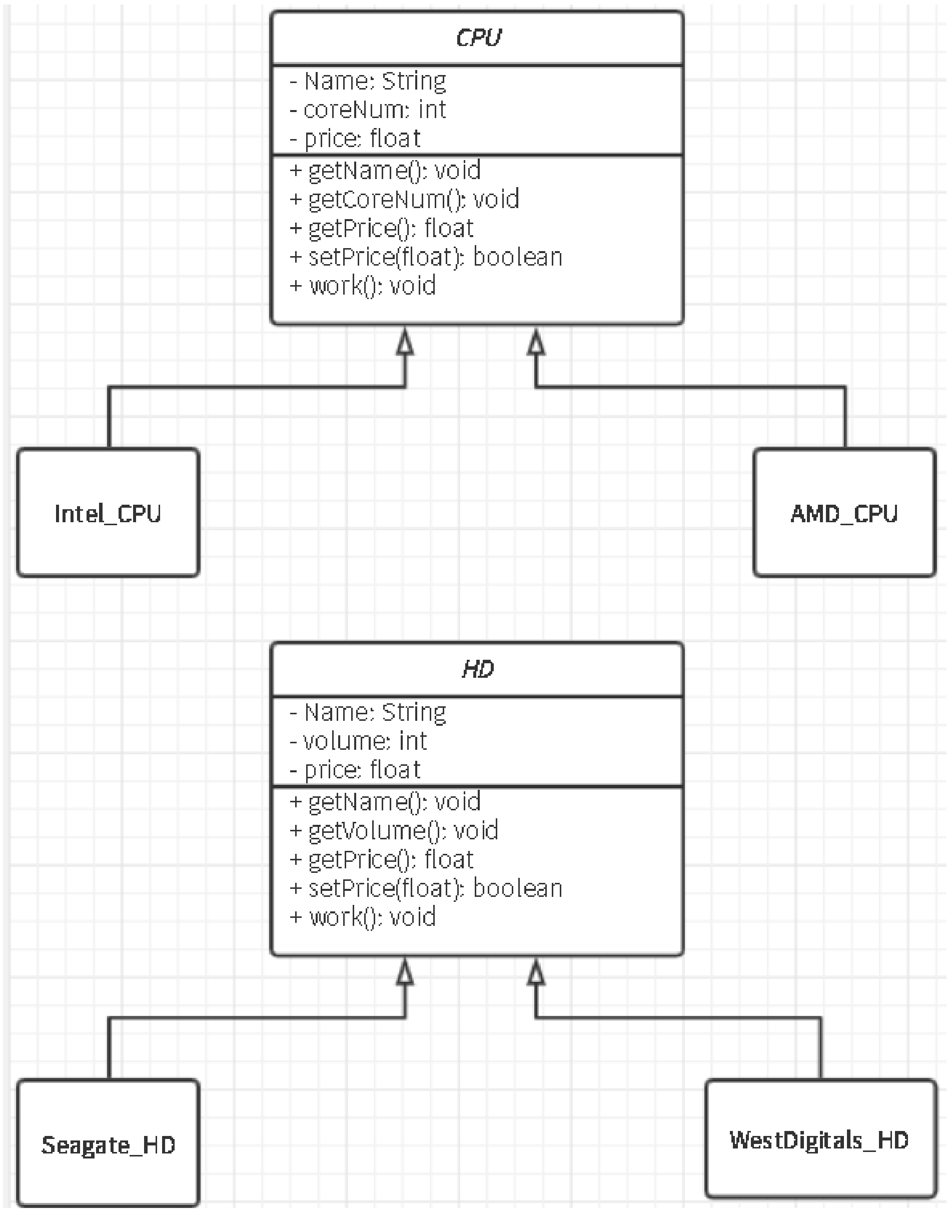
一、实验要求

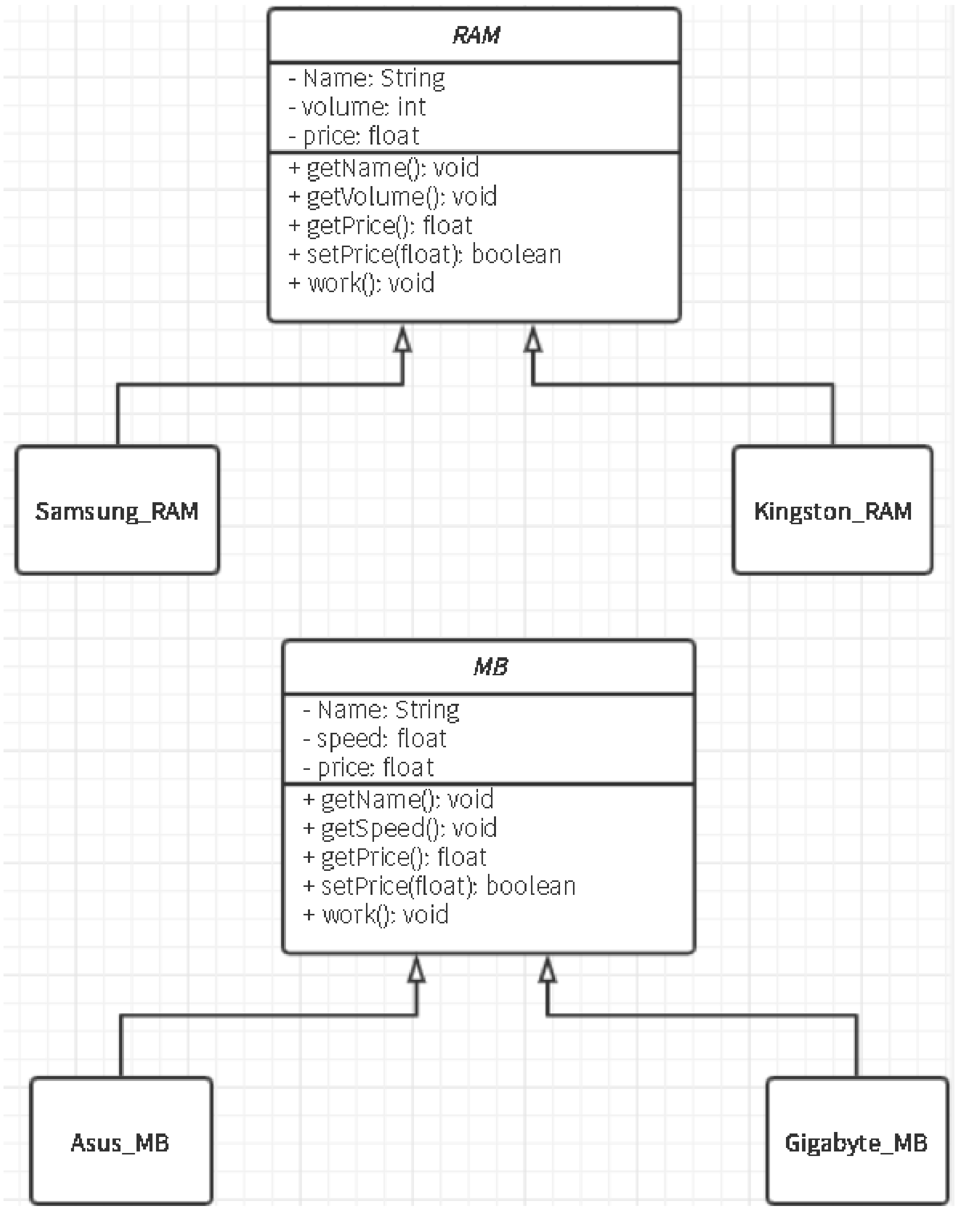
1. 设计组件类图

对于每个组件构造一个抽象类。抽象类中，所有的方法均已实现，子类直接继承即可。若各品牌有属性、功能的添加，直接在子类中进行添加即可，且不会影响其他品牌。秉持了“低耦合”的原则

现以 **CPU** 为例说明数据类型及方法的选取：

- 为了数据安全，需将数据域的所有成员设为 `private`
- 考虑到在出厂时，每个组件的核心数 `coreNum` 已经确定为常值，故不提供 `set` 方法。`name` 同理
- 考虑到现实生活中各组件的价格，使用 `float` 作为数据类型足矣



