**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\_\_3018216056\_\_**

**学 号\_\_3018216056\_\_**

**年 级\_ 2018级 \_**

**班 级\_软件工程一班\_**

1. **实验要求**

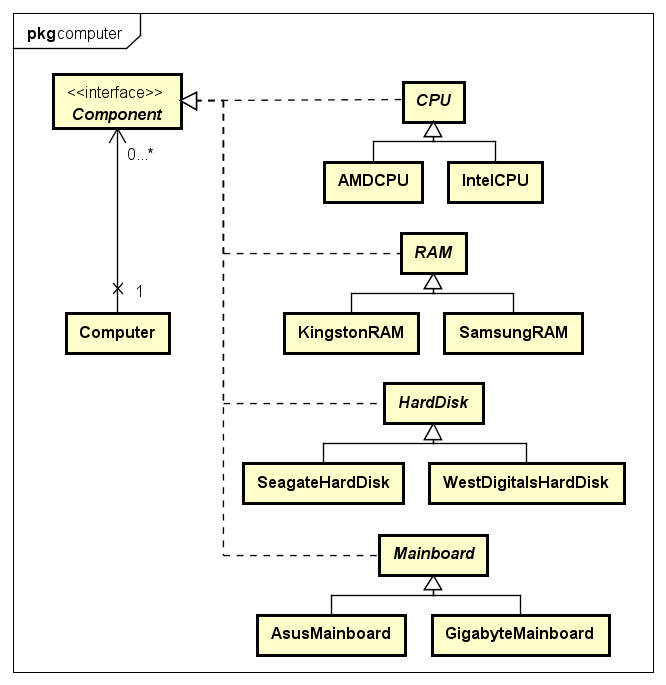


1. **设计思路**

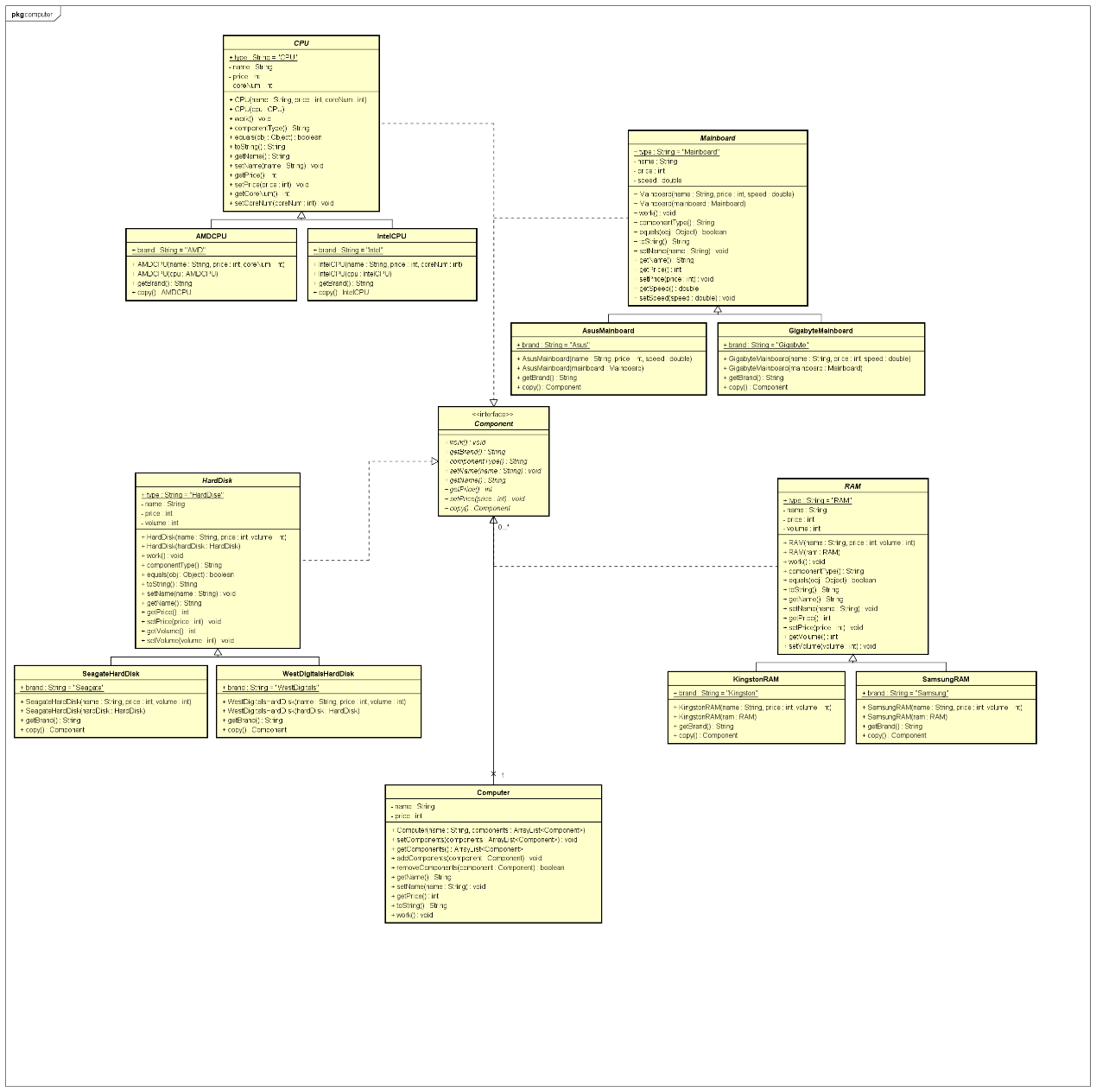
首先，四种组件可以抽象为一个“组件”接口，接口中定义一些组件共有的方法，即统一了标准，又方便了后续的组件的扩展。

