

# 深圳大学实验报告

课程名称：Java 程序设计

实验项目名称：课程实验 2：类的高级应用、包、继承和接口回调

学院：计算机与软件学院

专业：数计班

指导教师：潘微科

报告人：詹耿羽 学号：2023193026 班级：数计

实验时间：2024 年 10 月 4 日（周五）-2024 年 10 月 23 日（周三）

实验报告提交时间：2024/9/27

教务部制

### 实验目的与要求:

**实验目的:** 熟悉面向对象编程中类的编写; 熟悉面向对象编程中 package, import 等语句的使用; 熟悉集合类的应用, 掌握接口的定义、实现类的编写和接口回调等技术。

### 实验要求:

#### Part 1 (25 分)

(1.1).2024 巴黎奥运会包含众多比赛项目。请通过分析, 抽象它们所共有的性质, 定义一个关于比赛项目的抽象类——Item。在报告中附上程序截图、运行结果截图(要求以中国队获得奖牌数量最多的三个比赛项目为例)和详细的文字说明。(5 分)

(1.2).编写一个运动员类——Athlete。该类包含五个成员变量 name、gender、age、item 和 medal, 分别代表一个运动员的姓名、性别、年龄、最擅长的比赛项目和在 2024 巴黎奥运会获得的奖牌数量。在该类中重写 Object 类的 toString()方法, 当调用它重写的 toString()方法时, 输出这个运动员的姓名、性别、年龄、比赛项目和奖牌数量。在报告中附上程序截图、运行结果截图(要求以 2024 巴黎奥运会中国队前三块金牌获得者为例)和详细的文字说明。(5 分)

(1.3).编写一个队列类——Queue, 用来存储 double 型数据, 队列中的数据是先进先出的。具体要求如下: 成员变量 double [] elements 用来存储 double 型数据; 成员变量 int size 用来表示存储的 double 型数据的个数; 构造方法 Queue 在初始化队列的时候, 设置队列的容量为 32; 方法 enqueue(double v)用来往队列中添加一个 double 型数据; 方法 dequeue()从队列中删除并返回一个 double 型数据; 方法 getHead()返回队列中的第一个元素; 方法 getTail()返回队列中的最后一个元素; 方法 isEmpty()判断队列是否为空; 方法 isFull()判断队列是否为满; 方法 getSize()用来返回队列的大小。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

(1.4).编写一个复数类——Complex: 成员变量包括 realPart 和 imagePart, 分别代表实数部分和虚数部分; 构造方法 Complex()用于将实数部分和虚数部分都置为 0; 构造方法 Complex(double r, double i)用于将实数部分置为 r、虚数部分置为 i; 方法 Complex complexSub(Complex c)将当前复数对象与形参复数对象相减; 方法 Complex complexMult(Complex c)将当前复数对象与形参复数对象相乘; public String toString()把当前复数对象的实数部分和虚数部分组合成 a+bi 的字符串形式。在报告中附上程序截图、运行结果截图(要求输出复数 3+5i 和复数 2+7i 相减与相乘的结果)和详细的文字说明。(5 分)

(1.5).编写一个全球计算机科学排名的类——CSRankings, 要求包含 public String toString()方法用于返回某一研究方向的相关信息(便于输出), 其他成员变量和方法自定。要求输入相应的研究方向, 能够输出相应的顶级会议名称和网址, 例如,

输入: Machine Learning & Data Mining

输出: 会议名称: ICML 网址: dblp.org/db/conf/icml/index.html

会议名称: KDD 网址: dblp.org/db/conf/kdd/index.html

会议名称: NeurIPS 网址: dblp.org/db/conf/nips/index.html

要求以 Databases、Software Engineering、The Web & Information Retrieval、Computer Graphics 为例, 在报告中附上程序截图、运行结果截图和详细的文字说明。CSRankings 介绍 <https://mp.weixin.qq.com/s/ISQklhjUKjuzJ3Y049rF7A>。(5 分)

#### Part 2 (25 分)

(2.1).编写一个计算机与软件学院类 CSSE、一个研究所/中心类 Institute 和一个教学系类 Department。CSSE 类中包含有多个 Institute 类的实例和多个 Department 类的实例。调用 CSSE 类的实例中的 getInstituteNames()和 getDepartmentNames()方法时, 能够分别

输出所有研究所/中心的名字及负责人和所有教学系的名字及系主任；调用 CSSE 类的实例中的 `getInstituteNumber()` 和 `getDepartmentNumber()` 方法时，能够分别输出研究所/中心的数量和教学系的数量。在报告中附上程序截图、运行结果截图和详细的文字说明。相关信息见 <https://csse.szu.edu.cn/pages/organization/index>（5 分）

(2.2) 根据 <https://csse.szu.edu.cn/pages/organization/index> 中的介绍，进一步完善 CSSE 类中关于“行政办公室”、“实验中心”和“期刊编辑部”的成员变量和成员方法。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

(2.3) 把 CSSE 类、Institute 类和 Department 类放进 `cn.edu.szu` 包中。编写一个测试类，在源代码中用 `import` 语句引入 `cn.edu.szu` 包中的所有类，并对它们所包含的方法进行测试。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

(2.4) 通过文字解释或程序来说明以下各种组合是否允许，如果允许表示什么意思，如果不允许是因为什么。同时，在下表中，对不允许的组合，填入 NO。（5 分）

	类	类中的成员变量	类中的成员方法	类中的构造方法	接口中的成员变量
private					
public					
final					
abstract					
static					

(2.5) 面向对象编程有三个特性（封装、继承和多态），请对“封装”、“继承”和“多态”这三个特性，通过类比、关联或演绎的方式，举一个在日常的学习生活中可以应用的例子（要求积极向上且能自圆其说）。（5 分）

### **Part 3 (30 分)**

(1). 抽象类和接口的实验。（10 分）

(i) 定义一个抽象类 `Human`：包含一个成员变量 `String name`；构造方法 `Human(String name)`，用于初始化姓名 `name`；一个抽象方法 `sayHello()`。在报告中附上程序截图和详细的文字说明。

(ii) 定义三个继承抽象类 `Human` 的类，分别命名为 `Chinese`、`Spaniard` 和 `Italian`，在这三个类中重写 `sayHello()` 方法，分别输出一句中文、西班牙语和意大利语的问候；在报告中附上程序截图、运行结果和详细的文字说明。

(iii) 定义一个测试类 `HumanTest`：创建一个包含 3 个 `Human` 对象的数组，3 个 `Human` 对象来自 `Chinese`、`Spaniard` 和 `Italian` 类，循环调用该数组中的元素的 `sayHello()` 方法。在报告中附上程序截图、运行结果和详细的文字说明。

(iv) 通过一个接口（命名为 `Human`）和三个实现类（命名为 `Chinese`、`Spaniard` 和 `Italian`）来达到如上类似的效果。在报告中附上程序截图、运行结果和详细的文字说明。

(2). 一个四维整数向量由四个分量组成。四维向量的相加、相减和点乘等价于对应四个分量的相加、相减和相乘，四维向量的内积等价于点乘所得向量中各个元素的和。比如两个四维向量 `[3,9,2,7]` 和 `[2,-8,-1,6]`，它们的和为 `[5,1,1,13]`，它们的差为 `[1,17,3,1]`，它们的点乘为 `[6,-72,-2,42]`，它们的内积为 -26。向量的模（norm）表示该向量所有分量的平方和的根，例如向量 `[3,9,2,7]` 的模为 11.96。编写一个接口 `Computable`，它具有 6 个抽象方法 `add`、`minus`、`elementwiseProduct`、`innerProduct`、`norm` 和 `compare`。编写一个 `Vector` 类，通过 `Computable` 接口实现四维向量的相加、相减、点乘、内积、模和比较（根据模的大小）。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

(3). 编写 Java 应用程序，通过字符串解析，计算字符串“上述消息提到，4 月 27 日

晚举行的深圳大学 40 周年校庆捐赠仪式暨“海岸之声”音乐晚会上，多家企业向深圳大学 40 周年校庆进行捐赠。明礼德教育科技集团有限公司向深圳大学捐赠 1000 万元；心里程控股集团向深圳大学捐赠 1 亿元；工勘岩土集团捐赠 4000 万元；正中集团捐赠 5000 万元；海岸集团捐赠 6000 万元；腾讯公益慈善基金会捐赠 2 亿元。此前，正中集团已向深大捐赠 4700 万元，海岸集团已向深大捐赠 2200 万元，腾讯创始人校友团队和腾讯公益慈善基金会已向深大捐赠 3.9 亿元。除此之外，平安集团捐赠 5000 万元，点维文化传播捐赠 1000 万元，叶晓彬校友捐赠 1000 万元，已于日前完成相关签约。”的总金额。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（5 分）

(4). 编写 Java 应用程序，随机生成一个包含有大写英文字母、小写英文字母、数字和其他字符混杂的字符串(例如 Aa123bEFGa\$aa@49023)，解析该字符串并要求按顺序输出大写英文字母（例如 AEFG）、小写英文字母（abaaa）、数字（12349023）和其他字符（\$@）。要求循环连续测试 5 次，在报告中附上程序截图、完整的运行结果截图和简要文字说明。（5 分）

(5). 编写 Java 应用程序，统计分析网页 <https://csse.szu.edu.cn/en/pages/university/index> 中关于深圳大学计算机与软件学院的重要科研平台（platform）的英文介绍中每个英文单词出现的次数（统一转为小写，不需要写爬虫，可以把整篇文章的内容当作一个字符串读入），并输出出现次数最多的 10 个英文单词（按出现次数排序从大到小排列，如次数相同则按字母顺序）。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（5 分）

报告写作。要求：主要思路有明确的说明，重点代码有详细的注释，行文逻辑清晰可读性强，报告整体写作较为专业。（20 分）

#### 说明：

- (1) 本次实验课作业满分为 100 分，占总成绩的比例 7%。
- (2) 本次实验课作业截至时间 2024 年 10 月 23 日（周三）21:59。
- (3) 报告正文：请在**指定位置填写**，本次实验**不需要单独提交源程序文件**。
- (4) 个人信息：WORD 文件名中的“姓名”、“学号”，请改为你的**姓名和学号**；实验报告的首页，请**准确填写“学院”、“专业”、“报告人”、“学号”、“班级”、“实验报告提交时间”**等信息。
- (5) 提交方式：截至时间前，请在 Blackboard 平台中提交。
- (6) 发现抄袭（包括复制&粘贴整句话、整张图），**抄袭者和被抄袭者的成绩记零分**。
- (7) 延迟提交，不得分；如有特殊情况，请于截至日期之后的**48 小时内**发邮件到 [panweike@szu.edu.cn](mailto:panweike@szu.edu.cn)，并在邮件中注明课程名称、作业名称、姓名、学号等信息，以及特殊情况的说明，我收到后会及时回复。
- (8) 期末考试阶段补交无效。

### Part 1 (25 分)

(1.1).2024 巴黎奥运会包含众多比赛项目。请通过分析，抽象它们所共有的性质，定义一个关于比赛项目的抽象类——Item。在报告中附上程序截图、运行结果截图（要求以中国队获得奖牌数量最多的三个比赛项目为例）和详细的文字说明。（5 分）

- 上网查询：

约 95,100 个结果

## 射击、乒乓球和举重

射击、乒乓球和举重各夺5金，综合表现都非常出色，其中乒乓球包揽这个项目的5枚金牌，而举重6人参赛5人夺金。

巴黎奥运综述：中国队创造多项历史 金牌与美国队并列第一

[new.qq.com/rain/a/20240811A00F7000](https://new.qq.com/rain/a/20240811A00F7000)

- 完整代码呈现：

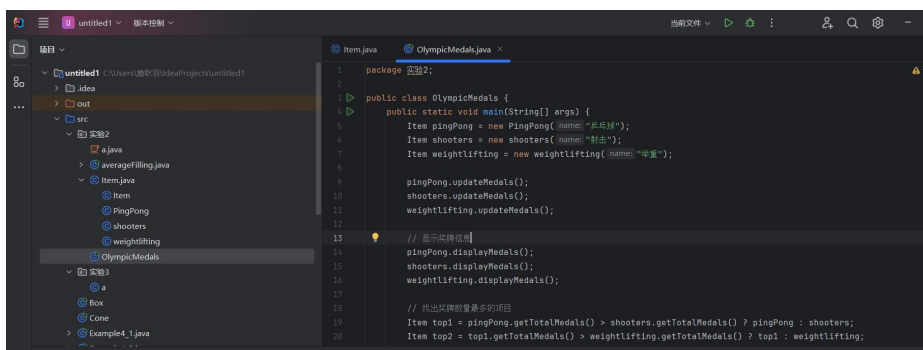
```
1. package 实验 2;
2. // 抽象类 Item
3. abstract class Item {
4.     private String name; // 比赛项目名称
5.     private int goldMedals; // 金牌数量
6.     private int silverMedals; // 银牌数量
7.     private int bronzeMedals; // 铜牌数量（在这里我都定义成私有）
8.
9.     public Item(String name) { // 构造函数
10.         this.name = name;
11.         this.goldMedals = 0;
12.         this.silverMedals = 0;
13.         this.bronzeMedals = 0;
14.     }
15.
16.     public abstract void updateMedals(); // 抽象方法，用于更新奖牌数量 (abstract)
17.
18.     public String getName() {
19.         return name; //
20.     }
21.
22.     public int getGoldMedals() {
23.         return goldMedals;
24.     }
25.
26.     public int getSilverMedals() {
27.         return silverMedals;
```

```
28.     }
29.
30.     public int getBronzeMedals() {
31.         return bronzeMedals;
32.     }
33.
34.     public void setMedals(int gold, int silver, int bronze) {
35.         this.goldMedals = gold;
36.         this.silverMedals = silver;
37.         this.bronzeMedals = bronze; // 传入修改的参数
38.     }
39.
40.     public int getTotalMedals() {
41.         return goldMedals + silverMedals + bronzeMedals; // 计算
           总金牌量
42.     }
43.
44.     public void displayMedals() {
45.         System.out.println(name + " - 金
           牌: " + goldMedals + ", 银牌: " + silverMedals + ", 铜
           牌: " + bronzeMedals);
46.     }
47. } // Item 组结束
48.
49. // 具体类乒乓球
50. class PingPong extends Item { // 由 Item 组延申而来
51.     public PingPong(String name) {
52.         super(name); // 新用法 super
53.     }
54.
55.     @Override
56.     public void updateMedals() {
57.         setMedals(5, 1, 0); // 乒乓球项目获得 5 金 1 银 0 铜
58.     }
59. }
60.
61. // 具体类射击
62. class shooters extends Item {
63.     public shooters(String name) {
64.         super(name);
65.     }
66.
67.     @Override
68.     public void updateMedals() {
```

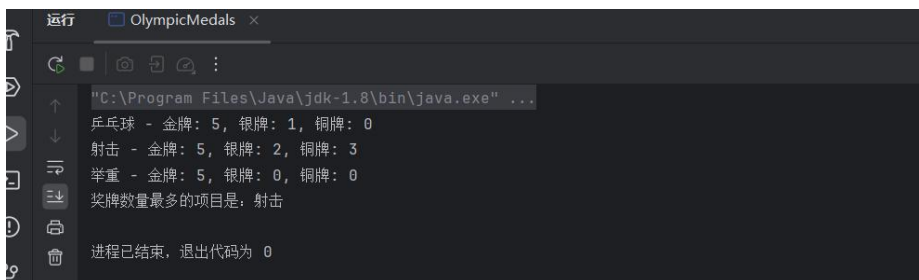
```

69.         setMedals(5, 2, 3); // 射击项目获得5金2银3铜
70.     }
71. }
72.
73. // 具体类举重
74. class weightlifting extends Item {
75.     public weightlifting(String name) {
76.         super(name);
77.     }
78.
79.     @Override
80.     public void updateMedals() {
81.         setMedals(5, 0, 0); // 举重项目获得5金0银0铜
82.     }
83. }
84.
85. // 主类OlympicMedals 在另一个java 中
86.
87.