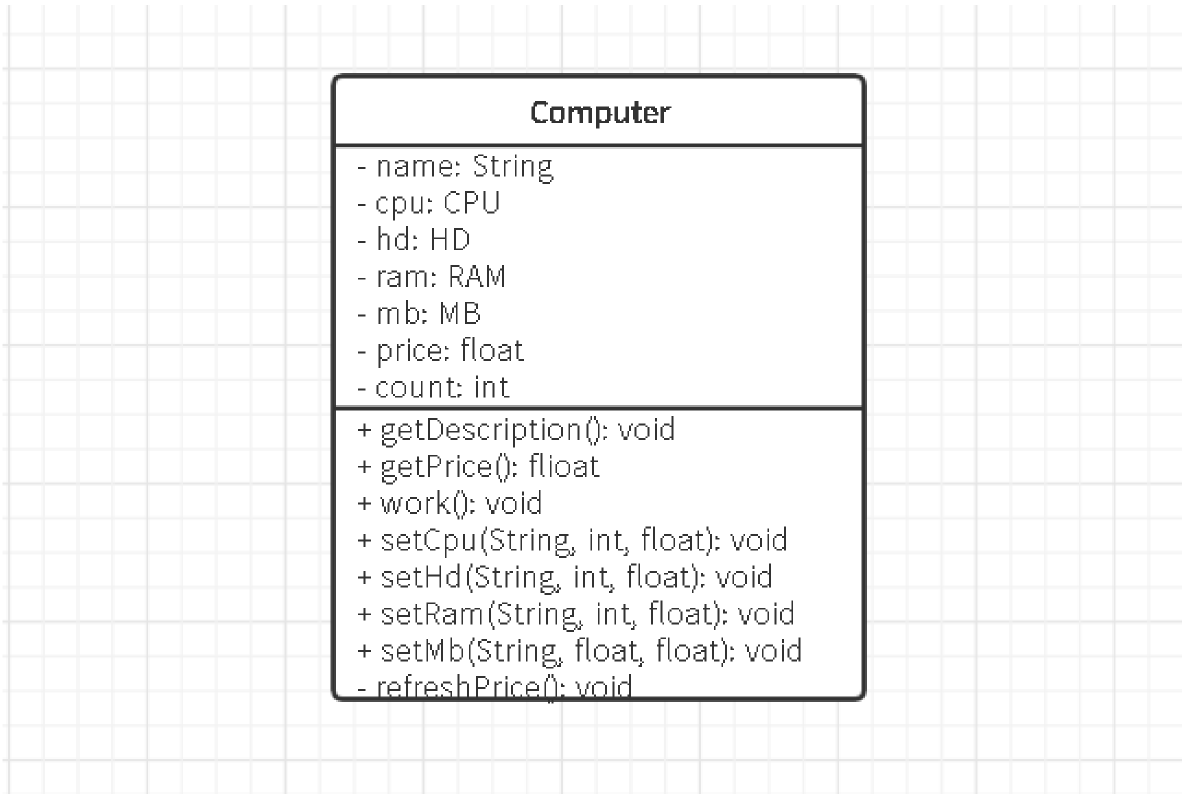


2. 设计计算机类(`Computer`)

该类的每一个对象均是一个拥有4个组件的计算机，拥有默认的构造方法。若客户有任何需求，均可以调用 `set` 方法，对组件进行改装，改装后会自动刷新价格。

该类还有一个静态的成员变量 `count`，用于记录现有创建的所有计算机。该值将作为每个计算机名称 `name` 的后缀

- 采用如下设计，该类是由上述定义的类组建而成的新类，体现了“**组合**”的思想
- 而组件的类型选取其父类，即抽象类，从而可以兼容各个品牌的组件。体现了“**多态**”的思想

