

**2022年春季学期**

**计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 胡宏彬 |
| 学号 | 2021111204 |
| 班号 | 2137101 |
| 电子邮件 | 1953802638@qq.com |
| 手机号码 | 17816078651 |

**目录**

[1 实验目标概述 1](#_Toc17441)

[2 实验环境配置 1](#_Toc26217)

[2.1 配置Java8 1](#_Toc1421)

[2.2 下载IDEA、Junit 2](#_Toc14298)

[3 实验过程 4](#_Toc19637)

[3.1 Poetic Walks 5](#_Toc14862)

[3.1.1 Get the code and prepare Git repository 5](#_Toc8373)

[3.1.2 Problem 1: Test Graph <String> 5](#_Toc2394)

[3.1.3 Problem 2: Implement Graph <String> 6](#_Toc16272)

[3.1.3.1 Implement ConcreteEdgesGraph 6](#_Toc15021)

[3.1.3.2 Implement ConcreteVerticesGraph 10](#_Toc25993)

[3.1.4 Problem 3: Implement generic Graph<L> 15](#_Toc28680)

[3.1.4.1 Make the implementations generic 15](#_Toc20546)

[3.1.4.2 Implement Graph.empty() 17](#_Toc3491)

[3.1.5 Problem 4: Poetic walks 19](#_Toc25538)

[3.1.5.1 Test GraphPoet 19](#_Toc32079)

[3.1.5.2 Implement GraphPoet 20](#_Toc18650)

[3.1.5.3 Graph poetry slam 21](#_Toc6614)

[3.1.6 使用Eclemma检查测试的代码覆盖度 22](#_Toc22397)

[3.1.7 Before you’re done 22](#_Toc27237)

[3.2 Re-implement the Social Network in Lab1 24](#_Toc29958)

[3.2.1 FriendshipGraph类 24](#_Toc22088)

[3.2.2 Person类 25](#_Toc3459)

[3.2.3 客户端main() 26](#_Toc31613)

[3.2.4 测试用例 27](#_Toc18555)

[3.2.5 提交至Git仓库 27](#_Toc26348)

[4 实验进度记录 29](#_Toc23757)

[5 实验过程中遇到的困难与解决途径 29](#_Toc9023)

[6 实验过程中收获的经验、教训、感想 30](#_Toc17559)

[6.1 实验过程中收获的经验和教训（必答） 30](#_Toc31672)

[6.2 针对以下方面的感受（必答） 30](#_Toc22856)

