

LAB-2 Buy A Computer

Chen Junhua, Chen Yiyang, Xu Bo

1 Task Analysis

From the description of tasks, we can clearly feel that three design patterns can be used. They are factory method pattern, decorator pattern and singleton pattern. There are many entities in the task. We choose to analyze the relationship between them from the perspective of three design patterns

1.1 Factory Method Pattern

This is a typical factory pattern. Consumers buy computers from companies, and companies buy computer accessories from factories and assemble them according to customers' orders. All production activities are carried out in the factory, while Intel factory and Samsung factory only produce their own brand of CPU, motherboard and memory.

Users have their own ideas about the composition of computers. He doesn't just go to the company to buy a mass-produced computer, so the company can't predict in advance which factory to buy the required accessories, so we exclude the possibility of using the simple factory pattern.

1.2 Decorator Pattern

A computer is like a bowl of rice. You have to add all kinds of ingredients to make it a dish. You can add a variety of CPUs and several blocks of memory.

What's more, after you have used your computer for a while, you may want to add a new CPU to improve its speed. In this case, each accessory has its own price. In order to get the final cost of an assembled computer, we use the decorator pattern. In this way, it is convenient to add new functions on the basis of the original functions.

1.3 Singleton Pattern

Each computer has a unique ID, which is easy to associate with singleton pattern. In order to prevent the ID of this computer from being modified by others, especially when multithreading, it is necessary to design a single idpool. There are many government assigned IDs in the ID pool, and the qualified computer will get a unique ID.

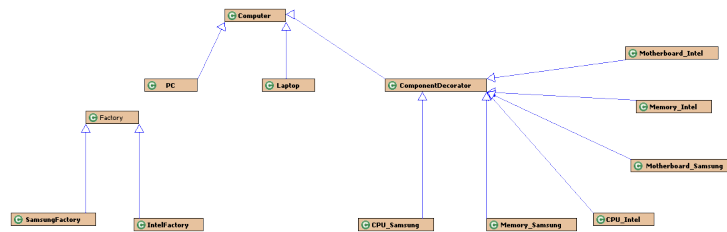


Figure 1: UML

```
public class SamsungFactory {
    protected Computer createComputer(String type) {
        Computer computer=null;
        if(type.equals("CPU_Samsung")) {
            computer=new CPU_Samsung(computer);
        }else if (type.equals("Memory_Samsung")) {
            computer=new Memory_Samsung(computer);
        }else if (type.equals("Motherboard_Samsung")) {
            computer=new Motherboard_Samsung(computer);
        }
        return computer;
    }
}
```

Figure 2: Take Samsung factory as an example

2 Design Scheme

From the overall logic, customer, computer, company and government need to be created. The computer class is a key class, which is divided into PC and laptop, and consists of CPU, memory and motherboard.

2.1 Factory Method Pattern

There is no doubt that an abstract factory class is required. The only responsibility of the factory class is to produce computer accessories, which are specifically implemented by its subclasses Intel factory and Samsung factory.

2.2 Decorator Pattern

First, we need a computer class, and two subclasses PC and laptop. In order to use the decorator pattern, we need to create another subclass componentdecorator class of the same level.

It is worth noting that we implement the `getcost()` method and `getdesc()` method in each computer class to implement the decorator pattern.

```

Computer computer;
double cost=2200;
String desc = "Intel CPU |"+cost+"$";

    public double getCost() {
        return cost+computer.getCost();
    }

```

2.3 Singleton Pattern

An idpoolsingleton class is designed by double_check lock.

This singleton class creates multiple instances and returns singletons based on the number passed in.

2.4 How to Purchase a Computer You Need

First, get the number of CPUs from different manufacturers, and judge whether it is used for PC or laptop. The same for other accessories.

Then in company a, we use the factory pattern to get the required CPU, and put it into a computer array kept in company A.

Finally, after the customer confirms the purchase, assemble all the accessories in the company and use the decorator pattern.

Of course, the PC itself cannot be forgotten.