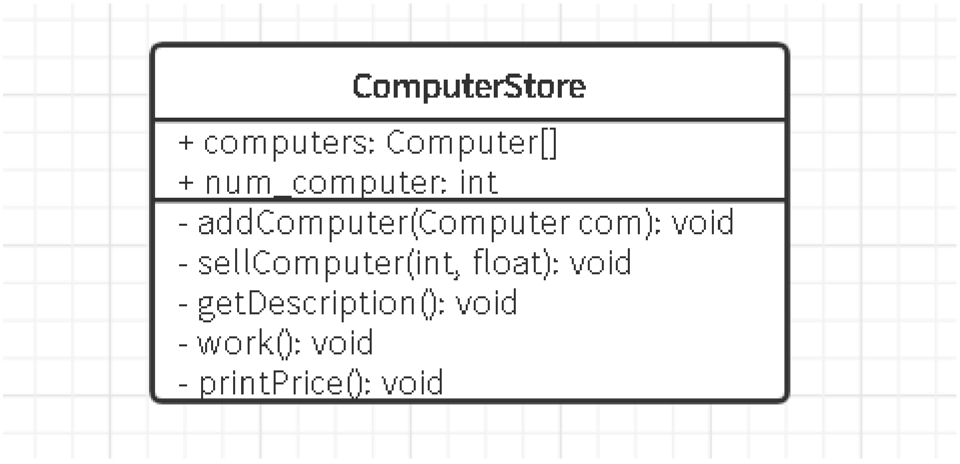


3. 设计计算机销售类 (`ComputerStore`)

　在默认情况下，该类实例化后，对象数组 `computers` 为空数组。需要调用 `addComputer()` 添加需要销售的计算机实例。

　`num_computer` 用来记录当前对象可供销售的 计算机数量，仅能由方法 `addComputer()` 与 `sellComputer()` 修改。

　调用 `getDescription()` 后，对象数组中的每个计算机实例均会给出自己的组件参数描述



二、源代码

由于考虑到"低耦合"，组件部分代码相似度较高，仅以 `CPU` 为例，给出其抽象类以及派生类的源代码

1. `component/CPU.java`

```
package component;

abstract public class CPU {
    private String name;
```

```java
    private int coreNum;
    private float price;

    public CPU(){

    }

    public CPU(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }

    public int getCoreNum() {
        return coreNum;
    }

    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    protected void setCoreNum(int coreNum){
        this.coreNum = coreNum;
    }

    public void work(){
        System.out.println(name + " work");
        System.out.println("========================");
    }

    abstract public void getInfo();
}
```

2. `brandcomp/Intel_CPU.java`

```java
package brandcomp;
import component.CPU;

public class Intel_CPU extends CPU {
    public Intel_CPU(int coreNum, float price){
        super("Intel_CPU");
        setCoreNum(coreNum);
        setPrice(price);
    }

    @Override
    public void getInfo(){
        System.out.println("name: " + getName());
        System.out.println("coreNum: " + getCoreNum());
        System.out.println("price: $" + getPrice());
        System.out.println("========================");
    }
}
```

3. `brandcomp/AMD_CPU.java`

```
package brandcomp;
import component.CPU;

public class AMD_CPU extends CPU {
    public AMD_CPU(int coreNum, float price){
        super("AMD_CPU");
        setCoreNum(coreNum);
        setPrice(price);
    }

    @Override
    public void getInfo(){
        System.out.println("name: " + getName());
        System.out.println("coreNum: " + getCoreNum());
        System.out.println("price: $" + getPrice());
        System.out.println("========================");
    }
}
```