# 实验目标概述

本次实验训练抽象数据类型（ADT）的设计、规约、测试，并使用面向对象

编程（OOP）技术实现 ADT。具体来说：

l 针对给定的应用问题，从问题描述中识别所需的 ADT；

l 设计 ADT 规约（pre-condition、post-condition）并评估规约的质量；

l 根据 ADT 的规约设计测试用例；

l ADT 的泛型化；

l 根据规约设计 ADT 的多种不同的实现；针对每种实现，设计其表示

（representation）、表示不变性（rep invariant）、抽象过程（abstraction

function）

l 使用 OOP 实现 ADT，并判定表示不变性是否违反、各实现是否存在表

示泄露（rep exposure）；

l 测试 ADT 的实现并评估测试的覆盖度；

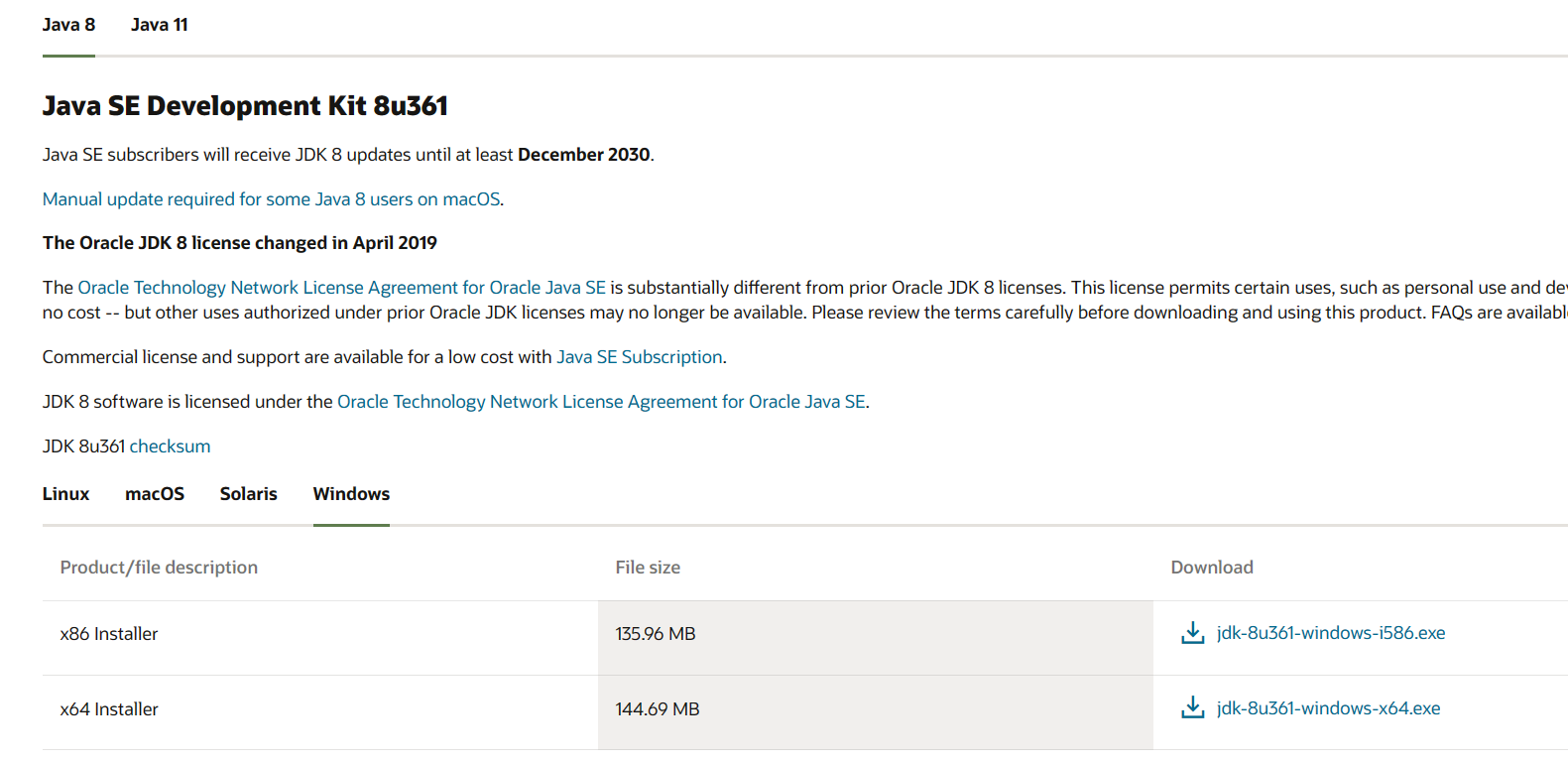
l 使用 ADT 及其实现，为应用问题开发程序；

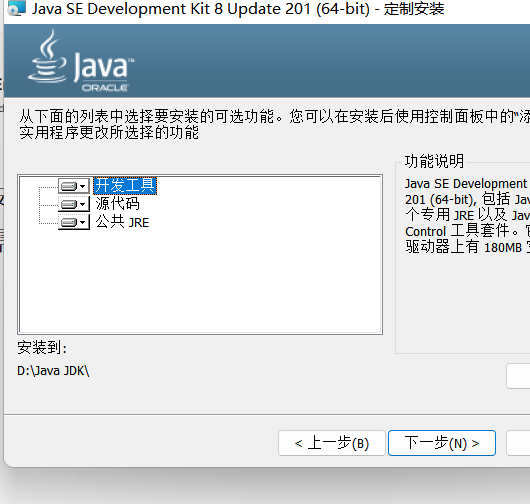
l 在测试代码中，能够写出 testing strategy 并据此设计测试用例。

