



面向对象 JML 系列第一次代码作业指导书



写在前面：请勿提交官方包代码，仅提交自己实现的类。更不要将官方包的 JML 或代码粘贴到自己的类中，否则以作弊、抄袭论处。



第一部分：训练目标



本次作业，需要完成的任务为实现简单社交关系的模拟和查询，学习目标为 **入门级JML规格理解与代码实现**。

第二部分：预备知识

需要同学们了解基本的JML语法和语义，以及具备根据JML给出的规格编写Java代码的能力。

需要同学们了解OK方法，以及具备为较简单规格写OK测试的能力。

第三部分：题目描述

一、作业基本要求

- 社交网络的整体框架官方已经给出了JML表述并提供了相应接口。同学们需要**阅读JML规格，依据规格实现自己的类和方法**。

具体来说，各位同学需要新建两个类 `MyPerson`、`MyNetwork`（仅举例，具体类名可自行定义并配置），并实现相应的接口方法，每个方法的代码实现需要**严格满足**给出的JML规格定义。

- 阅读指导书中关于异常类行为的描述，通过**继承官方提供的各抽象异常类，实现自己的异常类**。

抽象异常类已在官方包内给出，**这一部分没有提供 JML 规格**，各位同学需要仔细阅读指导书中关于异常类的详细描述，结合样例理解其行为，然后继承这些抽象类实现自己的异常类，使其 `print()` 方法能够正确输出指定的信息。

- 为了检验大家对于规格的理解，请同学们为部分方法编写OK测试，检查前置条件 `requires`、后置条件 `ensures` 和 `pure` 类方法等限制。





< 第九次作业 - 2023面向对象设计与构造



Person类

Person 的具体接口规格见官方包的开源代码，此处不加赘述。



除此之外， Person 类必须实现一个构造方法



```
1 public class MyPerson implements Person {  
2     public MyPerson(int id, String name, int age);  
3 }
```



构造函数的逻辑为生成并初始化 Person 对象。

Person 的属性：

id：对当前 Network 中所有 Person 对象实例而言独一无二的 id

name：姓名

age：年龄

此外，Person类内部还维护两个属性数组 acquaintance 和 value ,具体表述参见 Person 类。

请确保构造函数正确实现，且类和构造函数均定义为 public。 Runner 内将自动获取此构造函数进行 Person 实例的生成。

Network类

Network 的具体接口规格见官方包的开源代码，此处不加赘述。

除此之外， Network 类必须实现一个构造方法

```
1 public class MyNetwork implements Network {  
2     public MyNetwork();  
3 }
```

构造函数的逻辑为生成一个 Network 对象。

此外，Network类内部还维护属性数组 people ,具体表述参见 Network 类。

请确保构造函数正确实现，且类和构造函数均定义为 public。 Runner 内将自动获取此构造函数进行 Network 实例的生成。





< 第九次作业 - 2023面向对象设计与构造



每个异常类必须正确实现指定参数的构造方法。



除此之外，还需要实现一个无参的 `print()` 方法。 `print()` 方法需将包含计数结果的指定信息输出到标准输出中， `Runner` 类会自动调用该方法。为实现计数功能，同学们可以在异常类中自定义其他属性、方法（例如：可以构造一个计数器类，其实例作为每个异常类的 `static` 属性，管理该类型异常的计数）。

详细的异常类行为请参考代码和样例以及官方包中的 JML 约束，大致要求如下：

- `PersonIdNotFoundException`:

```
1 public class MyPersonIdNotFoundException extends PersonIdNotFc
2     public MyPersonIdNotFoundException(int id);
3 }
```

- 输出格式： `pinf-x, id-y` , `x` 为此类异常发生的总次数, `y` 为该 `Person.id` 触发此类异常的次数
- 当 `network` 类某方法中有多个参数都会触发此异常时，只以第一个触发此异常的参数抛出一次异常
 - 比如对方法 `func(id1, id2)` , 如果 `id1` 和 `id2` 均会触发该异常，则仅认为“ `id1` 触发了该异常”，且只计算触发一次该异常。

- `EqualPersonIdException`:

```
1 public class MyEqualPersonIdException extends EqualPersonIdExc
2     public MyEqualPersonIdException(int id);
3 }
```

- 输出格式： `epi-x, id-y` , `x` 为此类异常发生的总次数, `y` 为该 `Person.id` 触发此类异常的次数

- `RelationNotFoundException`:

```
1 public class MyRelationNotFoundException extends RelationNotFc
2     public MyRelationNotFoundException(int id1, int id2);
3 }
```