4. `Computer.java`

```
import brandcomp.*;
import component.*;

public class Computer {
    private String  name;
    private CPU     cpu;
    private HD      hd;
    private RAM     ram;
    private MB      mb;
    private float   price;
    private static  int count = 0;

    public Computer(){
        final int   coreNum = 8;
        final float price1  = 1688;
        cpu = new AMD_CPU(coreNum, price1);

        final int   volume2 = 168;
        final float price2  = 688;
        hd  = new Seagate_HD(volume2, price2);

        final int   volume3 = 16;
        final float price3  = 288;
        ram = new Kingston_RAM(volume3, price3);

        final float speed   = (float)3.4;
        final float price4  = 588;
        mb  = new Gigabyte_MB(speed, price4);
        price = cpu.getPrice() + hd.getPrice() + ram.getPrice() + mb.getPrice();
        name = "Computer " + (++count);
    }

    public void getDescription(){
        System.out.println("==============================");
        System.out.println("      Info of " + name);
        System.out.println("==============================");
        cpu.getInfo();
        hd.getInfo();
        ram.getInfo();
        mb.getInfo();
    }

    public void work(){
        cpu.work();
        hd.work();
        ram.work();
        mb.work();
    }

    private void refreshPrice(){
        price = cpu.getPrice() + hd.getPrice() + ram.getPrice() + mb.getPrice();
    }

    public float getPrice(){
        return price;
    }

    public String getName() {
        return name;
    }

    public void setCpu(String brand, int coreNum, float price) throws Exception{
        if (brand.equalsIgnoreCase("Intel"))
            cpu = new Intel_CPU(coreNum, price);
        else if (brand.equalsIgnoreCase("AMD"))
            cpu = new AMD_CPU(coreNum, price);
        else throw new Exception("Illegal brand");

        refreshPrice();
    }

    public void setHd(String brand, int volume, float price) throws Exception{
        if (brand.equalsIgnoreCase("Seagate"))
            hd = new Seagate_HD(volume, price);
        else if (brand.equalsIgnoreCase("WestDigitals"))
            hd = new WestDigitals_HD(volume, price);
        else throw new Exception("Illegal brand");

        refreshPrice();
    }

    public void setRam(String brand, int volume, float price) throws Exception{
        if (brand.equalsIgnoreCase("Kingston"))
            ram = new Kingston_RAM(volume, price);
        else if (brand.equalsIgnoreCase("Samsung"))
```

```java
            ram = new Samsung_RAM(volume, price);
        else throw new Exception("Illegal brand");

        refreshPrice();
    }

    public void setMb(String brand, float speed, float price) throws Exception{
        if (brand.equalsIgnoreCase("Asus"))
            mb = new Asus_MB(speed, price);
        else if (brand.equalsIgnoreCase("Gigabyte"))
            mb = new Gigabyte_MB(speed, price);
        else throw new Exception("Illegal brand");

        refreshPrice();
    }

}
```

5. `ComputerStore.java`

```java
public class ComputerStore {
    private Computer[] computers;
    private int num_computer;

    public ComputerStore(){
        final int NUM_COMPUTER = 3;
        computers = new Computer[NUM_COMPUTER];
        num_computer = 0;
    }

    public void addComputer(Computer com){
        computers[num_computer++] = com;
    }

    public void getDescription(){
        for (int i = 0; i < num_computer; i++)
            computers[i].getDescription();
    }

    public void work(){
        for (int i = 0; i < num_computer; i++)
            computers[i].work();
    }

    public void printPrice(){
        for (int i = 0; i < num_computer; i++)
            System.out.println(computers[i].getName() + ": $" +
computers[i].getPrice());
        System.out.println("========================");
    }
}
```