# 实验环境配置

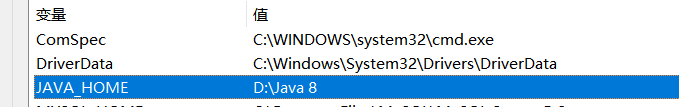
## 配置Java8

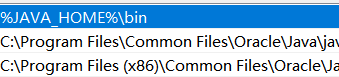
**2.1.1下载java8：从Oracle官网找到JAVA模块，选择JAVA8的Windows版本，根据电脑配置选择x64安装包安装，最后打开安装包按提示下载即可。**





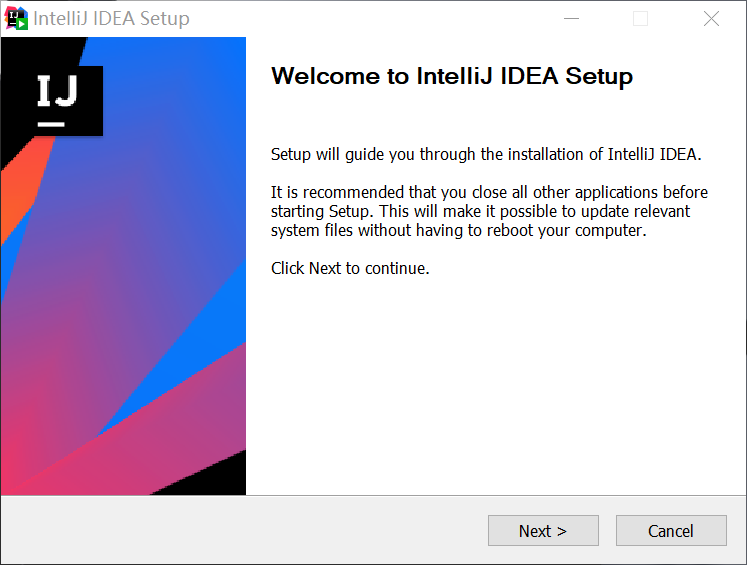
**2.1.2配置java8环境变量：**





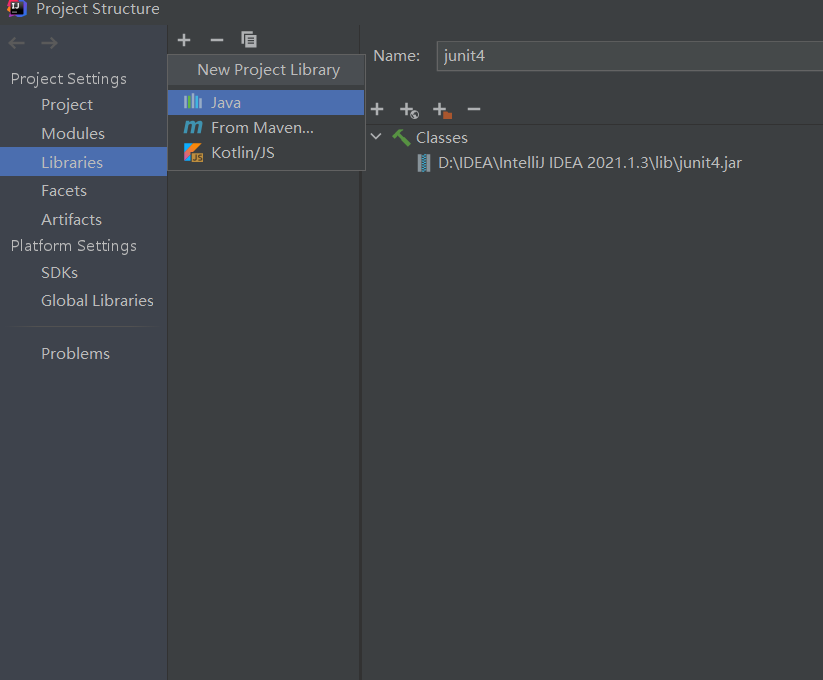
## **下载IDEA、Junit**

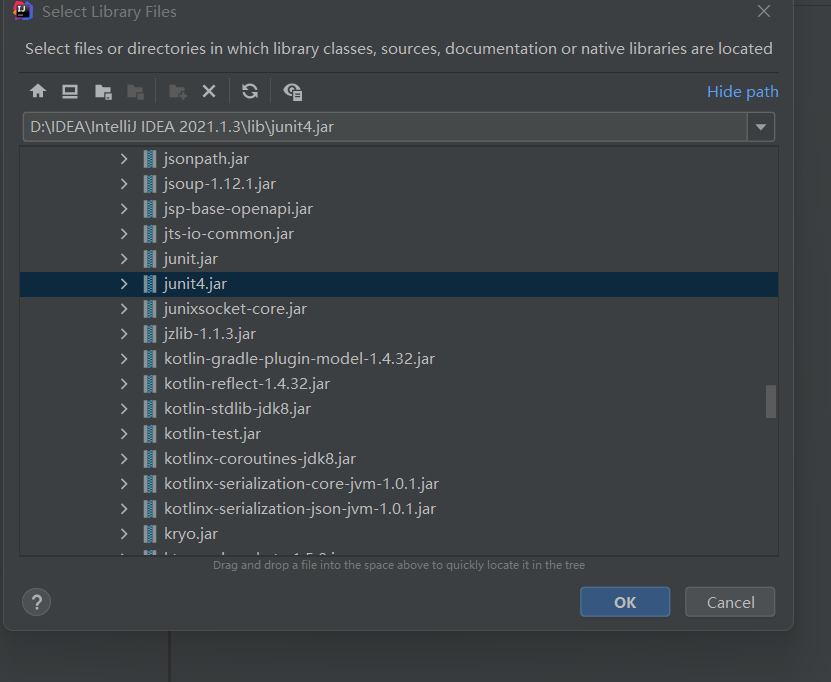