接下来，我们只需要设计拥有组件的计算机类即可，在其中定义和实现一些对组件的操作，以及对电脑本身的描述。

整体类图如下：



详细类图：



类图及代码内容较多，附件中查看更方便。

1. **测试代码及结果**

ComputerStore主类

package edu.tju.computer;

import java.util.ArrayList;

public class ComputerStore {

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

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

        System.out.println("1.New a Computer Lenovo-999");

        ArrayList<Component> components = new ArrayList<Component>();

        components.add(new IntelCPU("Inter-i7", 3000, 4));

        components.add(new SamsungRAM("Sam99", 1000, 500));

        components.add(new SeagateHardDisk("Sea88", 500, 1024));

        components.add(new AsusMainboard("Asus33", 200, 3.2));

        computers.add(new Computer("Lenovo-999", components));

        System.out.println(computers.get(0).toString());

*// 测试结果：*

*//*

*// 1.New a Computer Lenovo-999*

*// name: Lenovo-999*

*// price: 4700*

*// components:*

*//  CPU [name: Inter-i7, price: 3000, coreNum: 4, brand: Intel]*

*//  RAM [name: Sam99, price: 1000, volume: 500, brand: Samsung]*

*//  HardDisk [name: Sea88, price: 500, volume: 1024, brand: Seagate]*

*//  Mainboard [name: Asus33, price: 200, speed: 3.2, brand: Asus]*

        System.out.println("2.Add a hard disk for Lenvo-999");

        computers.get(0).addComponents(new SeagateHardDisk("Seagate-00", 300, 500));

        System.out.println(computers.get(0).toString());

*// 测试结果*

*//*

*// 2.Add a hard disk for Lenvo-999*

*// name: Lenovo-999*

*// price: 5000*

*// components:*

*//  CPU [name: Inter-i7, price: 3000, coreNum: 4, brand: Intel]*

*//  RAM [name: Sam99, price: 1000, volume: 500, brand: Samsung]*

*//  HardDisk [name: Sea88, price: 500, volume: 1024, brand: Seagate]*

*//  Mainboard [name: Asus33, price: 200, speed: 3.2, brand: Asus]*

*//  HardDisk [name: Seagate-00, price: 300, volume: 500, brand: Seagate]*

        System.out.println("3.Test Lenvo-999's work");

        computers.get(0).work();