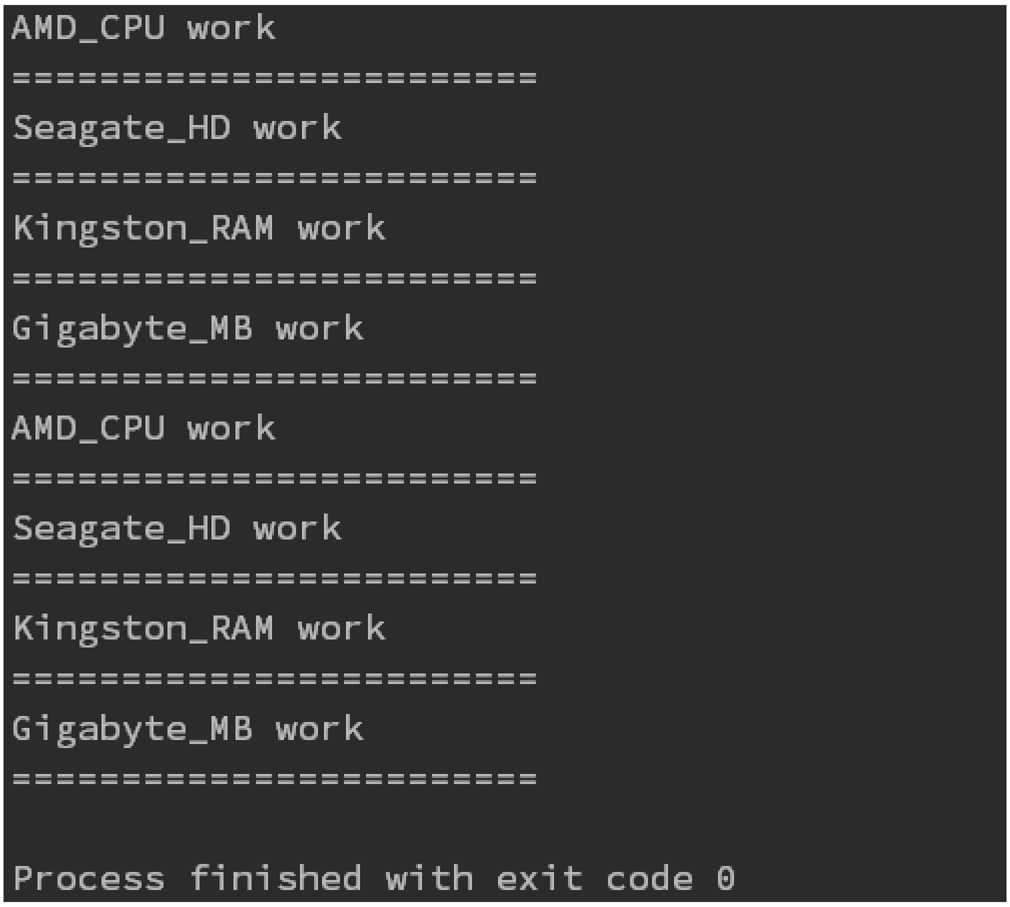
三、运行结果

1. 创建 `Computer`

测试代码如下:

```
Computer test1 = new Computer();
test1.work();

Computer test2 = new Computer();
test2.work();
```

测试结果如下:



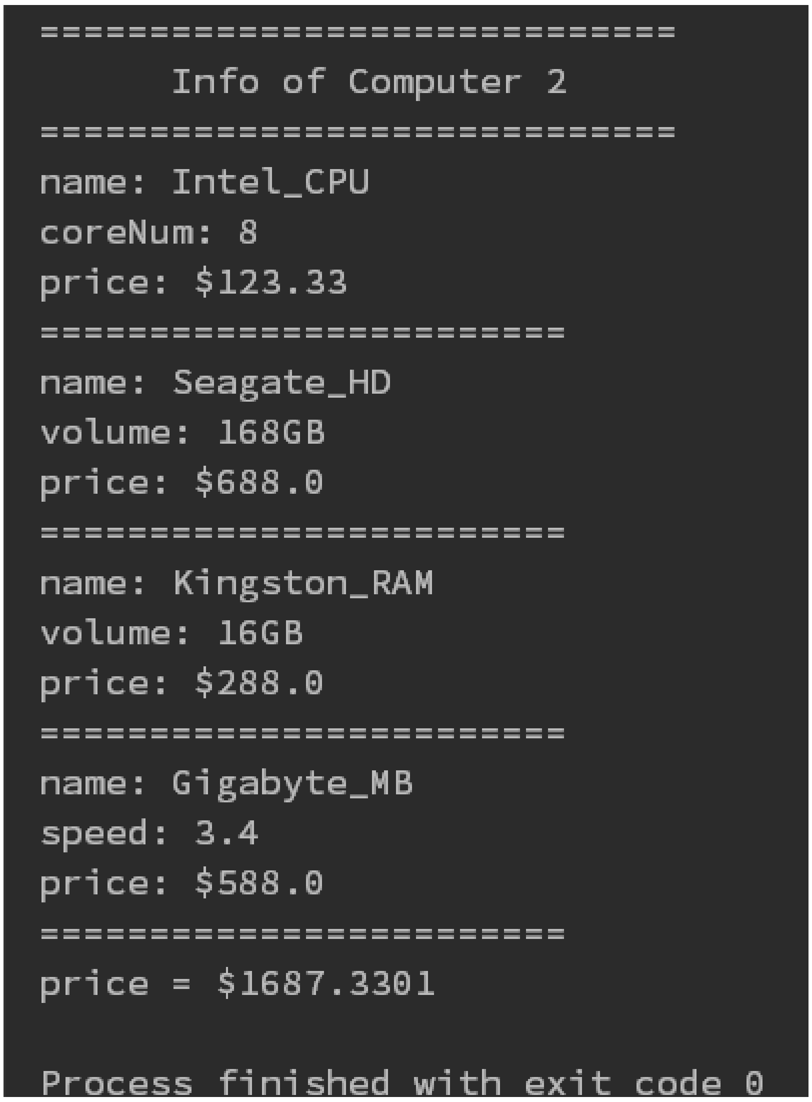
```
AMD_CPU work
========================
Seagate_HD work
========================
Kingston_RAM work
========================
Gigabyte_MB work
========================
AMD_CPU work
========================
Seagate_HD work
========================
Kingston_RAM work
========================
Gigabyte_MB work
========================

Process finished with exit code 0
```

