





介绍

测验

实验预习题

评测

互测 BUG 修复

讨论



面向对象规格化设计系列第二次代码作业指 导书



第零部分: 提交要求

 \vdash

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,否则可能以作弊、抄袭论处。

请保证提交项目的顶层目录至少存在两个文件夹: src 和 test (命名需严格与此保持 一致),请将作业的**功能代码**文件存放于 src 文件夹下,同时将相关 Junit 测试代码文件 存放于 test 文件夹下,以保证评测的正常进行(评测时只会针对 src 目录下的文件进 行**程序功能**的评测以及**代码风格**检测,也就是说, test 目录下的 Junit 测试代码风格**不** 会被检测)。参考目录结构如下:

```
-src
 |- MainClass.java
  |- ...
-test
  |- Test.java
  - . . .
```

注意: 为了通过 Junit 测试的编译,请大家实现课程组提供的接口时不要分包(在 src 下 创建子目录),而是将所有实现接口的类都直接放在 src 目录下,否则本地运行正常的 Junit 测试类代码在评测机上会无法找到课程组提供的待测试类文件。

第一部分: 训练目标

本次作业中、需要完成的任务为实现简单社交关系的模拟和查询、学习目标为对规格化开 发(以入门级 JML 规格为例)的理解与相应的代码实现。

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

个人中心

所有课程



我的图床



课程团队









或者对象的修改,但是不可以有如下的 side effect,具体体现为:

- 1. 不可在任何容器或对象中增加 JML 没有要求加入的对象。
- 2. 不可在任何容器或对象中删除 JML 没有要求删除的对象。
- **
- 3. 不可对 JML 描述中涉及之外的对象或涉及对象中的非涉及属性进行内容的修改,即 JML 涉及之外的对象或属性的 object representation(对应二进制序列)应该前后一致。



 \vdash

第三部分: 题目描述

一、作业基本要求

本次作业要求同学们升级已有的社交网络,新增"公众号"和"文章"的概念。

● 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,各位同学需要在上一次作业的基础上,一方面继续维护已有的 Person 、 Network 、 Tag 类,根据官方包的调整新增一些查询方法;另一方面新建一个 OfficialAccount 类,并实现官方包中提供的 OfficialAccountInterface 接口,最终 类中每个方法的代码实现都需要**严格满足**接口中给出的 JML 规格定义。

● 阅读指导书中关于**异常**行为的描述,结合官方包中提供的异常类的 javadoc,体会异常处理的流程。

异常类已在官方包内给出,这一部分**没有提供 JML 规格**,而是提供了一些相对标准的 文档注释来向大家介绍类或方法的功能和参数的含义。各位同学需要仔细阅读指导书 中关于异常类的详细描述,恰当地使用这些异常类,正确处理我们规定的各种异常情况,并保证所有的 print() 方法能够正确输出指定的信息。

此外,还需要同学们在**主类**中通过调用官方包的 Runner 类,并载入自己实现的 Person 、 Network 、 Tag 、 OfficialAccount 类,来使得程序完整可运行,具体形式下文中有提示。

二、类规格要求

注意:

• 同学们需要保证实现 NetworkInterface 接口的类命名为 Network ,实现 PersonInterface 接口的类命名为 Person ,实现 TagInterface 接口的类命名为

Tag 守珂 OfficialAccountThtonface 控口的米命夕为 OfficialAccount

个人中心



所有课程



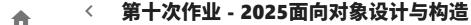
我的图床

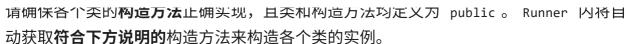


课程团队









Person 类 【Modify】

- 构造方法的要求与上次作业保持一致,仅需要新增一个属性数组 receivedArticles ,具 体表述参见 PersonInterface 接口。
- Network 类 【Modify】
- 构造方法的要求与上次作业保持一致,仅需要新增三个属性数组 accounts 、 articles \vdash 和 articleContributors ,具体表述参见 NetworkInterface 接口。

OfficialAccount 类

构造方法,用以生成和初始化一个 Official Account 对象:

```
public class OfficialAccount implements OfficialAccountInterface
1
        public OfficialAccount(int ownerId, int id, String name);
2
3
   }
```

属性:

- ownerId: 公众号主人(Person)的id
- id: 在整个程序运行过程中的所有时刻,在当前 Network 中出现过的所有 OfficialAccount 对象中独一无二的 id
- name: 公众号名称

此外, OfficialAccount 内部还需要维护三个**属性数组** followers 、 articles 和 contributions,具体表述参见 OfficialAccountInterface 接口。

异常类

本次作业的官方包中新增了8个异常类,同学们可以直接调用,官方包会捕获这些异常, 并处理异常输出(具体逻辑见官方包中的 Runner 类)。每个异常类的具体说明在官方包 中由 javadoc 给出,在此不作赘述,请同学们在实现过程中仔细阅读官方包中的相关代码 与说明,正确传递参数,同时 **将官方包中对类和方法的自然语言描述与各个接口中的** JML 进行对比,思考各自的优劣。

重要编写 Lunit 单元测试的专注

个人中心



所有课程



我的图床



课程团队









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。

评测时我们会使用若干正确代码与错误代码进行测试,保证错误代码仅 queryCoupleSum 出现错误,且**保证不涉及** Tag **类和 OfficialAccount 类有关的错误**,其余官方包要求方 法均正确实现,需要同学们编写的单元测试正确判断代码的 queryCoupleSum 方法是否出 现错误。

注意:

 \vdash

在 JUnit 测评时给出的样例中, Person 类会提供 boolean strictEquals(PersonInterface person) 方法, Network 类会提供 PersonInterface[] getPersons() 方法供同学调用。

person1.strictEquals(person2) 返回一个布尔值,表示两个 PersonInterface 对象是否相等。实现逻辑是:对于非容器的基本数据类型采用 == 、对象类型使用 equals() 进行比较。设计此函数的目的是便于同学们在检查方法执行前后 PersonInterface 对应的具体实现对象状态是否发生改变(被修改)。本方法仅仅是为了给同学提供便利,避免同学们进行大量遍历检查 PersonInterface 对象的各属性在方法执行前后是否发生变化,同学们可以选择不调用此方法,在 JUnit 测试时使用官方包中已有的方法进行检查。

getPersons()返回一个 PersonInterface 数组,表示 Network 中全体 PersonInterface 元素集合,使用浅拷贝实现。

补充说明:

• 本次作业的 JUnit 测评样例中, Network 类**没有给出**获取 accounts , articles , articleContributors 三个容器的方法,同时保证所有样例**没有**涉及这三个容器的错误,但是推荐大家自行设计测试时能够尽可能全面地测试。

第四部分:设计建议

推荐各位同学在课下测试时使用 Junit 单元测试来对自己程序的全部方法进行测试。

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

个人中心

所有课程



我的图床



课程团队





 \vdash

```
第十次作业 - 2025面向对象设计与构造
        <
        2
            public class xxx {
        3
public static void main(String[] args) throws Exception {
        4
                   Runner runner = new Runner(Person.class, Network.class,
5
                   runner.run():
        6
        7
            }
        8
```

输入输出格式

输入部分,一行或多行一条指令,形如 op arg1 arg2 ... ,表示指令类型和参数,部分 指令可能跟随若干行额外信息,具体格式见下。

输出部分,每条指令对应一行输出,为指令的执行结果或发生的异常。

输入输出实际由官方包进行处理。

指令格式一览

本次作业新增指令如下(括号内为变量类型):

```
create official account person id(int) account id(int) account n
1
    delete_official_account person_id(int) account_id(int)
2
    contribute_article person_id(int) account_id(int) article_id(int
3
    delete_article person_id(int) account_id(int) article_id(int)
4
    follow_official_account person_id(int) account_id(int)
5
6
7
    query_shortest_path id1(int) id2(int)
    query_best_contributor account_id(int)
8
    query_received_articles person_id(int)
9
    query_tag_value_sum person_id(int) tag_id(int)
10
    query_couple_sum
11
```

上次作业中的指令仍然有效 (括号内为变量类型):

```
add person id(int) name(String) age(int)
        add relation id1(int) id2(int) value(int)
个人中心
```