        System.out.println();

*// 测试结果*

*//*

*// 3.Test Lenvo-999's work*

*// CPU work*

*// RAM work*

*// HardDisk work*

*// Mainboard work*

*// HardDisk work*

        System.out.println("4.New a Computer Dell-888");

        components = new ArrayList<Component>();

        components.add(new AMDCPU("AMD11", 2000, 4));

        components.add(new KingstonRAM("King22", 800, 600));

        components.add(new WestDigitalsHardDisk("Wes77", 600, 2048));

        components.add(new GigabyteMainboard("Giga90", 400, 4.1));

        computers.add(new Computer("Dell-888", components));

        System.out.println(computers.get(1).toString());

*// 测试结果*

*//*

*// 4.New a Computer Dell-888*

*// name: Dell-888*

*// price: 3800*

*// components:*

*//  CPU [name: AMD11, price: 2000, coreNum: 4, brand: AMD]*

*//  RAM [name: King22, price: 800, volume: 600, brand: Kingston]*

*//  HardDisk [name: Wes77, price: 600, volume: 2048, brand: WestDigitals]*

*//  Mainboard [name: Giga90, price: 400, speed: 4.1, brand: Gigabyte]*

        System.out.println("5.New a Computer hp-777");

        components = new ArrayList<Component>();

        components.add(new IntelCPU("Intel-i5", 2500, 4));

        components.add(new KingstonRAM("King22", 1000, 600));

        components.add(new SeagateHardDisk("Sea77", 600, 2048));

        components.add(new GigabyteMainboard("Giga88", 400, 4.1));

        computers.add(new Computer("hp-777", components));

        System.out.println(computers.get(2).toString());

*// 测试结果*

*//*

*// 5.New a Computer hp-777*

*// name: hp-777*

*// price: 4500*

*// components:*

*//  CPU [name: Intel-i5, price: 2500, coreNum: 4, brand: Intel]*

*//  RAM [name: King22, price: 1000, volume: 600, brand: Kingston]*

*//  HardDisk [name: Sea77, price: 600, volume: 2048, brand: Seagate]*

*//  Mainboard [name: Giga88, price: 400, speed: 4.1, brand: Gigabyte]*

        System.out.println("6.Remove RAM from hp-777:");

        computers.get(2).removeComponents(new KingstonRAM("King22", 1000, 600));

        System.out.println(computers.get(2).toString());

*// 测试结果*

*//*

*// 6.Remove RAM from hp-777:*

*// name: hp-777*

*// price: 3500*

*// components:*

*//  CPU [name: Intel-i5, price: 2500, coreNum: 4, brand: Intel]*

*//  HardDisk [name: Sea77, price: 600, volume: 2048, brand: Seagate]*

*//  Mainboard [name: Giga88, price: 400, speed: 4.1, brand: Gigabyte]*

    }

}

1. **源代码**

Component接口

package edu.tju.computer;

public interface Component {

    public void work();

    public String getBrand();

    public String componentType();

    public void setName(String *name*);

    public String getName();

    public int getPrice();

    public void setPrice(int *price*);

    public Component copy();

}

CPU抽象类

package edu.tju.computer;

public abstract class CPU implements Component {

    final public static String type = "CPU";

    private String name;

    private int price;

    private int coreNum;

    public CPU(String *name*, int *price*, int *coreNum*) {

        this.name = name;

        this.price = price;

        this.coreNum = coreNum;

    }

    public CPU(CPU *cpu*) {

        this.name = cpu.name;

        this.price = cpu.price;

        this.coreNum = cpu.coreNum;