2.5 How to Perform Quality Detection

The government tests every computer. Be careful. Some motherboards and CPUs can't work together. We simply use string character judgment here.

```

    public String getDesc() {
        return computer.getDesc()+" "+this.desc;
    }

```

```

public class IDpoolSingleton {
    static IDpoolSingleton []singletons=new IDpoolSingleton[20];
    String id;
    private IDpoolSingleton() {
    }

    public static String getInstance(int i) {
        if (singletons[i] == null) {
            synchronized (IDpoolSingleton.class) {
                if (singletons[i] == null) {
                    singletons[i]=new IDpoolSingleton();
                    singletons[i].id="2000"+i;
                }
            }
        }
        return singletons[i].getId();
    }
}

```

```

public void SetCpu(int intelNumber,int SamNumber) {
    if(label==0) {
        a_PC.GetCpu(intelNumber,SamNumber);
    }
    else if(label==1) {
        b_Laptop.GetCpu(intelNumber,SamNumber);
    }
}

```

```

ArrayList<Computer> computers=new ArrayList<Computer>();

```

```

public void GetCpu(int intelCpuNum,int SamCpuNum) {
    for(int i=0;i<intelCpuNum;i++) {
        Computer block1=intelFactory.createComputer("CPU_Intel");
        computers.add(block1);
    }
    for(int i=0;i<SamCpuNum;i++) {
        Computer block1=samsungFactory.createComputer("CPU_Samsung");
        computers.add(block1);
    }
}

```

```

public Computer AssembleComputer() {
    computer.setComputer(pc);
    for(int i=1;i<computers.size();i++) {
        Computer computer1=computers.get(i);
        Computer computer2=computers.get(i-1);
        computer1.setComputer(computer2);
        computer=computer1;
    }

    return computer;
}

```

```

public CompanyA() {
    computers.add(pc);
}

```

```

public void check(Computer computer,int i) {
    //Intel cpu and Samsung motherboard can not work together
    if(computer.getDesc().contains("Intel CPU")&&computer.getDesc().contains("Samsung Motherboard"))
        quality=1;
    if(quality==0) {
        computer.id=IDpoolSingleton.getInstance(i);
        System.out.println("You get your unique Id");
        System.out.println(computer.id);
    }
    else {
        System.out.println("This is a pool quality computer,You can get your refund");
    }
}
}

```

```

Computer computer;
Government government=new Government();
Customer customer1=new Customer();
//computer type, PC or Laptop
customer1.WantType("Laptop");
customer1.SetCpu(0, 1);
customer1.SetMem(1, 2);
customer1.SetMotherboard(1, 0);
System.out.println("Total price:"+customer1.getPrice());
computer=customer1.check();
System.out.println(computer.getDesc());
government.check(computer,1);

```

```

Total price:7700.0
Laptop machine 3000.0$ + Samsung CPU 2000.0$ + Intel Memory 400.0$ + Samsung Memory 500.0$ + Samsung Memory 500.0$ + Intel Motherboard 1300.0$
You get your unique Id
20001
*****

```

```

Computer computer1=new Computer();
customer1.WantType("Laptop");
customer1.SetCpu(1,1 );
customer1.SetMem(1, 1);
customer1.SetMotherboard(2, 1);
System.out.println("Total price:"+customer1.getPrice());
computer1=customer1.check();
System.out.println(computer1.getDesc());
government.check(computer1,6);

```

```

Total price:16400.0
Laptop machine 3000.0$ + Samsung CPU 2000.0$ + Intel Memory 400.0$ + Samsung Memory 500.0$ + Samsung Memory 500.0$ + Intel Motherboard 1300.0$ + Intel CPU 2.
This is a pool quality computer,You can get your refund

```

3 Experimental Result

3.1 Qualified Computer

3.2 Unqualified Computer

4 Conclusion Analysis

4.1 Difficulties and Key Points

The difficulty lies in how to better apply the design patterns we have learned, and to figure out the connections between objects.

4.2 Group Reflection

This is the first time to complete a lab project in the form of a group. It is also a comprehensive use of various software design patterns. In the process of step-by-step analysis, we have a deeper understanding of the characteristics and use conditions of various modes. We can also learn a lot from group work. From the discussion in the analysis stage to the division of labor in the specific work, we have gained something hard to get from personal work.