```



- 运行结果的呈现：



- 详细文字说明：

需要注意的是，在本题目中，我使用了 `super()` 函数（通过查阅网上获得：[Java 中 super 关键字及 super\(\) 的使用 java super-CSDN 博客](#)）

- 抽象类 Item：

属性：定义了比赛项目的名称和三种奖牌数量（私有）。

构造函数：初始化项目名称，并将奖牌数量设为 0。

抽象方法：`updateMedals` 是一个抽象方法，子类必须实现该方法以更新奖牌数据。

访问方法：提供了获取名称、各类奖牌数量、总奖牌数量以及显示奖牌信息的方法。



- 继承类

继承: PingPong 类 (也可以是射击/举重) 继承自 Item 类。

构造函数: 调用父类构造函数初始化项目名称。

实现抽象方法: 通过重写 updateMedals 方法, 设置乒乓球获得的奖牌数量。

- 主类:

创建实例: 创建三个不同的 Item 实例, 分别代表乒乓球、射击和举重。

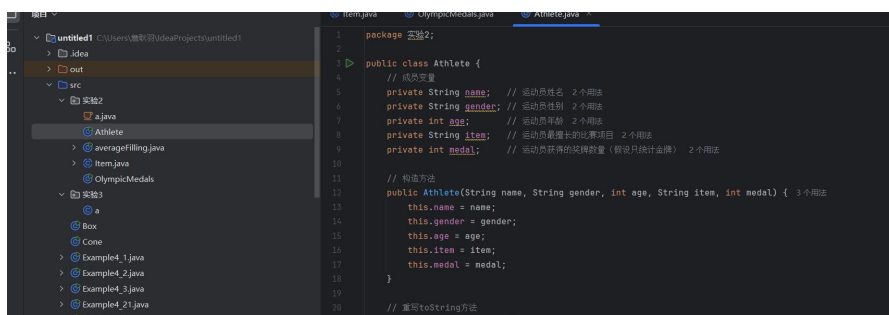
更新奖牌: 调用 updateMedals 方法更新每个项目的奖牌数量。

显示奖牌信息: 调用 displayMedals 方法输出每个项目的奖牌信息。

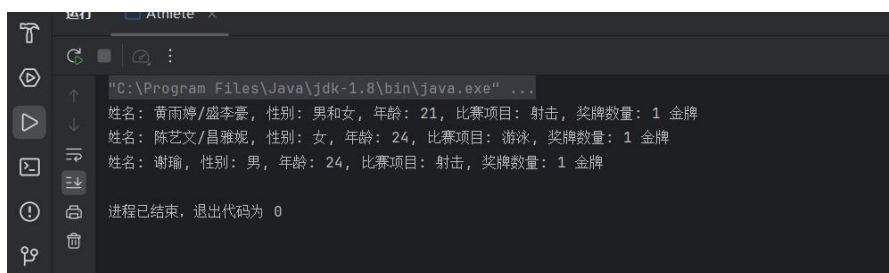
找出奖牌数量最多的项目: 通过比较总奖牌数量来找出奖牌数量最多的项目, 并打印结果。

(1.2).编写一个运动员类——Athlete。该类包含五个成员变量 name、gender、age、item 和 medal, 分别代表一个运动员的姓名、性别、年龄、最擅长的比赛项目和在 2024 巴黎奥运会获得的奖牌数量。在该类中重写 Object 类的 toString()方法, 当调用它重写的 toString()方法时, 输出这个运动员的姓名、性别、年龄、比赛项目和奖牌数量。在报告中附上程序截图、运行结果截图(要求以 2024 巴黎奥运会中国队前三块金牌获得者为例)和详细的文字说明。(5 分)

- 程序截图



- 运行结果



- 完整代码

```
1. package 实验2;
2.
3. public class Athlete {
4.     // 成员变量
5.     private String name; // 运动员姓名
6.     private String gender; // 运动员性别
7.     private int age; // 运动员年龄
8.     private String item; // 运动员最擅长的比赛项目
9.     private int medal; // 运动员获得的奖牌数量(假设只统计金牌)
10.
```



```

11.      // 构造方法
12.      public Athlete(String name, String gender, int age, String
        item, int medal) {
13.          this.name = name;
14.          this.gender = gender;
15.          this.age = age;
16.          this.item = item;
17.          this.medal = medal;
18.      }
19.
20.      // 重写 toString 方法
21.      @Override
22.      public String toString() {
23.          return "姓名: " + name + ", 性别: " + gender + ", 年
        龄: " + age +
24.              ", 比赛项目: " + item + ", 奖牌数
        量: " + medal + " 金牌";
25.      }
26.
27.      // 主方法, 用于创建实例并展示运动员信息
28.      public static void main(String[] args) {
29.          // 2024 年巴黎奥运会中国队前三块金牌获得者 (假设)
30.          Athlete athlete1 = new Athlete("黄雨婷/盛李豪", "男和女
        ", 21, "射击", 1);
31.          Athlete athlete2 = new Athlete("陈艺文/昌雅妮", "女
        ", 24, "游泳", 1);
32.          Athlete athlete3 = new Athlete("谢瑜", "男", 24, "射击
        ", 1);
33.
34.          // 输出每个运动员的信息
35.          System.out.println(athlete1);
36.          System.out.println(athlete2);
37.          System.out.println(athlete3);
38.      }
39.  }
40.

```

- 文字说明:

类声明: 定义了一个公共类 `Athlete`。

私有属性: 包含运动员的姓名、性别、年龄、最擅长的比赛项目和获得的金牌数量。这些属性被设置为私有, 以保护数据的封装性。

构造函数: 用于初始化 `Athlete` 对象的属性。通过参数传递运动员的姓名、性别、年龄、项目和奖牌数量。

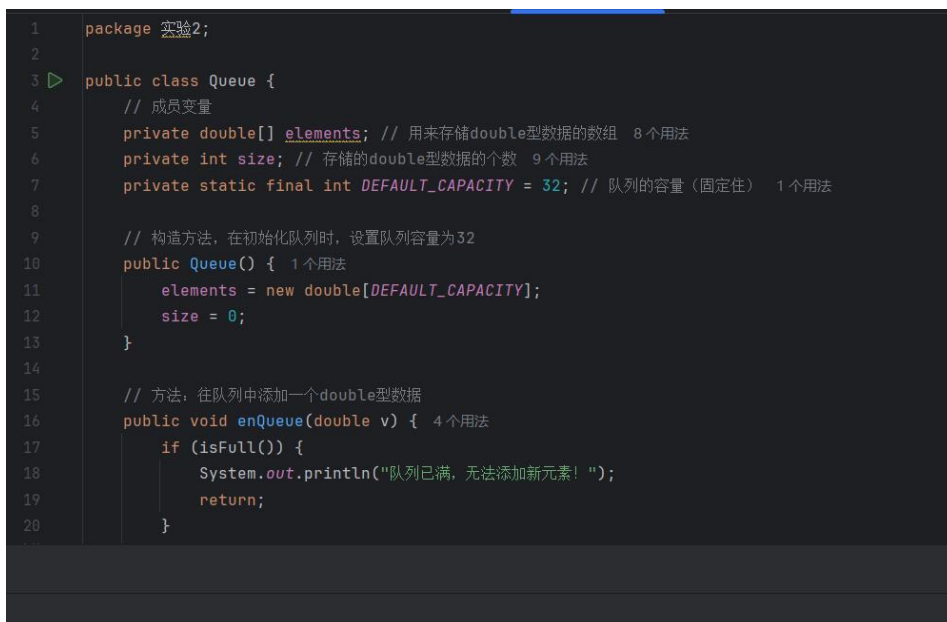
`toString` 方法: 重写了 `Object` 类的 `toString` 方法, 以便以更易读的格式输出运动员的信息。返回的字符串包含运动员的所有相关信息。

创建实例：在主方法中，创建了三个 `Athlete` 对象，分别表示假设的三位运动员及其信息。

输出信息：使用 `System.out.println` 打印每位运动员的详细信息，通过调用重写的 `toString` 方法实现。

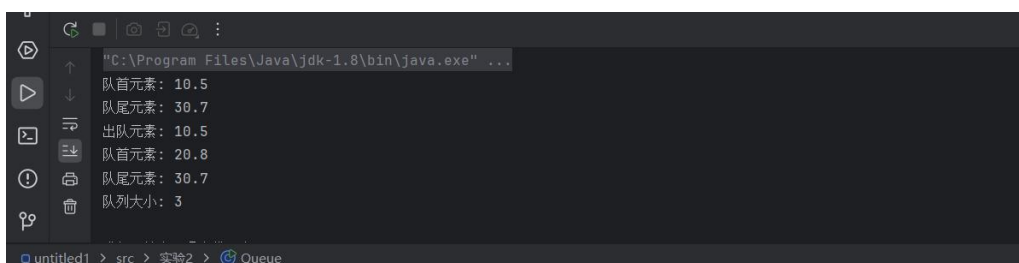
(1.3).编写一个队列类——`Queue`，用来存储 `double` 型数据，队列中的数据是先进先出的。具体要求如下：成员变量 `double [] elements` 用来存储 `double` 型数据；成员变量 `int size` 用来表示存储的 `double` 型数据的个数；构造方法 `Queue` 在初始化队列的时候，设置队列的容量为 32；方法 `enqueue(double v)` 用来往队列中添加一个 `double` 型数据；方法 `deQueue()` 从队列中删除并返回一个 `double` 型数据；方法 `getHead()` 返回队列中的第一个元素；方法 `getTail()` 返回队列中的最后一个元素；方法 `isEmpty()` 判断队列是否为空；方法 `isFull()` 判断队列是否为满；方法 `getSize()` 用来返回队列的大小。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

- 程序截图



```
1 package 实验2;
2
3 public class Queue {
4     // 成员变量
5     private double[] elements; // 用来存储double型数据的数组 8个用法
6     private int size; // 存储的double型数据的个数 9个用法
7     private static final int DEFAULT_CAPACITY = 32; // 队列的容量（固定住） 1个用法
8
9     // 构造方法，在初始化队列时，设置队列容量为32
10    public Queue() { 1个用法
11        elements = new double[DEFAULT_CAPACITY];
12        size = 0;
13    }
14
15    // 方法：往队列中添加一个double型数据
16    public void enqueue(double v) { 4个用法
17        if (isFull()) {
18            System.out.println("队列已满，无法添加新元素！");
19            return;
20        }
21    }
```

- 运行结果



```
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
队首元素: 10.5
队尾元素: 30.7
出队元素: 10.5
队首元素: 20.8
队尾元素: 30.7
队列大小: 3
```

- 完整代码

```
1. package 实验2;
2.
3. public class Queue {
4.     // 成员变量
5.     private double[] elements; // 用来存储double型数据的数组
6.     private int size; // 存储的double型数据的个数
```

```
7.     private static final int DEFAULT_CAPACITY = 32; // 队列的容量（固定住）
8.
9.     // 构造方法，在初始化队列时，设置队列容量为 32
10.    public Queue() {
11.        elements = new double[DEFAULT_CAPACITY];
12.        size = 0;
13.    }
14.
15.    // 方法：往队列中添加一个 double 型数据
16.    public void enqueue(double v) {
17.        if (isFull()) {
18.            System.out.println("队列已满，无法添加新元素！");
19.            return;
20.        }
21.        elements[size] = v;
22.        size++;
23.    }
24.
25.    // 方法：从队列中删除并返回一个 double 型数据（队首元素）
26.    public double dequeue() {
27.        if (isEmpty()) {
28.            System.out.println("队列为空，无法删除元素！");
29.            return -1;
30.        }
31.        double value = elements[0];
32.        // 将剩余元素前移
33.        for (int i = 1; i < size; i++) {
34.            elements[i - 1] = elements[i];
35.        }
36.        size--;
37.        return value;
38.    }
39.
40.    // 方法：返回队列中的第一个元素
41.    public double getHead() {
42.        if (isEmpty()) {
43.            System.out.println("队列为空，无法获取队首元素！");
44.            return -1;
45.        }
46.        return elements[0]; // gettop
47.    }
48.
49.    // 方法：返回队列中的最后一个元素
```

```
50.     public double getTail() {
51.         if (isEmpty()) {
52.             System.out.println("队列为空，无法获取队尾元素！");
53.             return -1;
54.         }
55.         return elements[size - 1];
56.     }
57.
58.     // 方法：判断队列是否为空
59.     public boolean isEmpty() {
60.         return size == 0;
61.     }
62.
63.     // 方法：判断队列是否为满
64.     public boolean isFull() {
65.         return size == elements.length;
66.     }
67.
68.     // 方法：返回队列的大小
69.     public int getSize() {
70.         return size;
71.     }
72.
73.     // 主方法用于测试队列
74.     public static void main(String[] args) {
75.         Queue queue = new Queue();
76.
77.         // 添加元素到队列中
78.         queue.enqueue(10.5);
79.         queue.enqueue(20.8);
80.         queue.enqueue(30.6);
81.         queue.enqueue(30.7);
82.
83.         // 显示队列的队首和队尾元素
84.         System.out.println("队首元素：" + queue.getHead());
85.         System.out.println("队尾元素：" + queue.getTail());
86.
87.         // 从队列中删除元素
88.         System.out.println("出队元素：" + queue.dequeue());
89.
90.         // 显示队列状态
91.         System.out.println("队首元素：" + queue.getHead());
92.         System.out.println("队尾元素：" + queue.getTail());
93.     }
```

```

94.          // 显示队列大小
95.          System.out.println("队列大小: " + queue.getSize());
96.      }
97.  }