2. 重新组装 `computer`

测试代码如下:

```
try
{
    test2.setCpu("inTel", 8, (float)123.33);
    test2.getDescription();
    System.out.println("price = $" + test2.getPrice());
}catch(Exception e){
    e.printStackTrace();
}
```

测试结果如下:



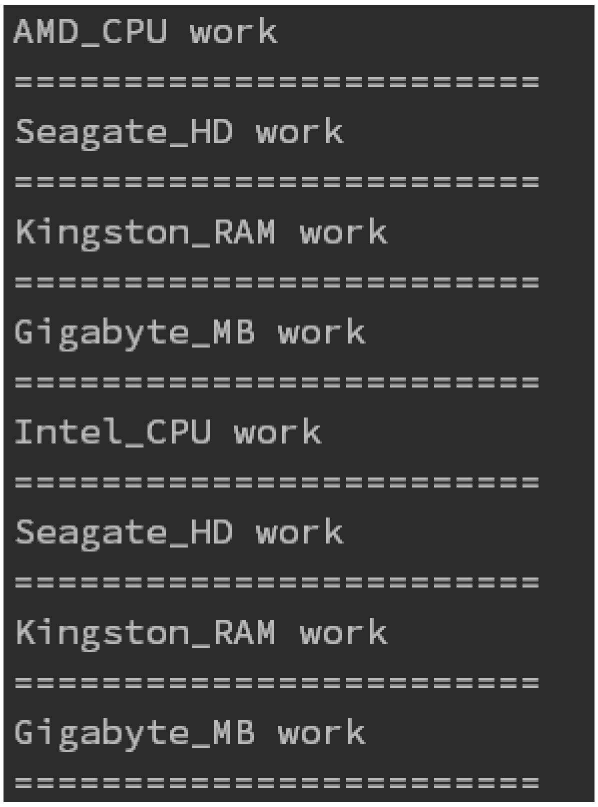
```
===============================
      Info of Computer 2
===============================
name: Intel_CPU
coreNum: 8
price: $123.33
=========================
name: Seagate_HD
volume: 168GB
price: $688.0
=========================
name: Kingston_RAM
volume: 16GB
price: $288.0
=========================
name: Gigabyte_MB
speed: 3.4
price: $588.0
=========================
price = $1687.3301

Process finished with exit code 0
```

3. 将 `computer` 加入商店

测试代码如下：

```
ComputerStore store = new ComputerStore();
store.addComputer(test1);
store.addComputer(test2);
store.work();
```

测试结果如下：



```
AMD_CPU work
========================
Seagate_HD work
========================
Kingston_RAM work
========================
Gigabyte_MB work
========================
Intel_CPU work
========================
Seagate_HD work
========================
Kingston_RAM work
========================
Gigabyte_MB work
========================
```

4. `ComputerStore` 的描述方法

测试代码如下：

```
store.getDescription();
```

测试结果如下：

```
=============================
      Info of Computer 1
=============================
name: AMD_CPU
coreNum: 8
price: $1688.0
========================
name: Seagate_HD
volume: 168GB
price: $688.0
========================
name: Kingston_RAM
volume: 16GB
price: $288.0
========================
name: Gigabyte_MB
speed: 3.4
price: $588.0
========================
=============================
      Info of Computer 2
=============================
name: Intel_CPU
coreNum: 8
price: $123.33
```

```
========================
name: Seagate_HD
volume: 168GB
price: $688.0
========================
name: Kingston_RAM
volume: 16GB
price: $288.0
========================
name: Gigabyte_MB
speed: 3.4
price: $588.0
========================
```