    }

    @Override

    public void work() {

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

    }

    @Override

    final public String componentType() {

        return type;

    }

    @Override

    public boolean equals(Object *obj*) {

        if (this == obj)

            return true;

        if (obj == null)

            return false;

        if (!(obj instanceof CPU))

            return false;

        CPU other = (CPU) obj;

        if (this.getBrand() != other.getBrand())

            return false;

        if (coreNum != other.coreNum)

            return false;

        if (name == null) {

            if (other.name != null)

                return false;

        } else if (!name.equals(other.name))

            return false;

        if (price != other.price)

            return false;

        return true;

    }

    @Override

    public String toString() {

        return "CPU [name: " + name + ", price: " + price + ", coreNum: " + coreNum + ", brand: " + this.getBrand() + "]";

    }

    @Override

    public String getName() {

        return name;

    }

    @Override

    public void setName(String *name*) {

        this.name = name;

    }

    @Override

    public int getPrice() {

        return price;

    }

    @Override

    public void setPrice(int *price*) {

        this.price = price;

    }

    public int getCoreNum() {

        return coreNum;

    }

    public void setCoreNum(int *coreNum*) {

        this.coreNum = coreNum;

    }

}

IntelCPU类

package edu.tju.computer;

public class IntelCPU extends CPU {

    final public static String brand = "Intel";

    public IntelCPU(String *name*, int *price*, int *coreNum*) {

        super(name, price, coreNum);

    }

    public IntelCPU(IntelCPU *cpu*) {

        super(cpu);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public IntelCPU copy() {

        return new IntelCPU(this);

    }

}

AMDCPU类

package edu.tju.computer;

public class AMDCPU extends CPU {

    final public static String brand = "AMD";

    public AMDCPU(String *name*, int *price*, int *coreNum*) {

        super(name, price, coreNum);

    }

    public AMDCPU(AMDCPU *cpu*) {

        super(cpu);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public AMDCPU copy() {

        return new AMDCPU(this);

    }

}

RAM抽象类

package edu.tju.computer;

public abstract class RAM implements Component {

    final public static String type = "RAM";

    private String name;

    private int price;

    private int volume;

    public RAM(String *name*, int *price*, int *volume*) {

        this.name = name;

        this.price = price;

        this.volume = volume;

    }

    public RAM(RAM *ram*) {

        this.name = ram.name;

        this.price = ram.price;

        this.volume = ram.volume;

    }

    @Override

    public void work() {

        System.out.println("RAM work");

    }

    @Override

    public String componentType() {

        return type;

    }

    @Override

    public boolean equals(Object *obj*) {

        if (this == obj)

            return true;

        if (obj == null)

            return false;

        if (!(obj instanceof RAM))

            return false;

        RAM other = (RAM) obj;

        if (this.getBrand() != other.getBrand())

            return false;

      if (name == null) {

            if (other.name != null)

                return false;

        } else if (!name.equals(other.name))

            return false;

        if (price != other.price)

            return false;

        if (volume != other.volume)

            return false;

        return true;

    }

    @Override

    public String toString() {

        return "RAM [name: " + name + ", price: " + price + ", volume: " + volume + ", brand: " + this.getBrand() + "]";

    }

    @Override

    public String getName() {

        return name;

    }

    @Override

    public void setName(String *name*) {

        this.name = name;

    }

    @Override

    public int getPrice() {

        return price;

    }

    @Override

    public void setPrice(int *price*) {

        this.price = price;

    }

    public int getVolume() {

        return volume;

    }

    public void setVolume(int *volume*) {

        this.volume = volume;

    }

}

SamsumgRAM类

package edu.tju.computer;

public class SamsungRAM extends RAM {

    final public static String brand = "Samsung";

    public SamsungRAM(String *name*, int *price*, int *volume*) {

        super(name, price, volume);

    }

    public SamsungRAM(RAM *ram*) {

        super(ram);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public Component copy() {

        return new SamsungRAM(this);