```

• 文字说明

类声明：定义了一个公共类 `Queue`，表示队列。

`elements`：用于存储队列中所有的 `double` 类型元素。

`size`：当前队列中元素的数量。

`DEFAULT_CAPACITY`：队列的默认容量，设置为 32。

构造函数：初始化队列，将 `elements` 数组创建为指定的默认容量，并将 `size` 设置为 0。

`enqueue` 方法：向队列中添加一个 `double` 类型的元素。如果队列已满，则输出提示信息；否则，将元素添加到数组中并增加 `size`。

`dequeue` 方法：从队列中删除并返回队首元素。如果队列为空，则输出提示信息并返回 -1。否则，保存队首元素，然后将后续元素前移，最后减少 `size`。

`getHead` 方法：返回队首元素。如果队列为空，则输出提示信息并返回 -1。

`getTail` 方法：返回队尾元素。如果队列为空，则输出提示信息并返回 -1。

`isEmpty` 方法：返回队列是否为空的布尔值。

`isFull` 方法：返回队列是否已满的布尔值。

`getSize` 方法：返回队列中当前元素的数量。

测试代码：

在主方法中创建 `Queue` 的实例，并进行一系列操作：

添加几个 `double` 元素到队列。

打印队首和队尾元素。

从队列中删除一个元素并打印出队首和队尾的变化。

打印当前队列的大小。

(1.4).编写一个复数类——`Complex`：成员变量包括 `realPart` 和 `imagePart`，分别代表实数部分和虚数部分；构造方法 `Complex()` 用于将实数部分和虚数部分都置为 0；构造方法 `Complex(double r, double i)` 用于将实数部分置为 `r`、虚数部分置为 `i`；方法 `Complex complexSub(Complex c)` 将当前复数对象与形参复数对象相减；方法 `Complex complexMult(Complex c)` 将当前复数对象与形参复数对象相乘；`public String toString()` 把当前复数对象的实数部分和虚数部分组合成 `a+bi` 的字符串形式。在报告中附上程序截图、运行结果截图（要求输出复数 `3+5i` 和复数 `2+7i` 相减与相乘的结果）和详细的文字说明。（5 分）

• 完整代码

```

1.  package 实验2;
2.
3.  public class Complex {
4.      // 成员变量，实部和虚部
5.      private double realPart; // 实数部分
6.      private double imagePart; // 虚数部分
7.
8.      // 无参构造方法，将实数部分和虚数部分都置为0

```

```

9.     public Complex() {
10.         this.realPart = 0;
11.         this.imagePart = 0;
12.     }
13.
14.     // 带参构造方法，实数部分置为r，虚数部分置为i
15.     public Complex(double r, double i) {
16.         this.realPart = r;
17.         this.imagePart = i;
18.     }
19.
20.     // 复数相减方法：当前复数对象与形参复数相减
21.     public Complex complexSub(Complex c) {
22.         double real = this.realPart - c.realPart;
23.         double imag = this.imagePart - c.imagePart;
24.         return new Complex(real, imag); // 返回相减后的新复数对
    象
25.     }
26.
27.     // 复数相乘方法：当前复数对象与形参复数相乘
28.     public Complex complexMult(Complex c) {
29.         double real = this.realPart * c.realPart - this.imagePa
    rt * c.imagePart;
30.         double imag = this.realPart * c.imagePart + this.imageP
    art * c.realPart;
31.         return new Complex(real, imag); // 返回相乘后的新复数对
    象
32.     }
33.
34.     // 重写toString()方法，将复数表示为字符串形式 a+bi
35.     @Override
36.     public String toString() {
37.         if (imagePart >= 0) {
38.             return realPart + " + " + imagePart + "i";
39.         } else {
40.             return realPart + " - " + (-imagePart) + "i";
41.         }
42.     }
43.
44.     // 主方法用于测试复数相减和相乘
45.     public static void main(String[] args) {
46.         // 创建两个复数对象
47.         Complex c1 = new Complex(3, 5); // 复数 3 + 5i
48.         Complex c2 = new Complex(2, 7); // 复数 2 + 7i

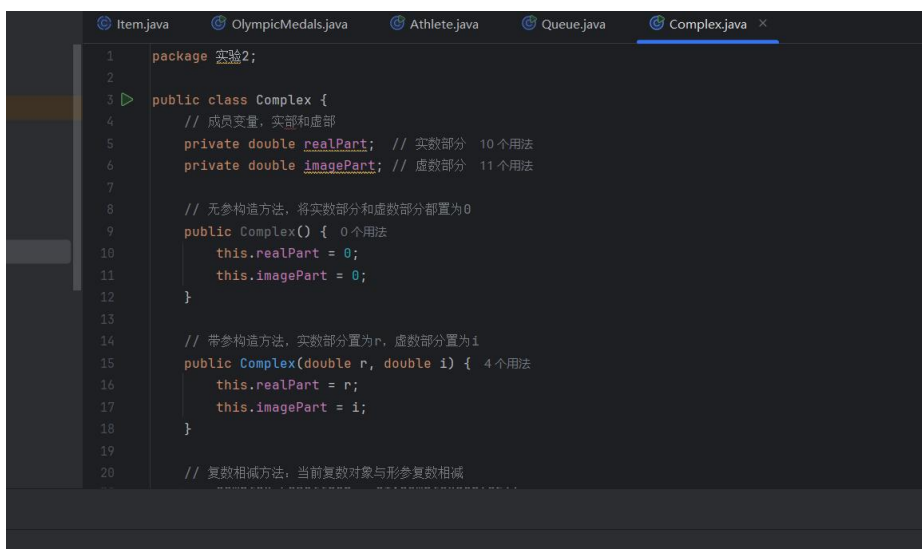
```

```

49.
50.         // 复数相减
51.         Complex resultSub = c1.complexSub(c2);
52.         System.out.println("复数相减结果: " + resultSub);
53.
54.         // 复数相乘
55.         Complex resultMult = c1.complexMult(c2);
56.         System.out.println("复数相乘结果: " + resultMult);
57.     }
58. }
59.

```

#### • 程序截图

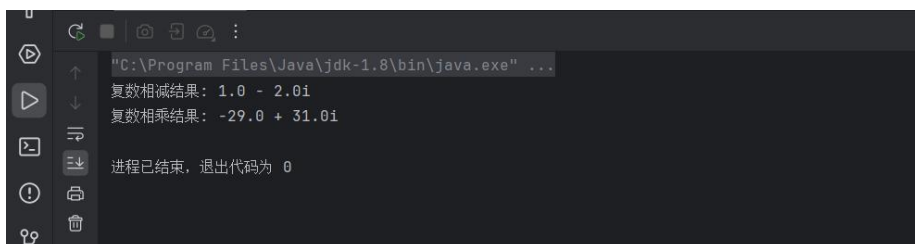


```

1 package 实验2;
2
3 public class Complex {
4     // 成员变量，实部和虚部
5     private double realPart; // 实数部分 10 个用法
6     private double imagePart; // 虚数部分 11 个用法
7
8     // 无参构造方法，将实数部分和虚数部分都置为0
9     public Complex() { 0 个用法
10         this.realPart = 0;
11         this.imagePart = 0;
12     }
13
14     // 带参构造方法，实数部分置为r，虚数部分置为i
15     public Complex(double r, double i) { 4 个用法
16         this.realPart = r;
17         this.imagePart = i;
18     }
19
20     // 复数相减方法：当前复数对象与形参复数相减

```

#### • 运行结果



```

"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
复数相减结果: 1.0 - 2.0i
复数相乘结果: -29.0 + 31.0i
进程已结束，退出代码为 0

```

#### • 文字解释

**Complex** 类有两个私有成员变量：**realPart** 和 **imagePart**，分别用于存储复数的实部和虚部。

**无参构造方法**：创建一个复数对象时，如果不提供参数，则将实部和虚部都设为 0，即  $0 + 0i$ 。

**带参构造方法**：允许用户在创建复数对象时指定实部和虚部的值。

**complexSub** 方法接受一个 **Complex** 对象作为参数，计算当前复数与参数复数的差，并返回一个新的 **Complex** 对象，表示相减结果。

**complexMult** 方法实现复数相乘的公式，计算当前复数与参数复数的积，并返回一个新的 **Complex** 对象，表示相乘结果。

重写了 **toString()** 方法，以便在打印复数对象时可以得到易读的格式（如  $a + bi$  或



a - bi)。

在 main 方法中，创建了两个复数对象 c1 和 c2。然后调用 complexSub 和 complexMult 方法，分别进行相减和相乘操作，并输出结果。

(1.5).编写一个全球计算机科学排名的类——CSRankings，要求包含 public String toString()方法用于返回某一研究方向的相关信息（便于输出），其他成员变量和方法自定。要求输入相应的研究方向，能够输出相应的顶级会议名称和网址，例如，

输入：Machine Learning & Data Mining

输出：会议名称：ICML 网址：dblp.org/db/conf/icml/index.html

会议名称：KDD 网址：dblp.org/db/conf/kdd/index.html

会议名称：NeurIPS 网址：dblp.org/db/conf/nips/index.html

要求以 Databases、Software Engineering、The Web & Information Retrieval、Computer Graphics 为例，在报告中附上程序截图、运行结果截图和详细的文字说明。CSRankings 介绍 <https://mp.weixin.qq.com/s/ISQklhjUKjuzJ3Y049rF7A>。（5 分）

• 完整代码

```
1. package 实验2;  
2. import java.util.HashMap;  
3. import java.util.Map;  
4. import java.util.Scanner;  
5.  
6. public class CSRanking {  
7.     private Map<String, String[]> researchFields;  
8.  
9.     public CSRanking() {  
10.         researchFields = new HashMap<>();  
11.         initializeResearchFields();  
12.     }  
13.  
14.     private void initializeResearchFields() {  
15.         // 添加各个研究方向及其对应的会议名称和网址  
16.         researchFields.put("Machine Learning & Data Mining", new  
17.             String[]{  
18.                 "ICML: dblp.org/db/conf/icml/index.html",  
19.                 "KDD: dblp.org/db/conf/kdd/index.html",  
20.                 "NeurIPS: dblp.org/db/conf/nips/index.html"  
21.             });  
22.         researchFields.put("Databases", new String[]{  
23.             "VLDB: dblp.org/db/conf/vldb/index.html",  
24.             "SIGMOD: dblp.org/db/conf/sigmod/index.html",  
25.             "ICDE: dblp.org/db/conf/icde/index.html"  
26.         });  
27.  
28.         researchFields.put("Software Engineering", new String[]
```

```

    {
29.         "ICSE: dblp.org/db/conf/icse/index.html",
30.         "FSE: dblp.org/db/conf/sigsoft/index.html",
31.         "ASE: dblp.org/db/conf/kbse/index.html"
32.     });
33.
34.     researchFields.put("The Web & Information Retrieval", new String[]{
35.         "WWW: dblp.org/db/conf/www/index.html",
36.         "SIGIR: dblp.org/db/conf/sigir/index.html"
37.     });
38.
39.     researchFields.put("Computer Graphics", new String[]{
40.         "SIGGRAPH: dblp.org/db/conf/siggraph/index.html",
41.         "SIGGRAPH ASIA: dblp.org/db/conf/siggrapha/index.html"
42.     });
43.     }// 加入对应关系
44.
45.     public String getResearchInfo(String field) {
46.         StringBuilder result = new StringBuilder();
47.         String[] conferences = researchFields.get(field);
48.
49.         if (conferences != null) {
50.             for (String conference : conferences) {
51.                 String[] parts = conference.split(": ");
52.                 result.append("会议名称: ").append(parts[0]).append(" 网址: ").append(parts[1]).append("\n");
53.             }
54.         } else {
55.             result.append("未找到该研究方向的相关信息。");
56.         }
57.
58.         return result.toString();
59.     }
60.
61.     public static void main(String[] args) {
62.         CSRanking rankings = new CSRanking();
63.         Scanner scanner = new Scanner(System.in);
64.
65.         System.out.println("请输入研究方向（如：Machine Learning & Data Mining）：");

```

```

66.         String inputField = scanner.nextLine();
67.
68.         // 输出相应的会议信息
69.         String output = rankings.getResearchInfo(inputField);
70.         System.out.println(output);
71.
72.         scanner.close();
73.     }
74. }
75.