**IDEA下载界面：从官网下载IDEA安装包，按步骤next下载即可。**



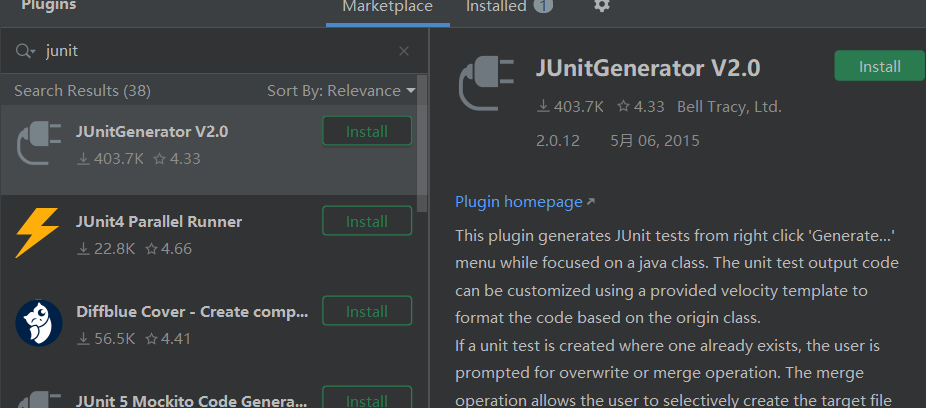
Junit 配置：

从File 选择Project structure，在Libraries点击+号，选择java，找到IDEA的lib目录下的Junit4.jar，添加并应用即可。





选择JUnit4相关插件下载，下载成功后重启IDEA即可



GitHub Lab2仓库的URL地址:[ComputerScienceHIT/HIT-Lab2-2021111204 (github.com)](https://github.com/ComputerScienceHIT/HIT-Lab2-2021111204)。

# 实验过程

请仔细对照实验手册，针对三个问题中的每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## Poetic Walks

