Java 核心技术

第十二章 Java案例实践和总结

第一节 Java案例实践

华东师范大学 陈良育

概要

- 矿机有限公司--面向对象设计
- WordCount--文件大数据统计

矿机有限公司的需求

- 本公司主要销售各类组装矿机，用于挖矿(比特币)服务。
- 矿机(各种组合)
  - CPU：Intel, AMD
  - GPU：Nvidia，ATI
  - 主板：Asus, MSI, Gigabyte,…
  - 电源：长城、金河田、航嘉，…
  - 硬盘：希捷、西数、三星，…
- 商品一览表，可以查询商品价格等

提取CPU公共特征

Intel i5
6核
1599￥

AMD r7
8核
2599￥

CPU提取公共特征
name
coreNum
price

提取CPU/Disk公共特征

CPU提取公共特征
name
coreNum
price

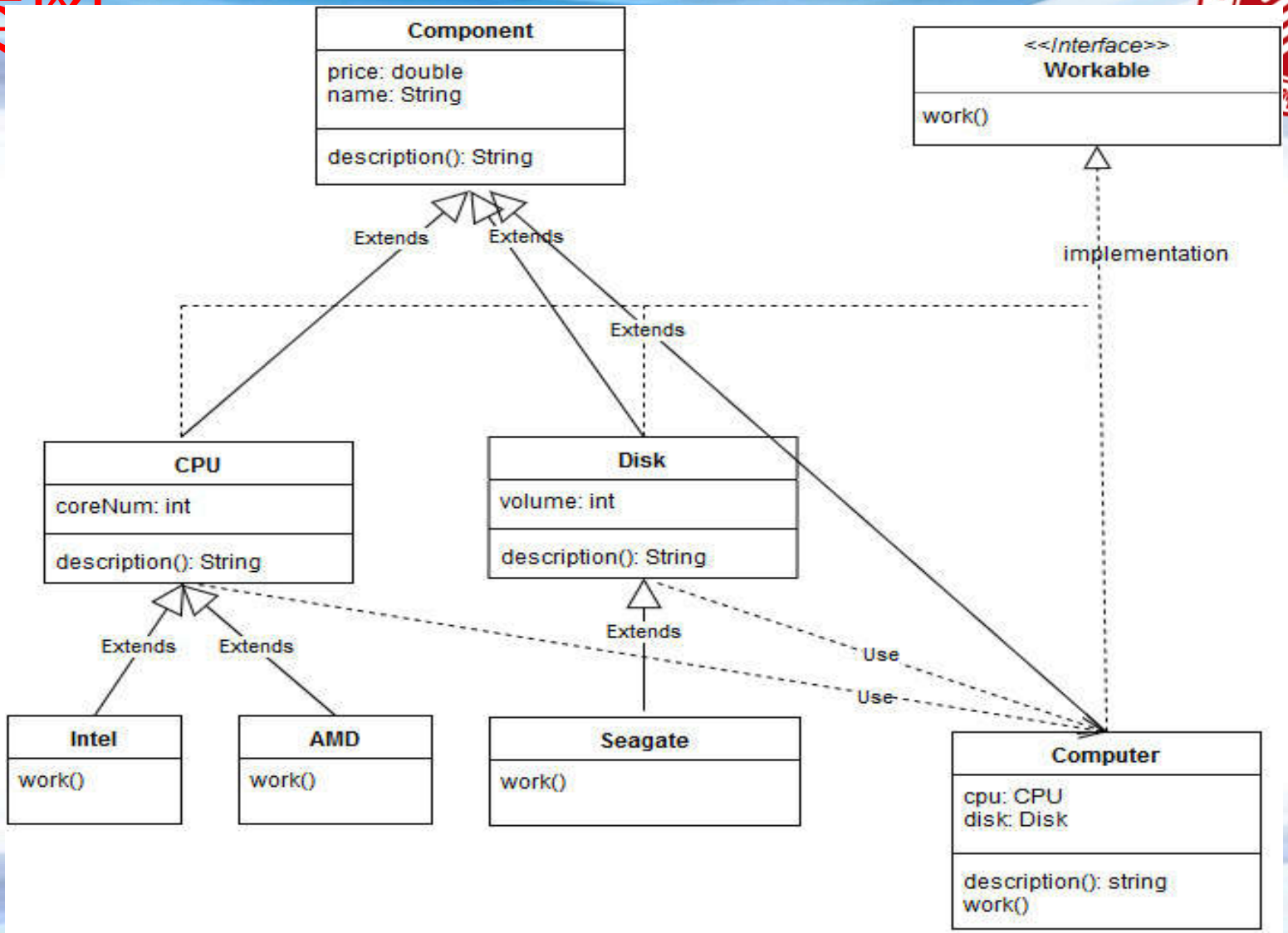
Disk提取公共特征
name

Component
name
price

CPU
coreNum

Disk
volume

UML类图


Component
price: double
name: String
description(): String
<<Interface>>
Workable
work()
Extends
Extends
implementation
Extends
CPU
coreNum: int
description(): String
Disk
volume: int
description(): String
Extends
Extends
Extends
Use
Use
Intel
work()
AMD
work()
Seagate
work()
Computer
cpu: CPU
disk: Disk
description(): string
work()