```

- 程序截图

```

1 package 实验2;
2 import java.util.HashMap;
3 import java.util.Map;
4 import java.util.Scanner;
5
6 public class CSRanking {
7     private Map<String, String[]> researchFields; // 1个用法
8
9     public CSRanking() { // 1个用法
10         researchFields = new HashMap<>();
11         initializeResearchFields();
12     }
13
14     private void initializeResearchFields() { // 1个用法
15         // 添加各个研究方向及其对应的会议名称和网址
16         researchFields.put("Machine Learning & Data Mining", new String[]{
17             "ICML: dblp.org/db/conf/icml/index.html",
18             "KDD: dblp.org/db/conf/kdd/index.html",
19             "NeurIPS: dblp.org/db/conf/nips/index.html"
20         });
21
22         researchFields.put("Databases", new String[]{
23             "VLDB: dblp.org/db/conf/vldb/index.html",
24             "SIGMOD: dblp.org/db/conf/sigmod/index.html",
25             "ICDE: dblp.org/db/conf/icde/index.html"
26         });
27
28         researchFields.put("Software Engineering", new String[]{
29             "ICSE: dblp.org/db/conf/icse/index.html",
30             "FSE: dblp.org/db/conf/sigsoft/index.html",
31             "ASE: dblp.org/db/conf/ase/index.html"
32         });
33     }
34 }

```

- 运行结果（这里我以输入 Databases 为例子）

```

运行 CSRanking
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
请输入研究方向 (如, Machine Learning & Data Mining):
Databases
会议名称: VLDB 网址: dblp.org/db/conf/vldb/index.html
会议名称: SIGMOD 网址: dblp.org/db/conf/sigmod/index.html
会议名称: ICDE 网址: dblp.org/db/conf/icde/index.html

```

- 文字解释

**public class CSRanking:** 定义一个公共类 CSRanking。

**private Map<String, String[]> researchFields;** 声明一个私有的 HashMap, 用于存储研究方向及其对应的会议名称和网址。键是研究方向的字符串, 值是包含会议信息的字符串数组。

**public CSRanking():** 构造方法, 在创建 CSRanking 对象时被调用。

**initializeResearchFields();** 调用 initializeResearchFields 方法来初始化 researchFields 字典。

**private void initializeResearchFields():** 此方法用于填充 researchFields 字典。

**public String getResearchInfo(String field):** 根据给定的研究方向返回相关会议信息的字符串。

**public static void main(String[] args):** 程序的入口点。（主方法）

## Part 2 (25 分)

(2.1).编写一个计算机与软件学院类 CSSE、一个研究所/中心类 Institute 和一个教学系类 Department。CSSE 类中包含有多个 Institute 类的实例和多个 Department 类的实例。调用 CSSE 类的实例中的 getInstituteNames()和 getDepartmentNames()方法时，能够分别输出所有研究所/中心的名字及负责人和所有教学系的名字及系主任；调用 CSSE 类的实例中的 getInstituteNumber()和 getDepartmentNumber()方法时，能够分别输出研究所/中心的数量和教学系的数量。在报告中附上程序截图、运行结果截图和详细的文字说明。相关信息见 <https://csse.szu.edu.cn/pages/organization/index> (5 分)

- 完整代码

```
1. package 实验2;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. // Institute 类
7. class Institute {
8.     private String name;
9.     private String director;
10.
11.     public Institute(String name, String director) {
12.         this.name = name;
13.         this.director = director;
14.     }
15.
16.     public String getName() {
17.         return name;
18.     }
19.
20.     public String getDirector() {
21.         return director;
22.     }
23. }
24.
25. // Department 类
26. class Department {
27.     private String name;
28.     private String head;
29.
30.     public Department(String name, String head) {
31.         this.name = name;
32.         this.head = head;
33.     }
34.
35.     public String getName() {
```

```
36.         return name;
37.     }
38.
39.     public String getHead() {
40.         return head;
41.     }
42. }
43.
44. // CSSE 类
45. class CSSE {
46.     private List<Institute> institutes;
47.     private List<Department> departments;
48.
49.     public CSSE() {
50.         institutes = new ArrayList<>();
51.         departments = new ArrayList<>();
52.
53.         // 添加研究所/中心
54.         institutes.add(new Institute("高性能计算研究所", "陈国良"));
55.         institutes.add(new Institute("大数据技术与应用研究所", "黄哲学"));
56.         institutes.add(new Institute("未来媒体技术与计算研究所", "江建民"));
57.         institutes.add(new Institute("软件工程研究中心", "明仲"));
58.         institutes.add(new Institute("可视计算研究中心", "黄惠"));
59.         institutes.add(new Institute("智能服务计算中心", "张良杰"));
60.         institutes.add(new Institute("计算机视觉研究所", "文振焜"));
61.         institutes.add(new Institute("智能技术与系统集成研究所", "未知"));
62.         institutes.add(new Institute("网络与信息安全研究所", "未知"));
63.
64.         // 添加教学系
65.         departments.add(new Department("计算机科学与技术系", "潘微科"));
66.         departments.add(new Department("软件工程系", "张良杰"));
67.         departments.add(new Department("人工智能系", "王熙熙"));
68.     }
69. }
```

```
70.     public List<String> getInstituteNames() {
71.         List<String> names = new ArrayList<>();
72.         for (Institute institute : institutes) {
73.             names.add(institute.getName() + " - " + institute.g
74.                 etDirector());
75.         }
76.         return names;
77.     }
78.     public List<String> getDepartmentNames() {
79.         List<String> names = new ArrayList<>();
80.         for (Department department : departments) {
81.             names.add(department.getName() + " - " + department
82.                 .getHead());
83.         }
84.         return names;
85.     }
86.     public int getInstituteNumber() {
87.         return institutes.size();
88.     }
89.
90.     public int getDepartmentNumber() {
91.         return departments.size();
92.     }
93. }
94.
95. // 主方法
96. public class Main {
97.     public static void main(String[] args) {
98.         CSSE csse = new CSSE();
99.
100.        // 输出研究所/中心信息
101.        System.out.println("研究所/中心:");
102.        List<String> names = csse.getInstituteNames();
103.        System.out.println(names);
104.
105.        System.out.println("数
106.            量: " + csse.getInstituteNumber());
107.        // 输出教学系信息
108.        System.out.println("\n 教学系:");
109.        for (String name : csse.getDepartmentNames()) {
110.            System.out.println(name);
```

```

111.         }
112.         System.out.println("数
    量: " + csse.getDepartmentNumber());
113.     }
114. }
115.

```

#### • 程序运行截图

```

1 package 实验2;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 // Institute 类
7 class Institute { 11个用法
8     private String name; 2个用法
9     private String director; 2个用法
10
11     public Institute(String name, String director) { 9个用法
12         this.name = name;
13         this.director = director;
14     }
15
16     public String getName() {
17         return name;
18     }
19
20     public String getDirector() { 1个用法
21         return director;
22     }
23 }
24
25 // Department 类
26 class Department { 5个用法
27     private String name; 2个用法
28     private String head; 2个用法
29
30     public Department(String name, String head) { 3个用法
31         this.name = name;
32         this.head = head;
33     }
34 }

```

#### • 运行结果（我分了两个格式，一个是一列完整输出，一个是分别输出）

```

研究系/中心:
【高性能计算研究所 - 陈国良, 大数据技术与应用研究所 - 曹哲宇, 未来媒体技术与计算研究所 - 江建民, 软件工程研究中心 - 明坤, 可视计算研究中心 - 曹惠, 智能服务计算中心 - 张良杰, 计算机视觉研究所 - 文景煜, 智能技术与
数量: 9

数学系:
计算机科学与技术系 - 潘微科
软件工程系 - 张良杰
人工智能系 - 王鹏翔
数量: 3

进程已结束, 退出代码为 0

```

#### • 文字说明



（有些实验室网址打不



开，查看不了负责人，以未知代替）

Institute 类：

属性：name 和 director，分别表示研究所的名称和院长。

构造函数：用于初始化这两个属性。

Getter 方法：提供对属性的访问。

Department 类：

属性：name 和 head，分别表示系的名称和系主任。

构造函数：用于初始化这两个属性。

Getter 方法：提供对属性的访问。

CSSE 类：

属性：institutes 和 departments，分别是存储研究所和教学系的列表。

构造函数：初始化研究所和教学系，并添加一些实例。

方法：

getInstituteNames()：返回所有研究所的名称和院长信息。

getDepartmentNames()：返回所有教学系的名称和系主任信息。

getInstituteNumber()：返回研究所的数量。

getDepartmentNumber()：返回教学系的数量。

Main 类：

主方法 main 创建一个 CSSE 实例，并打印研究所及教学系的信息，包括它们的数量。

(2.2)根据 <https://csse.szu.edu.cn/pages/organization/index> 中的介绍，进一步完善 CSSE 类中关于“行政办公室”、“实验中心”和“期刊编辑部”的成员变量和成员方法。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

由于行政办公公司内容信息过多，我可能对人员进行了些许的忽略。

• 完整代码

```
1. package 实验2;  
2. import java.util.ArrayList;  
3. import java.util.List;  
4.  
5. // Institute 类  
6. class Institute {  
7.     private String name;  
8.     private String director;  
9.  
10.    public Institute(String name, String director) {  
11.        this.name = name;  
12.        this.director = director;  
13.    }  
14.  
15.    public String getName() {  
16.        return name;  
17.    }  
18.
```

```
19.     public String getDirector() {
20.         return director;
21.     }
22. }
23.
24. // Department 类
25. class Department {
26.     private String name;
27.     private String head;
28.
29.     public Department(String name, String head) {
30.         this.name = name;
31.         this.head = head;
32.     }
33.
34.     public String getName() {
35.         return name;
36.     }
37.
38.     public String getHead() {
39.         return head;
40.     }
41. }
42.
43. // 实验中心类
44. class LabCenter {
45.     private String name;
46.     private String head;
47.
48.     public LabCenter(String name, String head) {
49.         this.name = name;
50.         this.head = head;
51.     }
52.
53.     public String getName() {
54.         return name;
55.     }
56.
57.     public String getHead() {
58.         return head;
59.     }
60. }
61.
62. // 行政办公室类
```

```
63. class Office {
64.     private String name;
65.     private String director;
66.
67.     public Office(String name, String director) {
68.         this.name = name;
69.         this.director = director;
70.     }
71.
72.     public String getName() {
73.         return name;
74.     }
75.
76.     public String getDirector() {
77.         return director;
78.     }
79. }
80.
81. // 期刊编辑部类
82. class Journal {
83.     private String name;
84.     private String editor;
85.
86.     public Journal(String name, String editor) {
87.         this.name = name;
88.         this.editor = editor;
89.     }
90.
91.     public String getName() {
92.         return name;
93.     }
94.
95.     public String getEditor() {
96.         return editor;
97.     }
98. }
99.
100. // CSSE 类
101. class CSSE {
102.     private List<Institute> institutes;
103.     private List<Department> departments;
104.     private List<LabCenter> labCenters;
105.     private List<Office> offices;
106.     private List<Journal> journals;
```

```
107.
108.     public CSSE() {
109.         institutes = new ArrayList<>();
110.         departments = new ArrayList<>();
111.         labCenters = new ArrayList<>();
112.         offices = new ArrayList<>();
113.         journals = new ArrayList<>();
114.
115.         // 添加研究所/中心
116.         institutes.add(new Institute("高性能计算研究所", "陈国良
            "));
117.         institutes.add(new Institute("大数据技术与应用研究所", "
            黄哲学"));
118.         institutes.add(new Institute("未来媒体技术与计算研究所
            ", "江建民"));
119.         institutes.add(new Institute("软件工程研究中心", "明仲
            "));
120.         institutes.add(new Institute("可视计算研究中心", "黄惠
            "));
121.         institutes.add(new Institute("智能服务计算中心", "张良杰
            "));
122.         institutes.add(new Institute("计算机视觉研究所", "文振焜
            "));
123.         institutes.add(new Institute("智能技术与系统集成研究所
            ", "未知"));
124.         institutes.add(new Institute("网络与信息安全研究所", "未
            知"));
125.
126.         // 添加教学系
127.         departments.add(new Department("计算机科学与技术系", "潘
            微科"));
128.         departments.add(new Department("软件工程系", "张良杰"));
129.         departments.add(new Department("人工智能系", "王熙熙"));
130.
131.         // 添加实验中心
132.         labCenters.add(new LabCenter("计算机实验教学中心", "黄惠
            "));
133.         labCenters.add(new LabCenter("网络工程虚拟仿真实验教学中
            心", "朱安民"));
134.
135.         // 添加行政办公室（包含各负责人）
136.         offices.add(new Office("党务工作", "杨国洪"));
137.         offices.add(new Office("教学业务", "胡沛"));
138.         offices.add(new Office("实验中心", "林佳利"));
```

```
139.         offices.add(new Office("辅导员", "黄晓聪"));
140.         offices.add(new Office("科研外事", "何文锋"));
141.         offices.add(new Office("综合业务", "刘晔"));
142.
143.         // 添加期刊编辑部
144.         journals.add(new Journal("期刊编辑部", "未知"));
145.     }
146.
147.     // 获取所有研究所/中心的名字和负责人
148.     public List<String> getInstituteNames() {
149.         List<String> names = new ArrayList<>();
150.         for (Institute institute : institutes) {
151.             names.add(institute.getName() + " - " + institute.getDirector());
152.         }
153.         return names;
154.     }
155.
156.     // 获取所有教学系的名字和系主任
157.     public List<String> getDepartmentNames() {
158.         List<String> names = new ArrayList<>();
159.         for (Department department : departments) {
160.             names.add(department.getName() + " - " + department.getHead());
161.         }
162.         return names;
163.     }
164.
165.     // 获取所有实验中心的名字和负责人
166.     public List<String> getLabCenterNames() {
167.         List<String> names = new ArrayList<>();
168.         for (LabCenter labCenter : labCenters) {
169.             names.add(labCenter.getName() + " - " + labCenter.getHead());
170.         }
171.         return names;
172.     }
173.
174.     // 获取所有行政办公室的名字和负责人
175.     public List<String> getOfficeNames() {
176.         List<String> names = new ArrayList<>();
177.         for (Office office : offices) {
178.             names.add(office.getName() + " - " + office.getDirector());
179.         }
180.         return names;
181.     }
182. }
```

```
179.     }
180.     return names;
181. }
182.
183. // 获取所有期刊编辑部的名字和负责人
184. public List<String> getJournalNames() {
185.     List<String> names = new ArrayList<>();
186.     for (Journal journal : journals) {
187.         names.add(journal.getName() + " - " + journal.getEditor());
188.     }
189.     return names;
190. }
191.
192. // 获取研究所/中心数量
193. public int getInstituteNumber() {
194.     return institutes.size();
195. }
196.
197. // 获取教学系数量
198. public int getDepartmentNumber() {
199.     return departments.size();
200. }
201.
202. // 获取实验中心数量
203. public int getLabCenterNumber() {
204.     return labCenters.size();
205. }
206.
207. // 获取行政办公室数量
208. public int getOfficeNumber() {
209.     return offices.size();
210. }
211.
212. // 获取期刊编辑部数量
213. public int getJournalNumber() {
214.     return journals.size();
215. }
216. }
217.
218. // 主方法
219. public class Main {
220.     public static void main(String[] args) {
221.         CSSE csse = new CSSE();
```

```

222.
223.         // 输出研究所/中心信息
224.         System.out.println("研究所/中心:");
225.         for (String name : csse.getInstituteNames()) {
226.             System.out.println(name);
227.         }
228.         System.out.println("数
    量: " + csse.getInstituteNumber());
229.
230.         // 输出教学系信息
231.         System.out.println("\n 教学系:");
232.         for (String name : csse.getDepartmentNames()) {
233.             System.out.println(name);
234.         }
235.         System.out.println("数
    量: " + csse.getDepartmentNumber());
236.
237.         // 输出实验中心信息
238.         System.out.println("\n 实验中心:");
239.         for (String name : csse.getLabCenterNames()) {
240.             System.out.println(name);
241.         }
242.         System.out.println("数
    量: " + csse.getLabCenterNumber());
243.
244.         // 输出行政办公室信息
245.         System.out.println("\n 行政办公室:");
246.         for (String name : csse.getOfficeNames()) {
247.             System.out.println(name);
248.         }
249.         System.out.println("数量: " + csse.getOfficeNumber());
250.
251.         // 输出期刊编辑部信息
252.         System.out.println("\n 期刊编辑部:");
253.         for (String name : csse.getJournalNames()) {
254.             System.out.println(name);
255.         }
256.         System.out.println("数量: " + csse.getJournalNumber());
257.     }
258. }