    }

}

KingstonRAM类

package edu.tju.computer;

public class KingstonRAM extends RAM {

    final public static String brand = "Kingston";

    public KingstonRAM(String *name*, int *price*, int *volume*) {

        super(name, price, volume);

    }

    public KingstonRAM(RAM *ram*) {

        super(ram);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public Component copy() {

        return new KingstonRAM(this);

    }

}

HardDisk抽象类

package edu.tju.computer;

public abstract class HardDisk implements Component {

    final public static String type = "HardDise";

    private String name;

    private int price;

    private int volume;

    public HardDisk(String *name*, int *price*, int *volume*) {

        this.name = name;

        this.price = price;

        this.volume = volume;

    }

    public HardDisk(HardDisk *hardDisk*) {

        this.name = hardDisk.name;

        this.price = hardDisk.price;

        this.volume = hardDisk.volume;

    }

    @Override

    public void work() {

        System.out.println("HardDisk work");

    }

    @Override

    public String componentType() {

        return type;

    }

    @Override

    public boolean equals(Object *obj*) {

        if (this == obj)

            return true;

        if (obj == null)

            return false;

        if (!(obj instanceof HardDisk))

            return false;

        HardDisk other = (HardDisk) obj;

        if (this.getBrand() != other.getBrand())

            return false;

        if (name == null) {

            if (other.name != null)

                return false;

        } else if (!name.equals(other.name))

            return false;

        if (price != other.price)

            return false;

        if (volume != other.volume)

            return false;

        return true;

    }

    @Override

    public String toString() {

        return "HardDisk [name: " + name + ", price: " + price + ", volume: " + volume + ", brand: " + this.getBrand() + "]";

    }

    @Override

    public void setName(String *name*) {

        this.name = name;

    }

    @Override

    public String getName() {

        return name;

    }

    @Override

    public int getPrice() {

        return price;

    }

    @Override

    public void setPrice(int *price*) {

        this.price = price;

    }

    public int getVolume() {

        return volume;

    }

    public void setVolume(int *volume*) {

        this.volume = volume;

    }

}

SeagateHardDisk类

package edu.tju.computer;

public class SeagateHardDisk extends HardDisk {

    final public static String brand = "Seagate";

    public SeagateHardDisk(String *name*, int *price*, int *volume*) {

        super(name, price, volume);

    }

    public SeagateHardDisk(HardDisk *hardDisk*) {

        super(hardDisk);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public Component copy() {

        return new SeagateHardDisk(this);

    }

}

WestDigitalsHardDisk类

package edu.tju.computer;

public class WestDigitalsHardDisk extends HardDisk {

    final public static String brand = "WestDigitals";

    public WestDigitalsHardDisk(String *name*, int *price*, int *volume*) {

        super(name, price, volume);

    }

    public WestDigitalsHardDisk(HardDisk *hardDisk*) {

        super(hardDisk);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public Component copy() {

        return new WestDigitalsHardDisk(this);

    }

}

Mainboard抽象类

package edu.tju.computer;

public abstract class Mainboard implements Component {

    final public static String type = "Mainboard";

    private String name;

    private int price;

    private double speed;

    public Mainboard(String *name*, int *price*, double *speed*) {

        this.name = name;

        this.price = price;

        this.speed = speed;

    }

    public Mainboard(Mainboard *mainboard*) {

        this.name = mainboard.name;

        this.price = mainboard.price;

        this.speed = mainboard.speed;

    }

    @Override

    public void work() {

        System.out.println("Mainboard work");

    }

    @Override

    public String componentType() {

        return type;

    }

    @Override

    public boolean equals(Object *obj*) {

        if (this == obj)

            return true;

        if (obj == null)

            return false;

        if (!(obj instanceof Mainboard))

            return false;

        Mainboard other = (Mainboard) obj;

        if (this.getBrand() != other.getBrand())

            return false;

        if (name == null) {

            if (other.name != null)

                return false;