WordCount的需求

- 统计一个目录下所有文本文件里面每个出现的单词次数
  - 大数据分析的入门案例
- 单词次数统计
  - 需要遍历所有的txt文件
  - 读取每个文件，逐行解析单词
  - 存储单词次数
  - 最后将单词次数按大小降序排列输出

WordCount的分析

- 需要遍历所有的txt文件
  - 给定一个目录，遍历所有的子文件夹，递归或者FileSystem
- 读取每个文件，逐行解析单词
  - 每次readLine读取一行，split为数组，再单独解析
- 存储单词次数
  - 相同的单词的次数需要累计，可以用HashMap
- 最后将单词次数按大小降序排列输出
  - 排序输出可以考虑Comparable接口

总结

- 善于分析系统，将问题分解，各个击破
- 从众多对象中提取共性，形成父类(接口)和子类
  - 父类(接口)负责规范，子类负责实现
- 善于查询API和使用API

代码(1) Component.java


```
public abstract class Component {

    private double price; //单位元
    private String name;

    public Component(String name, double price) {
        this.price = price;
        this.name = name;
    }

    public String description() {
        StringBuilder descriptionBuilder = new StringBuilder();
        descriptionBuilder.append("Name:{")
            .append(name)
            .append("}; Price:{")
            .append(price)
            .append("};");
        return descriptionBuilder.toString();
    }
```

代码(2) Component.java


```
    public double getPrice() {
        return price;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

代码(3) Workable.java


```java
/**
 * Workable 可工作能力的接口
 * @author Tom
 *
 */
public interface Workable {
    void work();
}
```

代码(4) CPU.java


```
public abstract class CPU extends Component implements Workable {

    protected int coreNum;

    public CPU(String name, int coreNum, double price) {
        super(name, price);
        this.coreNum = coreNum;
    }

    @Override
    public String description() {
        StringBuilder cpuDescBuilder = new StringBuilder();
        cpuDescBuilder.append(super.description())
            .append(" Core Num:{")
            .append(coreNum)
            .append("}");
        return cpuDescBuilder.toString();
    }

}
```

代码(5) Disk.java


```java
public abstract class Disk extends Component implements Workable {
    /**
     * 单位:MB
     */
    protected int volume;

    public Disk(String name, double price, int volume) {
        super(name, price);
        this.volume = volume;
    }


    @Override
    public String description() {
        StringBuilder cpuDescBuilder = new StringBuilder();
        cpuDescBuilder.append(super.description())
                .append(" size(MB):{")
                .append(volume)
                .append(" MB}");
        return cpuDescBuilder.toString();
    }

}
```

代码(6) AMDCPU.java


```java
public class AMDCPU extends CPU {

    public AMDCPU(String name, int coreNum, double price) {
        super(name, coreNum, price);
    }

    public void work() {
        System.out.println("This is AMD's cpu working");
    }

}
```

代码(7) IntelCPU.java


```
public class IntelCPU extends CPU {

    public IntelCPU(String name, int coreNum, double price) {
        super(name, coreNum, price);
    }

    public void work() {
        System.out.println("This is Intel's CPU working");
    }

}
```

代码(8) Seagate.java


```
public class Seagate extends Disk {

    public Seagate(String name, double price, int volume) {
        super(name, price, volume);
    }

    public void work() {
        System.out.println("This is seagate disk working");
    }

}
```

代码(9) WestDigital.java


```
public class WestDigital extends Disk {

    public WestDigital(String name, double price, int volume) {
        super(name, price, volume);
    }

    public void work() {
        System.out.println("This is westdigital disk working");
    }

}
```

代码(10) Computer.java


```
public class Computer extends Component implements Workable {

    private CPU cpu;
    private Disk disk;

    public Computer(String name, CPU cpu, Disk disk){
        super(name, -1);
        this.cpu = cpu;
        this.disk = disk;
        super.setPrice(cpu.getPrice() + disk.getPrice());
    }
```

代码(11) Computer.java


```
    public void work() {
        cpu.work();
        disk.work();
    }

    public String description() {
        StringBuilder cmputerDescBuilder = new StringBuilder();
        cmputerDescBuilder.append(super.description())
            .append("CPU description: {")
            .append(cpu.description())
            .append("}")
            .append("; Disk descripton: {")
            .append(disk.description())
            .append("}");
        return cmputerDescBuilder.toString();

    }
}
```

代码(12) ComputerStore.java


```
public class ComputerStore {
    public static void main(String[] args) {

        //定义第一台机器的部件
        CPU intel = new IntelCPU("intel cpu", 2, 1000);
        Disk seagate = new Seagate("Seagate disk", 1000, 40960);

        //组装第一台机器并工作
        Computer computer1 = new Computer("computer 1", intel, seagate);
        System.out.println("computer 1 description: " + computer1.description());
        System.out.println("Computer 1 work: ");
        computer1.work();

        //定义第二台机器的部件
        CPU amd = new AMDCPU("AMD cpu", 800, 2);
        Disk westdigital = new WestDigital("West Digital disk", 2000, 81920);