```

• 程序截图



```

package 实验2;
import java.util.ArrayList;
import java.util.List;

// Institute 类
class Institute { 11个用法
    private String name; 2个用法
    private String director; 2个用法

    public Institute(String name, String director) { 9个用法
        this.name = name;
        this.director = director;
    }

    public String getName() {
        return name;
    }

    public String getDirector() { 1个用法
        return director;
    }
}

// Department 类
class Department { 5个用法
    private String name; 2个用法
    private String head; 2个用法

    public Department(String name, String head) { 3个用法
        this.name = name;
        this.head = head;
    }
}

```

## • 运行结果

```

运行 Main x
网络与信息安全研究所 - 未知
数量: 9

教学系:
  计算机科学与技术系 - 潘霞科
  软件工程系 - 张良杰
  人工智能系 - 王熙熙
数量: 3

实验中心:
  计算机实验教学中心 - 黄惠
  网络工程虚拟仿真实验教学中心 - 朱安民
数量: 2

行政办公室:
  党务工作 - 杨国洪
  教学业务 - 胡沛
  实验中心 - 林佳利
  辅导员 - 黄晓聪
  科研外事 - 何文锋
  综合业务 - 刘晔
数量: 6

期刊编辑部:
  期刊编辑部 - 未知
数量: 1

进程已结束，退出代码为 0

```

## • 文字说明

在上题目的基础上，加入了实验中心、行政办公室、期刊编辑部类。

```

// 获取所有实验中心的名字和负责人
public List<String> getLabCenterNames() { 1个用法
    List<String> names = new ArrayList<>();
    for (LabCenter labCenter : labCenters) {
        names.add(labCenter.getName() + " - " + labCenter.getHead());
    }
    return names;
}

```

这一句就是把那个获取到的名字加入到names的列表中。

名字获取后的加入到 names 中。

```
// 获取期刊编辑部数量
public int getJournalNumber() { 1个用法
    return journals.size();
}
}
```

获得每个部门的数量。

```
public class Main {
    public static void main(String[] args) {
        // 输出实验中心信息
        System.out.println("\n实验中心:");
        for (String name : csse.getLabCenterNames()) {
            System.out.println(name);
        }
        System.out.println("数量: " + csse.getLabCenterNumber());

        // 输出行政办公室信息
        System.out.println("\n行政办公室:");
        for (String name : csse.getOfficeNames()) {
            System.out.println(name);
        }
        System.out.println("数量: " + csse.getOfficeNumber());

        // 输出期刊编辑部信息
        System.out.println("\n期刊编辑部:");
        for (String name : csse.getJournalNames()) {
            System.out.println(name);
        }
    }
}
```

主函数也加入了新增部分的函数调用，然后输出。

为了使得说理明白，我使用图片说明。加入的三个部分几乎和原来的方法一模一样，就是补充了一些原来没有获取到的数据。

(2.3).把 CSSE 类、Institute 类和 Department 类放进 cn.edu.szu 包中。编写一个测试类，在源代码中用 import 语句引入 cn.edu.szu 包中的所有类，并对它们所包含的方法进行测试。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

- 完整代码

值得注意的是，这个代码有三个类和一个测试代码

```
1. package cn.edu.szu;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. // Department 类
7. public class Department {
8.     private String name;
9.     private String head;
10.
11.     public Department(String name, String head) {
12.         this.name = name;
```

```

13.         this.head = head;
14.     }
15.
16.     public String getName() {
17.         return name;
18.     }
19.
20.     public String getHead() {
21.         return head;
22.     }
23. }
1.  package cn.edu.szu;
2.
3.  import java.util.ArrayList;
4.  import java.util.List;
5.
6.  // Institute 类
7.  public class Institute {
8.      private String name;
9.      private String director;
10.
11.     public Institute(String name, String director) {
12.         this.name = name;
13.         this.director = director;
14.     }
15.
16.     public String getName() {
17.         return name;
18.     }
19.
20.     public String getDirector() {
21.         return director;
22.     }
23. }
1.  package cn.edu.szu;
2.  import java.util.ArrayList;
3.  import java.util.List;
4.
5.  // CSSE 类
6.  public class CSSE {
7.      private List<Institute> institutes;
8.      private List<Department> departments;
9.
10.     public CSSE() {

```

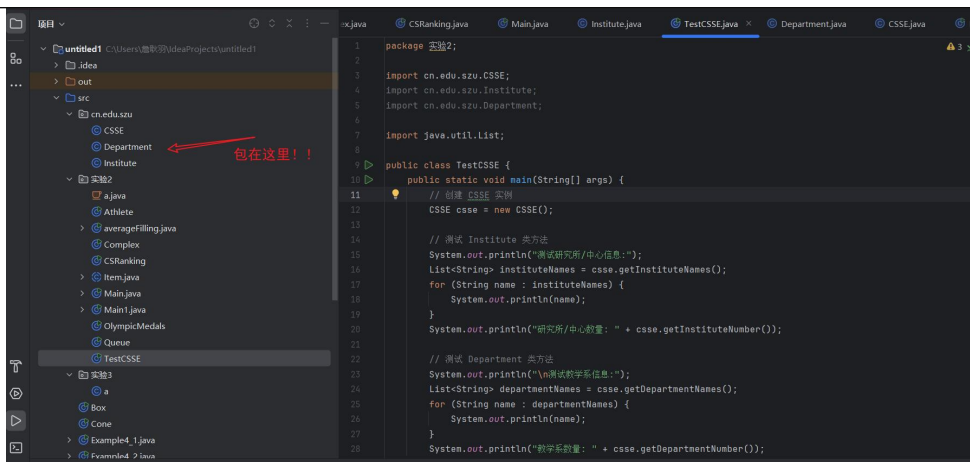
```
11.         institutes = new ArrayList<>();
12.         departments = new ArrayList<>();
13.
14.         // 添加研究所/中心
15.         institutes.add(new Institute("高性能计算研究所", "陈国良
    "));
16.         institutes.add(new Institute("大数据技术与应用研究所", "
    黄哲学"));
17.         institutes.add(new Institute("未来媒体技术与计算研究所
    ", "江建民"));
18.
19.         // 添加教学系
20.         departments.add(new Department("计算机科学与技术系", "潘
    微科"));
21.         departments.add(new Department("软件工程系", "张良杰"));
22.         departments.add(new Department("人工智能系", "王熙熙"));
23.     }
24.
25.     // 获取所有研究所/中心的名字和负责人
26.     public List<String> getInstituteNames() {
27.         List<String> names = new ArrayList<>();
28.         for (Institute institute : institutes) {
29.             names.add(institute.getName() + " - " + institute.g
    etDirector());
30.         }
31.         return names;
32.     }
33.
34.     // 获取所有教学系的名字和系主任
35.     public List<String> getDepartmentNames() {
36.         List<String> names = new ArrayList<>();
37.         for (Department department : departments) {
38.             names.add(department.getName() + " - " + department
    .getHead());
39.         }
40.         return names;
41.     }
42.
43.     // 获取研究所/中心数量
44.     public int getInstituteNumber() {
45.         return institutes.size();
46.     }
47.
48.     // 获取教学系数量
```

```
49.     public int getDepartmentNumber() {
50.         return departments.size();
51.     }
52. }
```

测试代码

```
1.  package 实验2;
2.
3.  import cn.edu.szu.CSSE;
4.  import cn.edu.szu.Institute;
5.  import cn.edu.szu.Department;
6.
7.  import java.util.List;
8.
9.  public class TestCSSE {
10.     public static void main(String[] args) {
11.         // 创建 CSSE 实例
12.         CSSE csse = new CSSE();
13.
14.         // 测试 Institute 类方法
15.         System.out.println("测试研究所/中心信息:");
16.         List<String> instituteNames = csse.getInstituteNames();
17.         for (String name : instituteNames) {
18.             System.out.println(name);
19.         }
20.         System.out.println("研究所/中心数
    量: " + csse.getInstituteNumber());
21.
22.         // 测试 Department 类方法
23.         System.out.println("\n 测试教学系信息:");
24.         List<String> departmentNames = csse.getDepartmentNames(
    );
25.         for (String name : departmentNames) {
26.             System.out.println(name);
27.         }
28.         System.out.println("教学系数
    量: " + csse.getDepartmentNumber());
29.     }
30. }
```

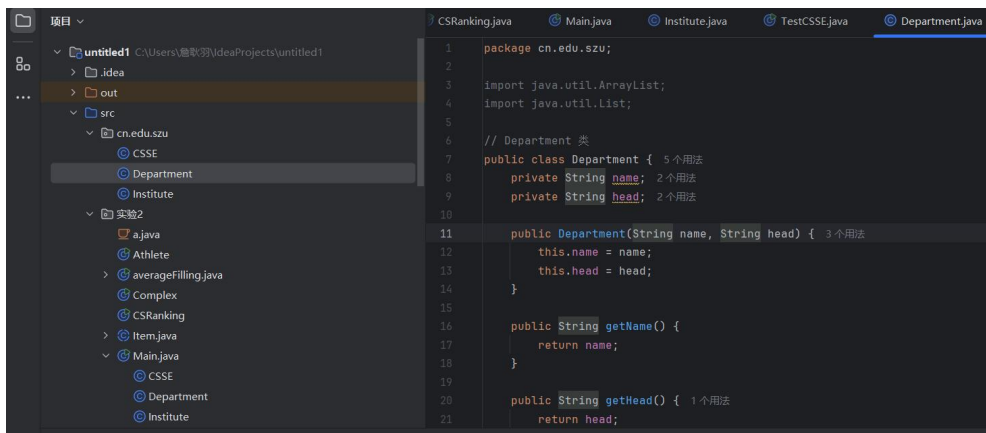
• 程序截图

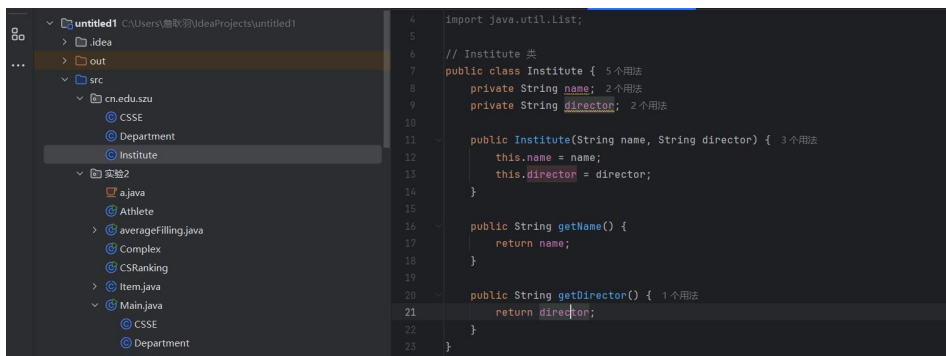


## 运行结果



## 文字说明





这两个类和上面的题目一样：  
带参构造函数、得到主任名字、得到机构名称。



在 CSSE 中加入数据。



此外，CSSE 中还调用了原来的方法。（输出机构名字、主任名字）

```
import cn.edu.szu.CSSE;
import cn.edu.szu.Institute;
import cn.edu.szu.Department;
调用之前写好的包。
```



```

import cn.edu.szu.Institute;
import cn.edu.szu.Department;

import java.util.List;

public class TestCSSE {
    public static void main(String[] args) {
        // 创建 CSSE 实例
        CSSE csse = new CSSE();

        // 测试 Institute 类方法
        System.out.println("测试研究所/中心信息:");
        List<String> instituteNames = csse.getInstituteNames();
        for (String name : instituteNames) {

```

调用的包的应用。

最后与原方法一样输出即可。

(2.4).通过文字解释或程序来说明以下各种组合是否允许，如果允许表示什么意思，如果不允许是为什么。同时，在下表中，对不允许的组合，填入 NO。（5 分）

	类	类中的成员变量	类中的成员方法	类中的构造方法	接口中的成员变量
private	Yes	Yes	Yes	Yes	NO
public	Yes	Yes	Yes	Yes	NO
final	NO	Yes	Yes	NO	NO
abstract	Yes	NO	Yes	NO	NO
static	NO	Yes	Yes	NO	NO

• Java 中，private 可以是内部私有类。Public 可以修饰类，这是很显然的。（上面的代码都是有 public 的类存在。）抽象类也可以存在。但是 final 是一旦定义后便不可以改变的，从而没有见到其能在声明类，static 同理，静态类不存在。

• 同理，也没有类中的成员变量是抽象类型的，抽象一般只使用在类。另外的，其余私有、共有、定义后不可以改变的、静态都可以运用到成员变量。

• 类中的方法是所有都可以使用的，这里我对抽象方法进行说明。（这个的意思是先定义一个抽象的方法，后面就会继续完善这个方法。）static 方法就是可以修改 static 变量的方法。

- 构造方法只可以用 private 和 public。（特定的，不然会报错）
- 接口中的成员变量只可以使用 final static 来定义！因此五个都不行。

(2.5).面向对象编程有三个特性（封装、继承和多态），请对“封装”、“继承”和“多态”这三个特性，通过类比、关联或演绎的方式，举一个在日常的学习生活中可以应用的例子（要求积极向上且能自圆其说）。（5 分）

面向对象编程（OOP）的三个特性：封装、继承和多态，可以通过一个学校的例子来形象地说明。

### 封装

类比：想象一下，一个学生的学习资料被整理在一个文件夹里。这个文件夹包含了他的课本、笔记、作业等。

解释：封装就像这个文件夹，学生把所有与学习相关的信息放在一起，外界无法直接访问这些信息，只有学生自己可以打开文件夹、查看和修改其中的内容。这种方式保护了学生的隐私，也避免了外部干扰。程序中，封装通过类和访问修饰符（如 'private' 和