本问题集的目的是练习设计、测试和实现抽象数据类型（ADT）。这个问题集侧重于实现可变类型，其中提供了规范。Java集合框架中提供了许多有用的数据结构，它们是objects类型的集合，如：lists, maps, queues, sets等。其中没有提供图的数据结构。该任务将实现图类型，并实现该类型在若干场景下的应用。

Problems 1-3: 实现一个用于表示顶点标签化的可变有向带权图的抽象数据型Graph。该类型的图具有如下的属性：

1） 可变图：边与顶点可被添加与移除

2） 有向边：图中的每条边由源点指向目标点

3） 带权边：每条边带有一个正整数权值

4） 标签化顶点：顶点由某种可变类型标识区分，例如顶点可能具有String类型的命名（names）和Integer类型的编号（ID）

在该问题集中，我们将用两种不同的表示分别实现Graph来练习选择抽象函数，维护表示不变性和防止表示泄露。

Problems 4：实现图类型后，我们将实现GraphPoet类，该类利用一个词系图产生诗节。我们实现的“诗人”将能够接收一串文本，而后生成一段具有某种特殊形式的诗节。

### Get the code and prepare Git repository

在指定的目录打开git Bash,从对应的仓库将http的地址复制下来，然后调用

git clone url 命令将代码clone下来。

### Problem 1: Test Graph <String>

思路：

本部分需要设计GraphInstanceTest类的测试策略，并且对GraphStaticTest进行Junit测试。

测试策略主要根据每一个不同的测试函数对输入空间进行等价类划分，并且以覆盖每个取值最少一次的方法进行测试，尽可能提高测试覆盖率。

GraphStaticTest的测试需要等代码填写完成后才能进行，并且由于是静态测试，将只有一种实现方式，我们只需进行一次。

结果：

测试策略如下：

// add方法 测试策略：

// graph：1、空的graph 2、非空的graph

// vertex：1、加入graph中的vertex 2、未加入graph中的vertex

// set方法 测试策略：