        } else if (!name.equals(other.name))

            return false;

        if (price != other.price)

            return false;

        if (Double.doubleToLongBits(speed) != Double.doubleToLongBits(other.speed))

            return false;

        return true;

    }

    @Override

    public String toString() {

        return "Mainboard [name: " + name + ", price: " + price + ", speed: " + speed + ", brand: " + this.getBrand() + "]";

    }

    @Override

    public void setName(String *name*) {

        this.name = name;

    }

    @Override

    public String getName() {

        return name;

    }

    @Override

    public int getPrice() {

        return price;

    }

    @Override

    public void setPrice(int *price*) {

        this.price = price;

    }

    public double getSpeed() {

        return speed;

    }

    public void setSpeed(double *speed*) {

        this.speed = speed;

    }

}

AsusMainboard类

package edu.tju.computer;

public class AsusMainboard extends Mainboard {

    final public static String brand = "Asus";

    public AsusMainboard(String *name*, int *price*, double *speed*) {

        super(name, price, speed);

    }

    public AsusMainboard(Mainboard *mainboard*) {

        super(mainboard);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public Component copy() {

        return new AsusMainboard(this);

    }

}

GigabyteMainboard类

package edu.tju.computer;

public class GigabyteMainboard extends Mainboard {

    final public static String brand = "Gigabyte";

    public GigabyteMainboard(String *name*, int *price*, double *speed*) {

        super(name, price, speed);

    }

    public GigabyteMainboard(Mainboard *mainboard*) {

        super(mainboard);

    }

    @Override

    public String getBrand() {

        return brand;

    }

    @Override

    public Component copy() {

        return new GigabyteMainboard(this);

    }

}

Computer类

package edu.tju.computer;

import java.util.ArrayList;

public class Computer {

    private String name;

    ArrayList<Component> components;

    private int price;

    public Computer(String *name*,ArrayList<Component> *components*) {

        this.name = name;

        this.setComponents(components);

    }

    public void setComponents(ArrayList<Component> *components*) {

        this.components = new ArrayList<Component>();

        this.price = 0;

        for (Component component : components) {

            this.components.add(component.copy());

            this.price += component.getPrice();

        }

    }

    public ArrayList<Component> getComponents() {

        ArrayList<Component> components = new ArrayList<Component>();

        for (Component component : this.components) {

            components.add(component.copy());

        }

        return components;

    }

    public void addComponents(Component *component*) {

        components.add(component.copy());

        this.price += component.getPrice();

    }

    public boolean removeComponents(Component *component*) {

        if (components.remove(component)) {

            this.price -= component.getPrice();

            return true;

        }

        else return false;

    }

    public String getName() {

        return name;

    }

    public void setName(String *name*) {

        this.name = name;

    }

    public int getPrice() {

        return price;

    }

    @Override

    public String toString() {

        String s = "name: " + name + '\n' + "price: " + price + '\n' + "components:\n";

        for (Component component: components) {

            s += '\t' + component.toString() + '\n';

        }

        return s;

    }

    public void work() {

        for (Component component: components) {

            component.work();

        }

    }

    public ArrayList<Component> getComponentsByType(String *type*) {

        ArrayList<Component> components = new ArrayList<Component>();

        for (Component component : this.components) {

            if (component.componentType().equals(type))

            components.add(component.copy());

        }

        return components;

    }

    public ArrayList<Component> getComponentsByBrand(String *brand*) {

        ArrayList<Component> components = new ArrayList<Component>();

        for (Component component : this.components) {

            if (component.getBrand().equals(brand))

                components.add(component.copy());

        }

        return components;

    }

    public ArrayList<Component> getComponentsByName(String *name*) {

        ArrayList<Component> components = new ArrayList<Component>();

        for (Component component : this.components) {

            if (component.getName().equals(name))

                components.add(component.copy());

        }

        return components;

    };

}