`public`) 来实现，只允许必要的方法对外暴露。

### 继承

类比：在学校的课程体系中，数学是一个基础学科，而高等数学可以被看作是数学的一部分，具有更多复杂的内容。

解释：继承就像这种学科关系，高等数学“继承”了数学的基本概念，但同时又增加了新的内容和重点。在编程中，继承使得一个类可以从另一个类派生，重用已有的代码并扩展功能。例如，`Math` 类可以作为一个基类，而 `AdvancedMath` 类在其基础上添加新方法和属性。

### 多态

类比：在学校里，老师可以根据不同的课程使用不同的教学方式。例如，同样是“讲解知识”，数学老师可能通过举例说明，而语文老师可能通过分析文章来实现。

解释：多态体现了同一操作在不同对象上的不同表现。在编程中，多态允许我们使用相同的方法名来执行不同的操作。例如，一个 `Teacher` 类可能有一个 `teach` 方法，而不同的教师（如 `MathTeacher` 和 `LanguageTeacher`）可以对这个方法进行不同的实现。这种灵活性使得程序更加简洁和易于扩展。

## Part 3 (30 分)

(1). 抽象类和接口的实验。（10 分）

(i) 定义一个抽象类 Human：包含一个成员变量 String name；构造方法 Human(String name)，用于初始化姓名 name；一个抽象方法 sayHello()。在报告中附上程序截图和详细的文字说明。

• 程序截图

```
package 实验2;

public abstract class Human { 0 个用法
    protected String name;

    public Human(String name) { 0 个用法
        this.name = name;
    }

    public abstract void sayHello(); 0 个用法
}
```

• 文字说明：

protected String name;：这是一个受保护的成员变量，允许子类访问。

public Human(String name)：构造方法接收一个字符串参数 name，用于初始化成员变量。

public abstract void sayHello();：这是一个抽象方法，子类必须实现这个方法，提供具体的问候方式。

(ii) 定义三个继承抽象类 Human 的类，分别命名为 Chinese、Spaniard 和 Italian，在这三个类中重写 sayHello()方法，分别输出一句中文、西班牙语和意大利语的问候；在报告中附上程序截图、运行结果和详细的文字说明。

• 程序截图

```
1 package 实验2;
2 @ abstract class Human { 6个用法 3个继承者
3     protected String name;
4
5     public Human(String name) { 3个用法
6         this.name = name;
7     }
8
9     public abstract void sayHello(); 3个用法 3个实现
10 }
11
12 // Chinese类
13 class Chinese extends Human { 1个用法
14
15     public Chinese(String name) { 1个用法
16         super(name);
17     }
18
19     @Override 3个用法
20     public void sayHello() {
21         System.out.println("你好, 我叫 " + name);
22     }
23 }
24
25 // Spaniard类
26 class Spaniard extends Human { 1个用法
```

• 完整代码

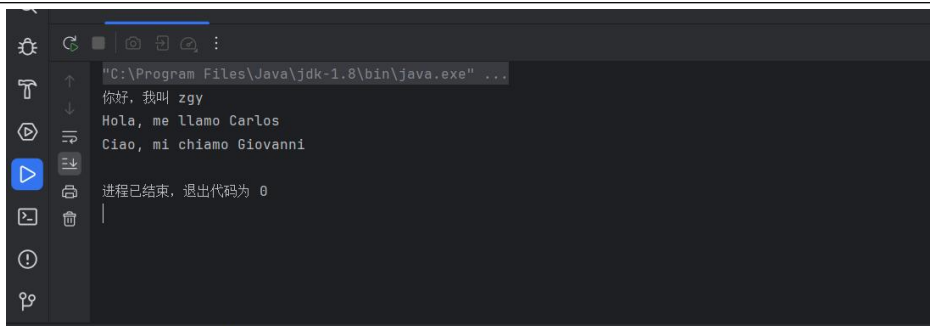
```
1. package 实验2;
2. abstract class Human {
3.     protected String name;
4.
5.     public Human(String name) {
6.         this.name = name;
7.     }
8.
9.     public abstract void sayHello();
10. }
11.
12. // Chinese 类
13. class Chinese extends Human {
14.
15.     public Chinese(String name) {
16.         super(name);
17.     }
18.
19.     @Override
20.     public void sayHello() {
21.         System.out.println("你好, 我叫 " + name);
22.     }
23. }
24.
25. // Spaniard 类
```

```

26. class Spaniard extends Human {
27.
28.     public Spaniard(String name) {
29.         super(name);
30.     }
31.
32.     @Override
33.     public void sayHello() {
34.         System.out.println("Hola, me llamo " + name);
35.     }
36. }
37.
38. // Italian 类
39. class Italian extends Human {
40.
41.     public Italian(String name) {
42.         super(name);
43.     }
44.
45.     @Override
46.     public void sayHello() {
47.         System.out.println("Ciao, mi chiamo " + name);
48.     }
49. }
50.
51. // 测试类 Main
52. public class Mian2 {
53.     public static void main(String[] args) {
54.         Human chinese = new Chinese("zgy");
55.         Human spaniard = new Spaniard("Carlos");
56.         Human italian = new Italian("Giovanni");
57.
58.         chinese.sayHello();
59.         spaniard.sayHello();
60.         italian.sayHello();
61.     }
62. }

```

- 运行结果



- 文字解释

定义抽象类: **Human** 是一个抽象类, 不能被实例化。它包含一个受保护的成员变量 **name**, 以及一个构造函数用于初始化 **name**。

抽象方法: 定义了一个抽象方法 **sayHello()**, 这个方法没有实现, 子类必须提供具体的实现。

继承: **Chinese** 类继承自 **Human** 类。(其他类都是相似的)

构造函数: 调用父类的构造函数以初始化 **name**。

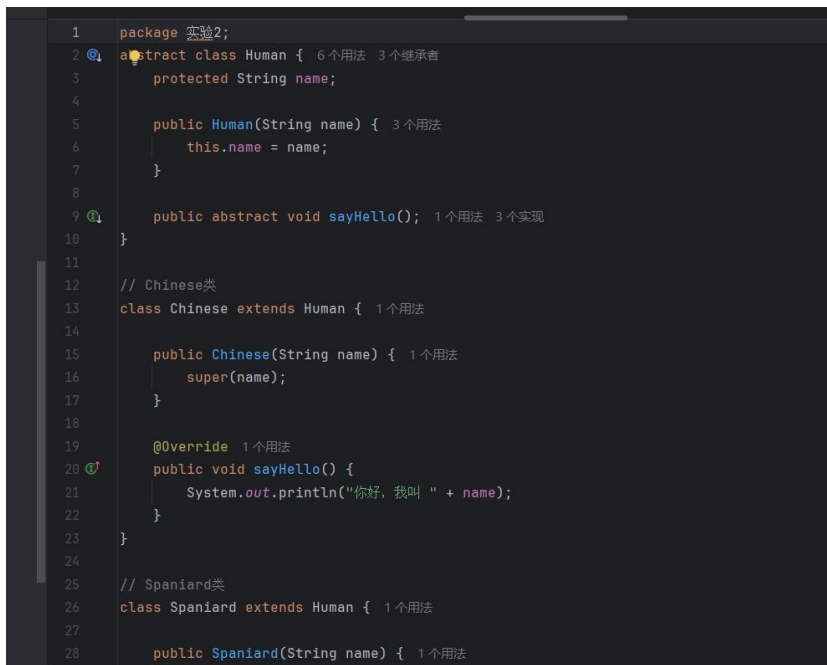
实现抽象方法: 实现 **sayHello()** 方法, 输出中文问候。

主方法: 创建了三个 **Human** 类型的对象, 分别是 **Chinese**、**Spaniard** 和 **Italian**。

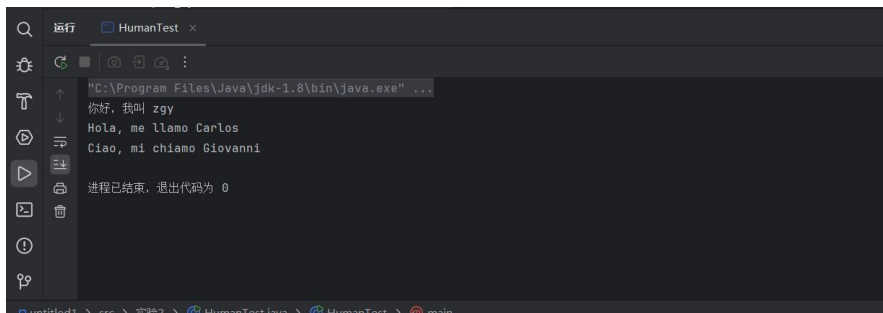
调用方法: 通过调用 **sayHello()** 方法, 打印出不同语言的问候语。

(iii) 定义一个测试类 **HumanTest**: 创建一个包含 3 个 **Human** 对象的数组, 3 个 **Human** 对象来自 **Chinese**、**Spaniard** 和 **Italian** 类, 循环调用该数组中的元素的 **sayHello()** 方法。在报告中附上程序截图、运行结果和详细的文字说明。

- 程序截图



- 运行结果



- 完整代码

```
1. package 实验2;  
2. abstract class Human {  
3.     protected String name;  
4.  
5.     public Human(String name) {  
6.         this.name = name;  
7.     }  
8.  
9.     public abstract void sayHello();  
10. }  
11.  
12. // Chinese 类  
13. class Chinese extends Human {  
14.  
15.     public Chinese(String name) {  
16.         super(name);  
17.     }  
18.  
19.     @Override  
20.     public void sayHello() {  
21.         System.out.println("你好, 我叫 " + name);  
22.     }  
23. }  
24.  
25. // Spaniard 类  
26. class Spaniard extends Human {  
27.  
28.     public Spaniard(String name) {  
29.         super(name);  
30.     }  
31.  
32.     @Override  
33.     public void sayHello() {  
34.         System.out.println("Hola, me llamo " + name);  
35.     }
```

```

36. }
37.
38. // Italian 类
39. class Italian extends Human {
40.
41.     public Italian(String name) {
42.         super(name);
43.     }
44.
45.     @Override
46.     public void sayHello() {
47.         System.out.println("Ciao, mi chiamo " + name);
48.     }
49. }
50. public class HumanTest {
51.     public static void main(String[] args) {
52.         // 创建 Human 对象的数组
53.         Human[] humans = new Human[3];
54.         // 初始化数组中的元素
55.         humans[0] = new Chinese("zgy");
56.         humans[1] = new Spaniard("Carlos");
57.         humans[2] = new Italian("Giovanni");
58.         // 循环调用 sayHello() 方法
59.         for (Human human : humans) {
60.             human.sayHello();
61.         }
62.     }
63. }

```

- 文字说明

与上题类似，不同点是：测试类 HumanTest。

定义: HumanTest 类包含 main 方法，是程序的入口。

创建数组: 定义一个 Human 类型的数组 humans，容量为 3。

初始化对象: 将不同国家的 Human 子类对象添加到数组中。

调用 sayHello 方法: 使用增强型 for 循环遍历数组，调用每个对象的 sayHello 方法，输出对应的问候语。

(iv) 通过一个接口（命名为 Human）和三个实现类（命名为 Chinese、Spaniard 和 Italian）来达到如上类似的效果。在报告中附上程序截图、运行结果和详细的文字说明。

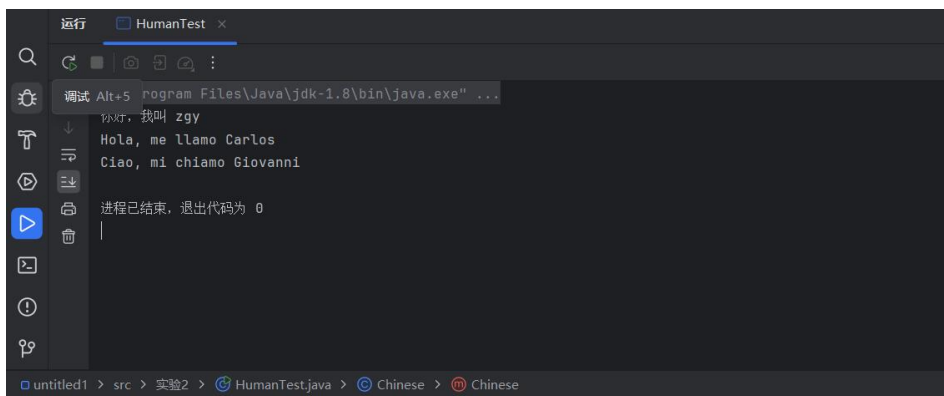
- 程序截图

```

1  package 实验2;
2
3  interface Human { 6个用法 3个实现
4      void sayHello(); 1个用法 3个实现
5  }
6  class Chinese implements Human { 1个用法
7      private String name; 2个用法
8
9      public Chinese(String name) { 1个用法
10         this.name = name;
11     }
12
13     @Override 1个用法
14     public void sayHello() {
15         System.out.println("你好, 我叫 " + name);
16     }
17 }
18
19 // 实现类 Spaniard
20 class Spaniard implements Human { 1个用法
21     private String name; 2个用法
22
23     public Spaniard(String name) { 1个用法
24         this.name = name;
25     }

```

#### • 运行截图



```

运行 HumanTest x
调试 Alt+5 program Files\Java\jdk-1.8\bin\java.exe ...
你好, 我叫 zgy
Hola, me llamo Carlos
Ciao, mi chiamo Giovanni
进程已结束, 退出代码为 0
untitled1 > src > 实验2 > HumanTest.java > Chinese > Chinese

```

#### • 完整代码

```

1. package 实验2;
2. interface Human {
3.     void sayHello();
4. }
5. class Chinese implements Human {
6.     private String name;
7.
8.     public Chinese(String name) {
9.         this.name = name;
10.    }
11.
12.    @Override

```



```
13.     public void sayHello() {
14.         System.out.println("你好, 我叫 " + name);
15.     }
16. }
17.
18. // 实现类 Spaniard
19. class Spaniard implements Human {
20.     private String name;
21.
22.     public Spaniard(String name) {
23.         this.name = name;
24.     }
25.
26.     @Override
27.     public void sayHello() {
28.         System.out.println("Hola, me llamo " + name);
29.     }
30. }
31.
32. // 实现类 Italian
33. class Italian implements Human {
34.     private String name;
35.
36.     public Italian(String name) {
37.         this.name = name;
38.     }
39.
40.     @Override
41.     public void sayHello() {
42.         System.out.println("Ciao, mi chiamo " + name);
43.     }
44. }
45.
46. // 测试类 HumanTest
47. public class HumanTest {
48.     public static void main(String[] args) {
49.         // 创建 Human 对象的数组
50.         Human[] humans = new Human[3];
51.
52.         // 初始化数组中的元素
53.         humans[0] = new Chinese("zgy");
54.         humans[1] = new Spaniard("Carlos");
55.         humans[2] = new Italian("Giovanni");
56.     }
57. }
```

```

57.          // 循环调用 sayHello() 方法
58.          for (Human human : humans) {
59.              human.sayHello();
60.          }
61.      }
62.  }