// graph：1、空的graph 2、非空的graph

// source：1、已经加入graph中的source 2、未加入graph中的source

// target：1、已经加入graph中的target 2、未加入graph中的target

// weight：1、weight>0 2、weight=0

// remove方法 测试策略：

// vertex：1、加入graph中的vertex，且与其他结点相连

// 2、加入graph中的vertex，且不与其他结点相连

// 3、未加入graph中的vertex

// vertices方法 测试策略：

// graph：1、空的graph 2、非空的graph

// sources方法 测试策略：

// target：1、target不是graph中的结点

// 2、target是graph中的结点，且有结点指向它

// 3、target是graph中的结点，且没有结点指向它

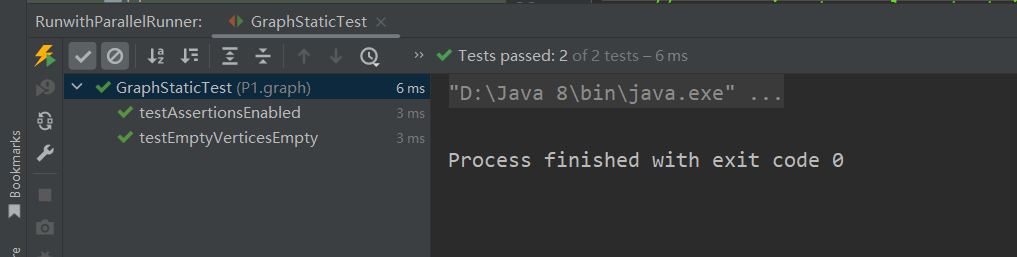
// targets方法 测试策略：

// source：1、source不是graph中的结点

// 2、source是graph中的结点，且它指向了某些结点

// 3、source是graph中的结点，且它不指向任何结点

GraphStaticTest的Junit测试结果：



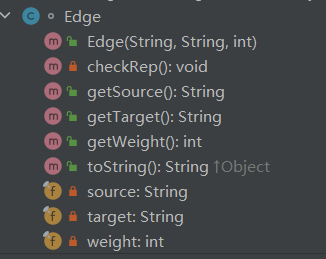
### Problem 2: Implement Graph <String>

以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

#### Implement ConcreteEdgesGraph

1. 完成Edge类

1.1确定Edge的field（属性），分别如下：



1.2确定Edge的方法，分别如下：

Edge ：含参数构造，根据输入的参数构造一条新的边

CheckRep：检测不变量，运用assert测试边的权值是否>0,source 和 target 是否都非空，否则直接截断程序