        //组装第二台机器并工作
        Computer computer2 = new Computer("computer 2", amd, westdigital);
        System.out.println("computer 2 description: " + computer2.description());
        System.out.println("Computer 2 work: ");
        computer2.work();
    }
}
```

代码(13) FileAnalyzer.java


```java
public class FileAnalyzer {
    private String fileStr;

    public FileAnalyzer(String fileStr)
    {
        this.fileStr = fileStr;
    }

    /**
     * getWordCount() 获取一个文件内的单词数
     * @return
     */
    public HashMap<String, Word> getWordCount()
    {
        HashMap<String, Word> result = new HashMap<String, Word>();

        String line;
```

代码(14) FileAnalyzer.java


```
        try (BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(fileStr)))) {
            while ((line = in.readLine()) != null) {
                String[] words = line.split(" ");
                for(String word : words)
                {
                    if(null!=word && word.length()>0)
                    {
                        if(result.containsKey(word))
                        {
                            Word w = result.get(word);
                            w.setTimes(w.getTimes() + 1);
                        }
                        else
                        {
                            result.put(word, new Word(word, 1));
                        }
                    }
                }
            }
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }

        return result;
    }
}
```

代码(15) Searcher.java


```
class Searcher implements FileVisitor {

    private final PathMatcher matcher;
    private ArrayList<String> filePaths = new ArrayList<String>();

    public Searcher(String ext) {
        matcher = FileSystems.getDefault().getPathMatcher("glob:" + ext);
    }

    public void judgeFile(Path file) throws IOException {
        Path name = file.getFileName();
        if (name != null && matcher.matches(name)) {
            //System.out.println("Searched file was found: " + name + " in " + file.toRealPath().toString());
            filePaths.add(file.toRealPath().toString());
        }
    }

    @Override
    public FileVisitResult postVisitDirectory(Object dir, IOException exc) throws IOException {
        //System.out.println("Visited: " + (Path) dir);
        return FileVisitResult.CONTINUE;
    }
```

代码(16) Searcher.java


```
    @Override
    public FileVisitResult preVisitDirectory(Object dir, BasicFileAttributes attrs) throws IOException {
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult visitFile(Object file, BasicFileAttributes attrs) throws IOException {
        judgeFile((Path) file);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult visitFileFailed(Object file, IOException exc) throws IOException {
        return FileVisitResult.CONTINUE;
    }

    /**
     * getFilePaths 返回满足条件的文件列表
     * @return
     */
    public ArrayList<String> getFilePaths() {
        return filePaths;
    }

}
```

代码(17) Word.java


```
public class Word implements Comparable<Word> {
    private String text;   //单词文本
    private int times;     //单词次数

    public String getText() {
        return text;
    }
    public void setText(String text) {
        this.text = text;
    }
    public int getTimes() {
        return times;
    }
    public void setTimes(int times) {
        this.times = times;
    }
    public Word(String text, int times) {
        this.text = text;
        this.times = times;
    }

    /**
     * 定义两个单词的排序,逆序输出
     */
    public int compareTo(Word a) {
        return (-1) * (this.getTimes() - a.getTimes());
    }
}
```

代码(18) WordCounter.java


```
public class WordCounter {

    public static void main(String[] args) throws IOException {

        //接收目录参数和扩展名
        Path fileTree = Paths.get("C:/temp/");
        Searcher walk = new Searcher("*.txt");


        //查找该目录下所有的txt文件
        EnumSet<FileVisitOption> opts = EnumSet.of(FileVisitOption.FOLLOW_LINKS);
        Files.walkFileTree(fileTree, opts, Integer.MAX_VALUE, walk);
        ArrayList<String> filePaths = walk.getFilePaths();
        //System.out.println(filePaths.get(0));
```

代码(19) WordCounter.java


```
        //解析每个文件的单词

        HashMap<String, Word> totalMap = new HashMap<String, Word>();

        for(String str:filePaths)
        {
            HashMap<String, Word> partMap = new FileAnalyzer(str).getWordCount();
            if(partMap != null && partMap.size() > 0)
            {
                combineMap(totalMap, partMap);
            }
        }

        //排序
        //Collection<Word> words = totalMap.values();
        ArrayList<Word> words = new ArrayList<Word>(totalMap.values());
        Collections.sort(words);

        //输出
        System.out.println("最后结果");
        for(Word w : words)
        {
            System.out.println(w.getText() + ", " + w.getTimes());
        }
    }
```

代码(20) WordCounter.java


```
    public static void combineMap(HashMap<String, Word> total, HashMap<String, Word> part)
    {
        Iterator<Entry<String, Word>> iter = part.entrySet().iterator();
        while(iter.hasNext()) {
            Map.Entry<String, Word> entry = iter.next();
            // 获取key
            String partKey = entry.getKey();
            // 获取value
            Word partWord = entry.getValue();
            if(total.containsKey(partKey))
            {
                Word totalWord = total.get(partKey);
                totalWord.setTimes(totalWord.getTimes() + partWord.getTimes());
            }
            else
            {
                total.put(partKey, partWord);
            }
        }
    }

}
```

谢 谢！