```

- 文字说明

与之前不同的是 **Human** 是一个接口,它定义了一个方法 **sayHello()**。接口是 Java 中的一种抽象类型,允许类实现这个接口并提供具体的实现。抽象方法: **sayHello()** 是一个没有实现的方法,任何实现 **Human** 接口的类都必须提供该方法的具体操作。

**Chinese** 类(其他类也一样)实现了 **Human** 接口。这意味着 **Chinese** 类必须实现 **sayHello()** 方法。

其他与之前一致。

(2).一个四维整数向量由四个分量组成。四维向量的相加、相减和点乘等价于对应四个分量的相加、相减和相乘,四维向量的内积等价于点乘所得向量中各个元素的和。比如两个四维向量[3,9,2,7]和[2,-8,-1,6],它们的和为[5,1,1,13],它们的差为[1,17,3,1],它们的点乘为[6,-72,-2,42],它们的内积为-26。向量的模(norm)表示该向量所有分量的平方和的根,例如向量[3,9,2,7]的模为 11.96。编写一个接口 **Computable**,它具有 6 个抽象方法 **add**、**minus**、**elementwiseProduct**、**innerProduct**、**norm** 和 **compare**。编写一个 **Vector** 类,通过 **Computable** 接口实现四维向量的相加、相减、点乘、内积、模和比较(根据模的大小)。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

- 程序截图

```

1  package 实验2;
2  interface Computable {
3      // 向量相加
4      Vector add(Vector other);
5      // 向量相减
6      Vector minus(Vector other);
7      // 元素逐个相乘
8      Vector elementwiseProduct(Vector other);
9      // 内积
10     int innerProduct(Vector other);
11     // 向量的模
12     double norm();
13     // 比较两个向量的模
14     int compare(Vector other);
15 }
16 class Vector implements Computable {
17     private int[] components;
18     // 构造函数
19     public Vector(int x, int y, int z, int w) {
20         components = new int[] { x, y, z, w };
21     }

```

- 完整代码

```

1.  package 实验2;
2.  interface Computable {
3.      // 向量相加
4.      Vector add(Vector other);
5.      // 向量相减
6.      Vector minus(Vector other);
7.      // 元素逐个相乘
8.      Vector elementwiseProduct(Vector other);
9.      // 内积
10.     int innerProduct(Vector other);
11.     // 向量的模

```

```
12.     double norm();
13.     // 比较两个向量的模
14.     int compare(Vector other);
15. }
16. class Vector implements Computable {
17.     private int[] components;
18.
19.     // 构造函数
20.     public Vector(int x, int y, int z, int w) {
21.         components = new int[] { x, y, z, w };
22.     }
23.
24.     @Override
25.     public Vector add(Vector other) {
26.         return new Vector(
27.             this.components[0] + other.components[0],
28.             this.components[1] + other.components[1],
29.             this.components[2] + other.components[2],
30.             this.components[3] + other.components[3]
31.         );
32.     }
33.
34.     @Override
35.     public Vector minus(Vector other) {
36.         return new Vector(
37.             this.components[0] - other.components[0],
38.             this.components[1] - other.components[1],
39.             this.components[2] - other.components[2],
40.             this.components[3] - other.components[3]
41.         );
42.     }
43.
44.     @Override
45.     public Vector elementwiseProduct(Vector other) {
46.         return new Vector(
47.             this.components[0] * other.components[0],
48.             this.components[1] * other.components[1],
49.             this.components[2] * other.components[2],
50.             this.components[3] * other.components[3]
51.         );
52.     }
53.
54.     @Override
55.     public int innerProduct(Vector other) {
```

```

56.         return (this.components[0] * other.components[0]) +
57.                 (this.components[1] * other.components[1]) +
58.                 (this.components[2] * other.components[2]) +
59.                 (this.components[3] * other.components[3]);
60.     }
61.
62.     @Override
63.     public double norm() {
64.         return Math.sqrt(
65.             Math.pow(this.components[0], 2) +
66.             Math.pow(this.components[1], 2) +
67.             Math.pow(this.components[2], 2) +
68.             Math.pow(this.components[3], 2)
69.         );
70.     }
71.
72.     @Override
73.     public int compare(Vector other) {
74.         double thisNorm = this.norm();
75.         double otherNorm = other.norm();
76.         return Double.compare(thisNorm, otherNorm); // 返回负
           数、零或正数
77.     }
78.
79.     @Override
80.     public String toString() {
81.         return "[" + components[0] + ", " + components[1] + ",
           " +
82.             components[2] + ", " + components[3] + "]";
83.     }
84. }
85. public class main3 {
86.     public static void main(String[] args) {
87.         Vector v1 = new Vector(3, 9, 2, 7);
88.         Vector v2 = new Vector(2, -8, -1, 6); // 给好的例子
89.
90.         System.out.println("v1: " + v1);
91.         System.out.println("v2: " + v2);
92.
93.         Vector sum = v1.add(v2);
94.         System.out.println("Sum: " + sum);
95.
96.         Vector difference = v1.minus(v2);
97.         System.out.println("相减: " + difference);

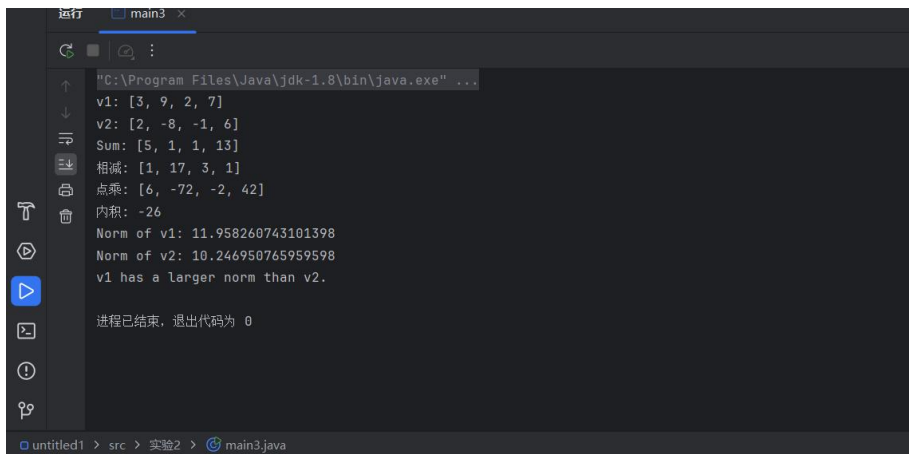
```

```

98.
99.         Vector product = v1.elementwiseProduct(v2);
100.        System.out.println("点乘: " + product);
101.
102.        int innerProduct = v1.innerProduct(v2);
103.        System.out.println("内积: " + innerProduct);
104.
105.        double normV1 = v1.norm();
106.        System.out.println("Norm of v1: " + normV1);
107.
108.        double normV2 = v2.norm();
109.        System.out.println("Norm of v2: " + normV2);
110.
111.        int comparison = v1.compare(v2);
112.        if (comparison < 0) {
113.            System.out.println("v1 has a smaller norm than v2.");
114.        } else if (comparison > 0) {
115.            System.out.println("v1 has a larger norm than v2.");
116.        } else {
117.            System.out.println("v1 and v2 have the same norm.");
118.        }
119.    }
120. }

```

- 运行结果



```

运行  main3
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
v1: [3, 9, 2, 7]
v2: [2, -8, -1, 6]
Sum: [5, 1, 1, 13]
相减: [1, 17, 3, 1]
点乘: [6, -72, -2, 42]
内积: -26
Norm of v1: 11.958260743101398
Norm of v2: 10.246950765959598
v1 has a larger norm than v2.

进程已结束, 退出代码为 0
untitled1 > src > 实验2 > main3.java

```

- 文字说明

- **Computable** 是一个接口, 定义了一组方法, 用于对向量进行计算。这些方法包括:

- add(Vector other):** 向量相加。
- minus(Vector other):** 向量相减。
- elementwiseProduct(Vector other):** 逐元素相乘。

**innerProduct(Vector other):** 计算内积。

**norm():** 计算向量的模（长度）。

**compare(Vector other):** 比较两个向量的模大小。

- Vector 类实现了 **Comparable** 接口，具体实现了接口中的方法：

字段 **components**: 使用一个整数数组来存储四个维度的值（x, y, z, w）。

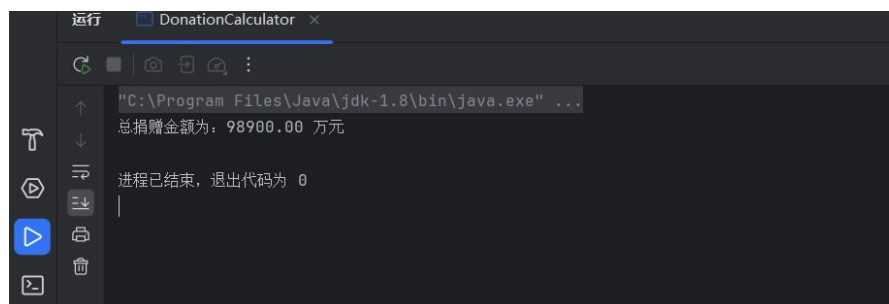
构造函数：接受四个整数参数并初始化 **components** 数组。

(3). 编写 Java 应用程序，通过字符串解析，计算字符串“上述消息提到，4 月 27 日晚举行的深圳大学 40 周年校庆捐赠仪式暨“海岸之声”音乐晚会上，多家企业向深圳大学 40 周年校庆进行捐赠。明礼德教育科技有限公司向深圳大学捐赠 1000 万元；心里程控股集团向深圳大学捐赠 1 亿元；工勘岩土集团捐赠 4000 万元；正中集团捐赠 5000 万元；海岸集团捐赠 6000 万元；腾讯公益慈善基金会捐赠 2 亿元。此前，正中集团已向深大捐赠 4700 万元，海岸集团已向深大捐赠 2200 万元，腾讯创始人校友团队和腾讯公益慈善基金会已向深大捐赠 3.9 亿元。除此之外，平安集团捐赠 5000 万元，点维文化传播捐赠 1000 万元，叶晓彬校友捐赠 1000 万元，已于日前完成相关签约。”的总金额。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（5 分）

#### • 程序截图

```
1 package 实验2;
2 import java.util.regex.Matcher;
3 import java.util.regex.Pattern;
4 public class DonationCalculator {
5     public static void main(String[] args) {
6         String text = "上述消息提到，4月27日晚举行的深圳大学40周年校庆捐赠仪式暨“海岸之声”音乐晚会上，多家企业向深圳大学40周年校庆进行捐赠。明礼德教育科技有限公司向深圳大学捐赠 1000 万元；心里程控股集团向深圳大学捐赠 1 亿元；工勘岩土集团捐赠 4000 万元；正中集团捐赠 5000 万元；海岸集团捐赠 6000 万元；腾讯公益慈善基金会捐赠 2 亿元。此前，正中集团已向深大捐赠 4700 万元，海岸集团已向深大捐赠 2200 万元，腾讯创始人校友团队和腾讯公益慈善基金会已向深大捐赠 3.9 亿元。除此之外，平安集团捐赠 5000 万元，点维文化传播捐赠 1000 万元，叶晓彬校友捐赠 1000 万元，已于日前完成相关签约。”的总金额。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（5 分）";
7         Pattern pattern = Pattern.compile("((\\d+(?:\\.\\d+)?)(万|亿))");
8         Matcher matcher = pattern.matcher(text);
9         double totalAmount = 0.0;
10        while (matcher.find()) {
11            double amount = Double.parseDouble(matcher.group(1));
12            String unit = matcher.group(2);
13            if ("亿".equals(unit)) {
14                amount *= 10000; // 将亿转换为万元
15            }
16            // 累加金额
17            totalAmount += amount;
18        }
19        System.out.printf("总捐赠金额为: %.2f 万元\n", totalAmount);
20    }
21 }
```

#### • 运行结果



```
运行 DonationCalculator x
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
总捐赠金额为: 98900.00 万元
进程已结束, 退出代码为 0
```

• 完整代码已在程序截图中展示。

• 文字说明

导入包：我们首先导入 **java.util.regex** 包，以便使用正则表达式。

文本定义：将包含捐赠信息的字符串存储在 **text** 变量中。

正则表达式：使用正则表达式 **"((\\d+(?:\\.\\d+)?)(万|亿))"** 来匹配金额：

- \d+ 匹配一个或多个数字。
- (?:\.\d+)? 可选地匹配小数部分。
- (万|亿) 匹配单位“万”或“亿”。

使用 `Matcher` 对象遍历所有匹配项：

- 将匹配到的金额字符串转换为 `double` 类型。
- 根据单位“万”或“亿”调整金额（“亿”乘以 10000）。
- 累加到 `totalAmount` 变量中。

输出结果：最后输出总捐赠金额，格式化为两位小数。

(4). 编写 Java 应用程序，随机生成一个包含有大写英文字母、小写英文字母、数字和其他字符混杂的字符串(例如 `Aa123bEFGa$aa@49023`)，解析该字符串并要求按顺序输出大写英文字母（例如 `AEFG`）、小写英文字母（`abaaa`）、数字（`12349023`）和其他字符（`$@`）。要求循环连续测试 5 次，在报告中附上程序截图、完整的运行结果截图和简要文字说明。（5 分）

#### • 程序截图

```

package 实验2;
import java.security.SecureRandom;
public class RandomStringParser {
    private static final String UPPERCASE = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // 1个测试
    private static final String LOWERCASE = "abcdefghijklmnopqrstuvwxyz"; // 1个测试
    private static final String DIGITS = "0123456789"; // 1个测试
    private static final String SPECIAL_CHARS = "!@#$%^&*()_+~='\";.<>?/'~"; // 1个测试

    public static void main(String[] args) {
        SecureRandom random = new SecureRandom();
        for (int i = 0; i < 5; i++) {
            String randomString = generateRandomString(random, 20); // 生成长度为20的随机字符串
            System.out.println("随机生成的字符串: " + randomString);
            parseAndPrintString(randomString);
            System.out.println(); // 输出空行分隔测试
        }
    }

    private static String generateRandomString(SecureRandom random, int length) { // 1个测试
        StringBuilder sb = new StringBuilder(length);
        String allChars = UPPERCASE + LOWERCASE + DIGITS + SPECIAL_CHARS;

        for (int i = 0; i < length; i++) {
            int index = random.nextInt(allChars.length());
            sb.append(allChars.charAt(index));
        }

        return sb.toString();
    }
}
  