< 第九次作业 - 2023面向对象设计与构造



发此类异常的次数, z 为 Person.id2 触发此类异常的次数

- id1 , id2 按数值大小排序, 由小到大输出

- EqualRelationException:

```
1 public class MyEqualRelationException extends EqualRelationExc
2     public MyEqualRelationException(int id1, int id2);
3 }
```

- 输出格式: er-x, id1-y, id2-z , x 为此类异常发生的总次数, y 为 Person.id1 触发此类异常的次数, z 为 Person.id2 触发此类异常的次数
- id1 , id2 按数值大小排序, 由小到大输出
- id1 与 id2 相等时, 视为该 id 触发了一次此类异常, 其中 id 满足 $id == id1 == id2$

三、需要书写OK测试的方法

本次作业中, 同学们需要对 Network 类中的 query_triple_sum 方法书写OK测试。

你需要调用官方包实现的 queryTripleSumOKTest 接口, 编写测试代码, 满足在给定前后状态的情况下, JML推出的结论和OKTest的结论一致。具体而言, 倘若给定的前后状态不符合规格, 那么OKTest应该返回False; 反之, 若给定的前后状态符合规格, 则OKTest应该返回True。其中, beforeData 和 afterData 表示调用 queryTripleSum 方法前后的"状态" (在第五部分有详细说明) , result 表示该方法得到的输出结果。

```
1 public boolean queryTripleSumOKTest(
2     HashMap<Integer, HashMap<Integer, Integer>> beforeData,
3     HashMap<Integer, HashMap<Integer, Integer>> afterData,
4     int result);
```

在OKTest中, 你需要对JML的全部内容进行检查, 除了检验requires和ensures, 还有 pure、assignable语句等等。例如, 对于一个pure方法, 调用方法前后的状态应该一致, 如果前后状态不一致, 那么我们认为这不符合给定的JML。

第四部分: 设计建议

推荐各位同学在课下测试时使用 Junit 单元测试来对自己的程序进行测试。





第五部分：输入输出



本次作业将会下发输入输出接口和全局测试调用程序，前者用于输入输出的解析和处理，后者会实例化同学们实现的类，并根据输入接口解析内容进行测试，并把测试结果通过输出接口进行输出。

输出接口的具体字符格式已在接口内部定义好，各位同学可以阅读相关代码，这里我们只给出程序黑箱的字符串输入输出。

关于 `main` 函数内对于 `Runner` 的调用，参见以下写法。

```
1 package xxx;
2
3 import com.oocourse.spec1.main.Runner;
4
5 public class xxx {
6     public static void main(String[] args) throws Exception {
7         Runner runner = new Runner(MyPerson.class, MyNetwork.class);
8         runner.run();
9     }
10 }
```

规则

- 输入一律在标准输入中进行，输出一律在标准输出。
- 输入内容以指令的形式输入，一条指令占一行，输出以提示语句的形式输出，一句输出占一行。
- 输入使用官方提供的输入接口，输出使用官方提供的输出接口。

指令格式一览(括号内为变量类型)

- 基本格式： 指令字符串 参数1 参数2 ...





< 第九次作业 - 2023面向对象设计与构造



```
2   add_relation id(int) id(int) value(int)
3   query_value id(int) id(int)
4   query_circle id(int) id(int)
5   query_block_sum
6   query_triple_sum
7   query_triple_sum_ok_test
```



实际上为了减小输入量，真实输入为简写：



指令	简写
add_person	ap
add_relation	ar
query_value	qv
query_circle	qci
query_block_sum	qbs
query_triple_sum	qts
query_triple_sum_ok_test	qtsok

样例

#	标准输入	标准输出
1	ap 1 2 1 ap 2 1 2 ar 1 2 1 ap 1 2 1 ap 2 1 2 ar 1 2 1	Ok Ok Ok epi-1, 1-1 epi-2, 2-1 er-1, 1-1, 2-1
2	ap 1 2 1 ap 2 1 2 ar 1 2 1 qts qbs qci 1 1	Ok Ok Ok 0 1 true
3	ap 1 2 1	Ok



🏠

☰

🖼️

👤

?