我的图床



课程团队





```
第十次作业 - 2025面向对象设计与构造
        <
             add_to_tag person_id1(int) person_id2(int) tag_id(int)
         7
del_from_tag person_id1(int) person_id2(int) tag_id(int)
         8
         9
query value id1(int) id2(int)
        10
             query_circle id1(int) id2(int)
        11
             query_triple_sum
        12
             query tag age var person id(int) tag id(int)
        13
             query_best_acquaintance id(int)
        14
        15
\vdash
             load network n(int)
        16
             load_network_local n(int) file(String)
        17
```

每条指令的执行结果为其对应方法的输出(void 方法则为 Ok),方法的对应详见官方包。

其中, load_network 和 load_network_local 指令为复合指令,由官方包实现。

load_network 指令后跟随 n+2 行:

- 第一行 $n \land id$: id_1, id_2, \ldots, id_n , 第二行 $n \land name$: $name_1, name_2, \ldots, name_n$, 第三行 $n \land age$: $age_1, age_2, \ldots, age_n$, 表示添加 $n \land Person$, 第 $i \land Ame_i \land age_i$;
- 接下来 n-1 行,第 i 行 i 个 $value(value \geq 0)$,其中第 j 个表示 id_{i+1} 和 id_j 对应 Person 间添加 value 的关系 (value > 0) 或不添加关系 (value = 0)。

load_network_local 指令与其类似,但仅占一行,而从 file 文件中读取所需 n+2 行数据。

此二指令会调用 add_person 和 add_relation 添加 Person 和关系,具体实现方式参考官方包。保证复合指令不会出现异常。

其余指令均为一行的简单指令。

指令的简称

实际评测使用的输入为简称,但官方包提供简称和全称的识别方式,在测试时同学们可以自由选择简称与全称。

指令	简称
create_official_account	coa

个人中心



所有课程



我的图床



课程团队





















contribute_article	ca
delete_article	da
follow_official_account	foa
query_shortest_path	qsp
query_best_contributor	qbc
query_received_articles	qra
query_tag_value_sum	qtvs
query_couple_sum	qcs
add_person	ар
add_relation	ar
modify_relation	mr
add_tag	at
del_tag	dt
add_to_tag	att
del_from_tag	dft
query_value	qv
query_circle	qci
query_triple_sum	qts
query_tag_age_var	qtav
query_best_acquaintance	qba
load_network	ln
load_network_local	Inl

样例

Case 1







▲ 我的图床



课程团队







ar 1 0 10 ar 2 0 10

qcs qsp 1 0

mr 1 0 -99

qsp 1 0

标准输出

0k

0k

0k

Ok

0k

0k

1

0k

1

2

Case 2

标准输入

```
ap 0 NagasakiSoyo 16
ap 1 ChihayaAnon 16
ap 2 MisumiUika 16
foa 2 0
coa 0 0 CRYCHIC
foa 0 0
foa 2 0
ca 0 0 HelloEveryone
qbc 0
```

qra 1

个人中心



所有课程



我的图床



课程团队











标准输出



0k

0k

0k

oainf-1, 0-1

0k

 \vdash

epi-1, 0-1

0k

0k

0

None

0

doapd-1, 0-1, 1-1 dapd-1, 0-1, 1-1

Case 3

标准输入

```
ln 5
   -2 -1 0 1 2
   NagasakiSoyo ChihayaAnon KanameRana TakamatsuTomori ShiinaTaki
   16 16 16 16 16
   10
   10 10
   10 10 10
   10 10 10 10
   coa -1 1 MyGo!!!!!
   foa 1 1
   foa -2 1
   foa 2 1
   ca -1 1 0 AnonTokyo
   ca -2 1 1 It'sMyGo!!!!!
   ca -1 1 2 SoyoLove~
   ca 0 1 1 meow
   ca 1 1 1 GUGUGAGA
abc 1
```









我的图床











P

att 0 0 5
att 1 0 5
att 2 0 5
qtvs 0 5

标准输出

?