```

#### • 完整代码

```

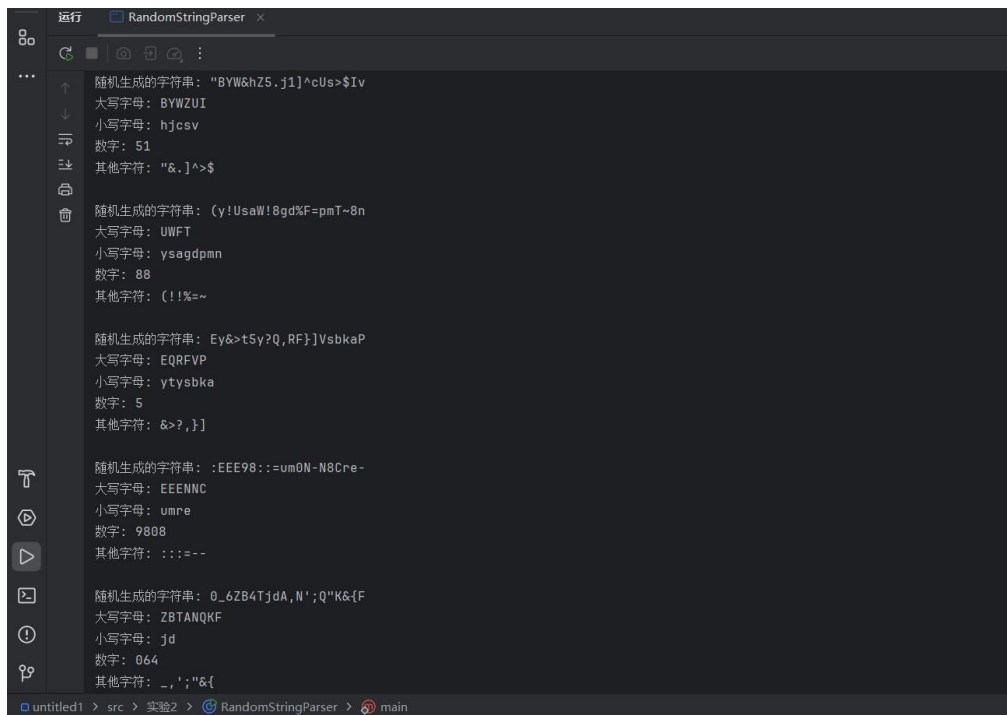
1. package 实验2;
2. import java.security.SecureRandom;
3. public class RandomStringParser {
4.     private static final String UPPERCASE = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
5.     private static final String LOWERCASE = "abcdefghijklmnopqrstuvwxyz";
6.     private static final String DIGITS = "0123456789";
7.     private static final String SPECIAL_CHARS = "!@#$%^&*()_+~='\";.<>?/'~";
8.
9.     public static void main(String[] args) {
10.         SecureRandom random = new SecureRandom();
11.         for (int i = 0; i < 5; i++) {
12.             String randomString = generateRandomString(random,
13.                 20); // 生成长度为20的随机字符串
14.             System.out.println("随机生成的字符串: " + randomString);
15.             parseAndPrintString(randomString);
16.             System.out.println(); // 输出空行分隔测试
17.         }
18.     }
19.
20.     private static String generateRandomString(SecureRandom random, int length) { // 1个测试
21.         StringBuilder sb = new StringBuilder(length);
22.         String allChars = UPPERCASE + LOWERCASE + DIGITS + SPECIAL_CHARS;
23.
24.         for (int i = 0; i < length; i++) {
25.             int index = random.nextInt(allChars.length());
26.             sb.append(allChars.charAt(index));
27.         }
28.
29.         return sb.toString();
30.     }
31. }
  
```

```
14.         parseAndPrintString(randomString);
15.         System.out.println(); // 输出空行分隔测试
16.     }
17. }
18.
19.     private static String generateRandomString(SecureRandom random, int length) {
20.         StringBuilder sb = new StringBuilder(length);
21.         String allChars = UPPERCASE + LOWERCASE + DIGITS + SPECIAL_CHARS;
22.
23.         for (int i = 0; i < length; i++) {
24.             int index = random.nextInt(allChars.length());
25.             sb.append(allChars.charAt(index));
26.         }
27.
28.         return sb.toString();
29.     }
30.
31.     private static void parseAndPrintString(String str) {
32.         StringBuilder upperCaseLetters = new StringBuilder();
33.         StringBuilder lowerCaseLetters = new StringBuilder();
34.         StringBuilder digits = new StringBuilder();
35.         StringBuilder specialChars = new StringBuilder();
36.
37.         for (char c : str.toCharArray()) {
38.             if (Character.isUpperCase(c)) {
39.                 upperCaseLetters.append(c);
40.             } else if (Character.isLowerCase(c)) {
41.                 lowerCaseLetters.append(c);
42.             } else if (Character.isDigit(c)) {
43.                 digits.append(c);
44.             } else {
45.                 specialChars.append(c);
46.             }
47.         }
48.         System.out.println("大写字母: " + upperCaseLetters.toString());
49.         System.out.println("小写字母: " + lowerCaseLetters.toString());
50.         System.out.println("数字: " + digits.toString());
51.         System.out.println("其他字符: " + specialChars.toString());
52.     }
```



## 53. }

### • 运行结果



```
运行 RandomStringParser x
随机生成的字符串: "BYW&hZ5.j1]^cUs>$Iv
大写字母: BYWZUI
小写字母: hjcsv
数字: 51
其他字符: "&.^>$

随机生成的字符串: {y!UsaW!8gd%F=pmT~8n
大写字母: UWFT
小写字母: ysagdpmm
数字: 88
其他字符: {!}%~

随机生成的字符串: Ey&>t5y?Q,Rf}}VsbkaP
大写字母: EQRFVP
小写字母: ytysbka
数字: 5
其他字符: &>?,}}

随机生成的字符串: :EEE98::=um0N-N8Cre-
大写字母: EEEENNC
小写字母: umre
数字: 9808
其他字符: :::=-

随机生成的字符串: 0_6ZB4TjdA,N';Q"K&{F
大写字母: ZBTANQKF
小写字母: jd
数字: 064
其他字符: _,';'"&{

untitled1 > src > 实验2 > RandomStringParser > main
```

### • 文字说明

字符集定义:

- 定义了大写字母、小写字母、数字和特殊字符的字符串常量。

随机数生成:

- 使用 `SecureRandom` 类生成安全的随机数。

主循环:

- 在主函数中, 使用一个 `for` 循环执行 5 次生成和解析字符串的过程。

- 每次生成一个长度为 20 的随机字符串。

字符串生成:

- `generateRandomString` 方法通过随机选择字符构建一个随机字符串。

解析字符串:

- `parseAndPrintString` 方法遍历字符串的每个字符, 将其分类到大写字母、小写字母、数字和特殊字符的对应字符串中。

最后输出这些分类后的结果。

(5). 编写 Java 应用程序, 统计分析网页 <https://csse.szu.edu.cn/en/pages/university/index> 中关于深圳大学计算机与软件学院的重要科研平台 (platform) 的英文介绍中每个英文单词出现的次数 (统一转为小写, 不需要写爬虫, 可以把整篇文章的内容当作一个字符串读入), 并输出出现次数最多的 10 个英文单词 (按出现次数排序从大到小排列, 如次数相同则按字母顺序)。在报告中附上程序截图、完整的运行结果截图和简要文字说明。

(5 分)

### • 程序截图

```

1 package 实验2;
2 import java.util.*;
3 public class WordCount {
4     public static void main(String[] args) {
5         // 提供的文章内容
6         String text = "The Computer Science and Technology discipline at Shenzhen University was founded with support from Tsinghua University in 1983 when the Shenzhen University was just established. College of Computer Science and Software Engineering was formally established in 2008, which was headed by Guoliang Chen, the Academician of the Chinese Academy of Sciences, as its founding Dean. Over the years' development, CSSE has now led by Professor Hui Huang and become a prominent college in education and research nationally and globally. The platforms include two national-level research platforms (National Engineering Laboratory for the Big Data System Computing Technology, MOE-GD Collaborative Innovation Center for GD-HK Modern Information Service), three national-level teaching platforms (Computer Experimental Teaching Center, Virtual Simulation Experimental Center for Network Engineering, and Tencent Cloud AI College), the Guangdong Laboratory of Artificial Intelligence and Digital Economy (Shenzhen), ten other provincial-level and ten municipal-level platforms.";
7
8         // 使用正则表达式分割字符串为单词数组，并转为小写
9         String[] words = text.toLowerCase().split(regex: "\\W+");
10
11        // 使用HashMap统计每个单词的出现次数
12        Map<String, Integer> wordCounts = new HashMap<>();
13        for (String word : words) {
14            if (!word.isEmpty()) {
15                wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);
16            }
17        }
18
19        // 将Map条目转移到列表中，以便排序
20        List<Map.Entry<String, Integer>> sortedWords = new ArrayList<>(wordCounts.entrySet());
21
22        // 按单词出现次数降序排序，次数相同按字母顺序排序
23        Collections.sort(sortedWords, (a, b) -> {
24            if (b.getValue().equals(a.getValue())) {
25                return a.getKey().compareTo(b.getKey());
26            }
27            return b.getValue() - a.getValue();
28        });
29
30        // 输出出现次数最多的10个单词
31        System.out.println("Top 10 most frequent words:");

```

#### • 完整代码

```

1. package 实验2;
2. import java.util.*;
3. public class WordCount {
4.     public static void main(String[] args) {
5.         // 提供的文章内容
6.         String text = "The Computer Science and Technology discipline at Shenzhen University was founded with support from Tsinghua University in 1983 when the Shenzhen University was just established. College of Computer Science and Software Engineering was formally established in 2008, which was headed by Guoliang Chen, the Academician of the Chinese Academy of Sciences, as its founding Dean. Over the years' development, CSSE has now led by Professor Hui Huang and become a prominent college in education and research nationally and globally. The platforms include two national-level research platforms (National Engineering Laboratory for the Big Data System Computing Technology, MOE-GD Collaborative Innovation Center for GD-HK Modern Information Service), three national-level teaching platforms (Computer Experimental Teaching Center, Virtual Simulation Experimental Center for Network Engineering, and Tencent Cloud AI College), the Guangdong Laboratory of Artificial Intelligence and Digital Economy (Shenzhen), ten other provincial-level and ten municipal-level platforms.";
7.         // 使用正则表达式分割字符串为单词数组，并转为小写
8.         String[] words = text.toLowerCase().split("\\W+");
9.
10.        // 使用HashMap 统计每个单词的出现次数
11.        Map<String, Integer> wordCounts = new HashMap<>();
12.        for (String word : words) {
13.            if (!word.isEmpty()) {
14.                wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);

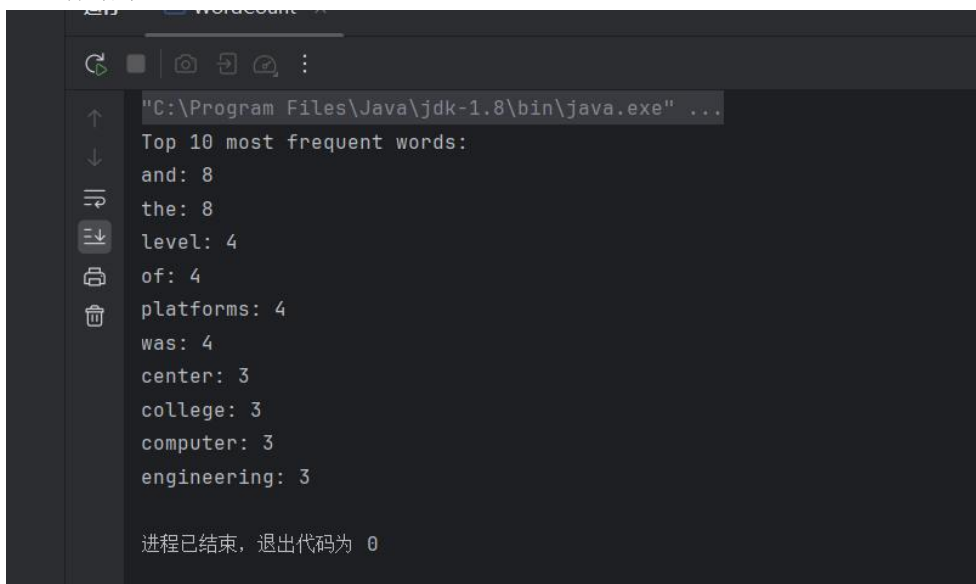
```

```

15.         }
16.     }
17.
18.     // 将Map 条目转移到列表中，以便排序
19.     List<Map.Entry<String, Integer>> sortedWords = new ArrayList<>(wordCounts.
        entrySet());
20.
21.     // 按单词出现次数降序排序，次数相同按字母顺序排序
22.     Collections.sort(sortedWords, (a, b) -> {
23.         if (b.getValue().equals(a.getValue())) {
24.             return a.getKey().compareTo(b.getKey());
25.         }
26.         return b.getValue() - a.getValue();
27.     });
28.
29.     // 输出出现次数最多的10个单词
30.     System.out.println("Top 10 most frequent words:");
31.     for (int i = 0; i < Math.min(10, sortedWords.size()); i++) {
32.         System.out.println(sortedWords.get(i).getKey() + ": " + sortedWords.ge
            t(i).getValue());
33.     }
34. }
35. }

```

#### • 运行结果



```

"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
Top 10 most frequent words:
and: 8
the: 8
level: 4
of: 4
platforms: 4
was: 4
center: 3
college: 3
computer: 3
engineering: 3

进程已结束，退出代码为 0

```

#### • 文字说明

字符串分割：使用 `split("\\W+")` 来分割非单词字符，这确保了只处理英文单词。

统计词频：使用 `HashMap` 来统计每个单词出现的次数。

排序：使用 `Collections.sort` 方法，首先按出现次数降序排序，如果次数相同，则按字母顺序排序。

输出：只输出列表中前 10 个单词和它们的出现次数。

+++++

其他（例如感想、建议等等）。

暂无。

指导教师批阅意见：
成绩评定：
指导教师签字：
2024 年    月    日
备注：

- 注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
- 2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。