👤

<

第九次作业 - 2023面向对象设计与构造

🔔

	ar 1 2 1 qv 1 1	Ok 0
4	qv 1 2 qv 2 1 ap 1 jack 100 ap 1 mark 100 ap 2 mark 100 qv 1 2 qv 2 1 ar 1 2 100 ar 1 2 200	pinf-1, 1-1 pinf-2, 2-1 Ok epi-1, 1-1 Ok rnf-1, 1-1, 2-1 1 rnf-2, 1-2, 2-2 2 Ok er-1, 1-1, 2-1
5	qtsok 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 0 qtsok 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 1 qtsok 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 3 1 2 3 1 2 3 2 1 2 3 4 1 2 4 0	true false false

关于判定

数据基本限制

指令条数不多于 10000 条

ap

- name(String) 长度不超过 10
- age(int) 值在 [0,200] 中

ar

- value(int) 值在 [1,100] 中

qtsok

- 单个测试点中不多于10条
- 不和其他指令出现在同一个测试点中
- 在官方包的 Runner 中，每测试一条 qtsok 都会新建一个 Network ，因此无需担心多条



< 第九次作业 - 2023面向对象设计与构造



人有id为 `id1` 的acquaintance, 且对应的value为 `v1`

- 给出的acquaintance的id一定存在, 且不会同自身id相同。即不会出现id为 `id1` 的人有id为 `id2` 的acquaintance, 但id为 `id2` 的人不存在或 `id2` 等于 `id1` 的情况

- afterData满足

- 人数在 $[0, 10]$ 中
- 给出的acquaintance的id一定存在, 且不会同自身id相同。即不会出现id为 `id1` 的人有id为 `id2` 的acquaintance, 但id为 `id2` 的人不存在或 `id2` 等于 `id1` 的情况

qtsok参数说明

原始输入:

```
qtsok 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 3 1 2 3 1 2 3 2 1 3 3 4 1 2 4 0
```



解释说明:

将qtsok后跟的参数展开成如下形式

```
1      # beforeData
2      3      # 有三个人
3      1 2 3      # 三个人id分别为 1 2 3
4      1 2 3      # 第1个人有1个熟人, id为2、value为3
5      2 1 3 3 4      # 第2个人有2个熟人, 第1个id为1、value为3, 第2个id
6      1 2 4      # 第3个人有1个熟人, id为2、value为4
7      # afterData
8      3
9      1 2 3
10     1 2 3
11     2 1 3 3 4
12     1 2 4
13     # result
14     0
```



ok方法的测试接口的格式说明:

```
public boolean queryTripleSumOKTest(
```




result 为 queryTripleSum 方法的返回结果。beforeData和afterData均为两层HashMap，在queryTripleSum这个方法中，由于只涉及person类，所以beforeData/afterData仅包含Person类在运行queryTripleSum前/后的状态，默认仅需要考虑给出的数据是否符合规格。

具体来说，外层HashMap以personId作为key，对应value为内层HashMap；内层HashMap以该person的AcquaintanceId为key，对应value为该person与acquaintance之间的value。

HashMap<personId, HashMap<该person的acquaintance的id, 对应的value> >

互测数据限制

- 指令条数不多于 1000 条
- 不出现 qtsok
- 其他同公测限制相同

测试模式

公测和互测都将使用指令的形式模拟容器的各种状态，从而测试各个类、接口的实现正确性，即是否满足 JML 规格的定义或者指导书描述。**可以认为，只要所要求的三个类的具体实现严格满足 JML，同时异常类的实现符合指导书描述，就能保证正确性，但是不保证满足时间限制。**

任何满足规则的输入，程序都应该保证不会异常退出，如果出现问题即视为未通过该测试点。

程序的最大运行 cpu 时间为 10s，虽然保证强测数据有梯度，但是还是**请注意时间复杂度的控制。**

第六部分：提示与警示

一、提示

- 请同学们参考源码，注意本单元中除了OKTest方法外，一切叙述的讨论范围实际限定于全局唯一的Network实例中。OKTest方法每条指令会新建一个临时的Network，仅用于测试当前这条OKTest指令。
- 本次作业中可以自行组织工程结构。任意新增 java 代码文件。只需要保证题目要求的几个类的继承与实现即可。





二、警示



- **不要试图通过反射机制来对官方接口进行操作**，我们有办法进行筛查。此外，在互测环节中，如果发现有人试图通过反射等手段 hack 输出接口的话，请私聊助教进行举报，**经核实后，将直接作为无效作业处理。**