 \rightarrow

Ok Ok Ok

Ok Ok Ok Ok

Ok
eai-1, 1-1
eai-2, 1-2
-1
2 1 0

Ok Ok Ok epi-1, 0-1 Ok

注意

120

以上样例对本次作业中涉及到的指令进行了较全面的覆盖,但是并不足以测试到所有情况,请大家尽量**全面分析、充分测试**。

本单元的训练栏目中为大家提供了一些测试方法的参考,欢迎大家自主学习、积极思考(不强制要求完成),如果有相关想法也欢迎在作业讨论区分享。

数据范围

公测数据限制

• 指令条数不多于 10000 条。

. 能士:1十一、世国市

个人中心



所有课程



我的图床



课程团队









- 1 ≤ n ≤ 300 且为整数。
- 若出现 $load_network$ 指令,则其一定为测试点中的第一条指令,且其中的 id 不重。
- 不出现 load_network_local 指令。

三 互测数据限制

- ?
- 指令条数不多于 3000 条。
- 1 ≤ n ≤ 100 且为整数。
- \rightarrow
- 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,只要所要求的所有类的具体实现严格满足 JML,同时异常处理符合指导书和官方包的描述,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 **10**s,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: 补充说明

关于提交代码部分的文件结构:

src 目录下包含主入口类(例如 MainClass.java),同学们实现官方包接口的类(Network , Person , Tag , OfficialAccount)以及同学们可能自行设计的辅助类。

请注意:提交的时候请不要在 src 目录下包含官方包,同学们自己实现的官方包接口的类的命名还请按照(Network , Person , Tag , OfficialAccount)命名,并直接放在 src 目录下,不要嵌套子目录,否则和测试程序一起编译的时候会无法通过。

test 目录下设置测试类(例如 Test.java),以及同学们可能自己设计的辅助类。

关于评测机制:

受到系统限制,我们只能统一编译 src 文件夹和 test 文件夹,同学提交代码之前请确

个人中心



所有课程



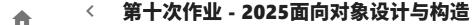
我的图床



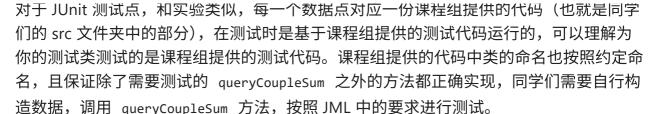
课程团队











注意:使用 JUnit 单元测试的过程中请<mark>避免使用 assert() 进行逻辑断言</mark>,而是使用 JUnit 测试框架下的断言类 **org.junit.**Assert 中的对应方法进行判定。

→ 关于 JUnit 评测限制:

课程组提供的 strictEquals 方法对于非容器的基本数据类型采用 == 、对象类型使用 equals() 进行比较。但是由于 Person 重写了 equals 方法,实际上对于 acquaintance 而言判断的是执行方法前后 acquaintance 中 Person 元素的 id 是否一致。 value 不涉及这个问题。

对于 queryCoupleSum 的正确性检查部分,课程组提供的错误测试点的 bug 比较明显,不会出现需要用很刁钻的数据才能覆盖的情况。

再次强调,Network.java 文件和 Person.java 文件一定要**直接**放在 src 目录下,否则本地正常工作的代码在评测机上运行 JUnit 评测时会找不到课程组提供的代码文件。

说明: JUnit评测只在公测存在,互测和强测不存在。

关于编译的说明:

由于评测是 src 文件夹和 test 文件夹统一编译的,同学们如果在 test 文件夹中的测试类使用了课程组提供的 getPersons 和 strictEquals 方法,请在src文件夹下的 Network 类中实现 getPersons 方法(不必正确实现该方法,写任意能通过编译的内容都可); Person 类中写出 strictEquals 方法(同上)。以下是一种解决方式的参考:

```
public class Network implements NetworkInterface {
    //...同学自己编写过的其他属性和方法,此处省略

    //声明方法即可,内容任意,仅仅是为了通过评测时的编译,不会被调用
    public PersonInterface[] getPersons() { return null; }

}
```

public class Person implements PersonInterface {

个人中心

所有课程



我的图床



课程团队









業 第七部分:提示与警示

?

一、提示

- \rightarrow
- 请同学们参考源码,注意本单元中一切叙述的讨论范围实际限定于全局唯一的 Network 实例中
- 本次作业中可以自行组织代码结构。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。
- 关于本次作业容器类的设计具体细节,本指导书中均不会进行过多描述,请自行去官方包仓库中查看接口的规格,并依据规格进行功能的具体实现,必要时也可以查看Runner 的代码实现。
- 仓库地址: 第十次作业公共仓库

二、警示

• **请勿试图对官方接口进行操作**。此外,在互测环节中,如果发现有人试图 hack 输出接口,请联系助教,经核实后,**将直接作为无效作业处理**。