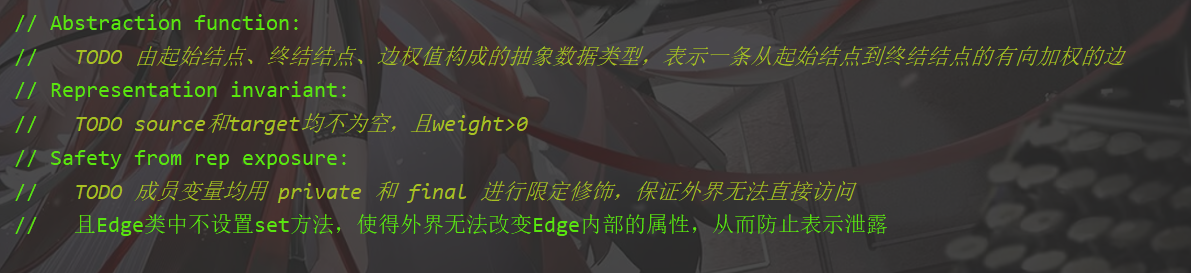
getSource：返回边的起始结点

getTarget：返回边的终止节点

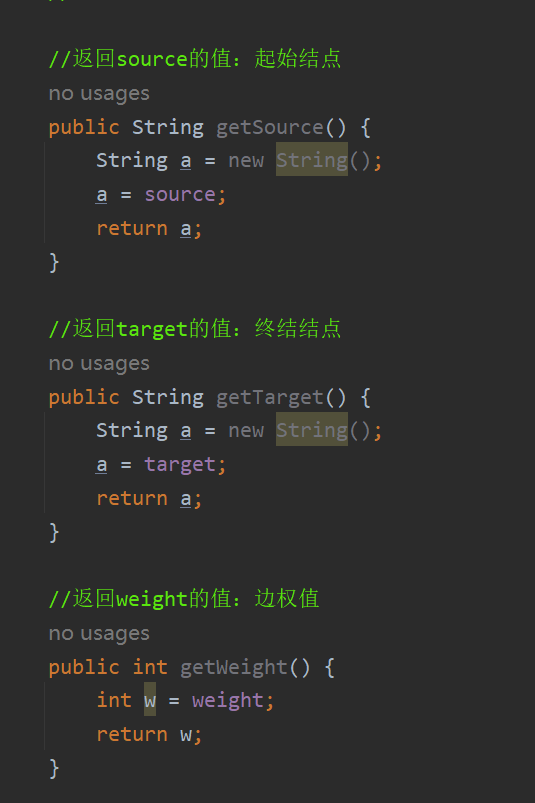
getWeight：返回边的权值

toString：按照格式打印边，格式设置为：source+“->”+target+“权值为：”+weight

1.3：Edge的AF、RI和Safety from rep exposure：

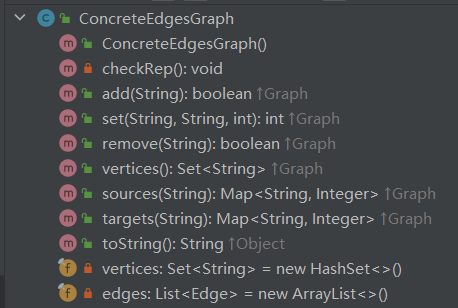


1.4：Edge部分代码截取：



1. 设计ConcreteEdgesGraph类

2.1 ConcreteEdgesGraph类的属性



2.2 ConcreteEdgesGraph类的AF、RI、Safety from rep exposure：

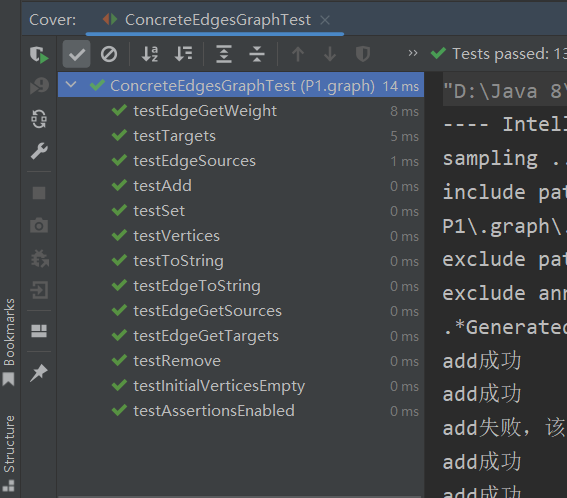


2.3 ConcreteEdgesGraph类的代码和测试代码，截取部分：

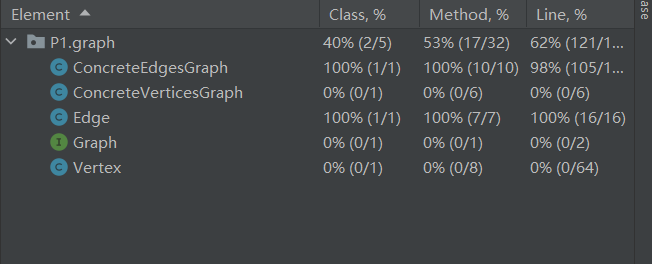


3运行结果：

测试结果：



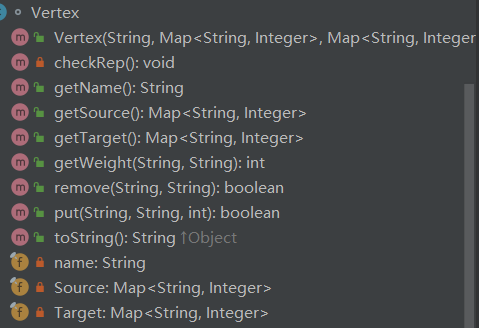
代码覆盖率：



#### Implement ConcreteVerticesGraph

1完成Vertex类：

* 1. 确定Vertex的field，分别如下:



* 1. 确定Vertex的方法，分别如下：

Vertex：含参构造函数，根据输入的参数构造一个点

checkRep:检查不变量，当name值为空或为null、Source和Target的键有与name相同的，值有小于等于0的时候，阶段程序。

getName:返回点的名字

getSource:返回以该点为起始的（终结点，权值）集合

getTarget:返回以该点为终结的（起始点，权值）集合

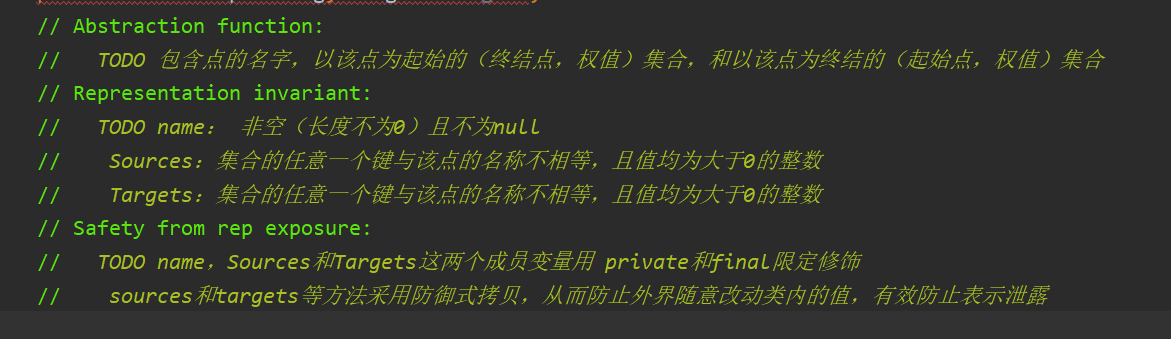
getWeight:返回某条边的权值

remove:删除某条边

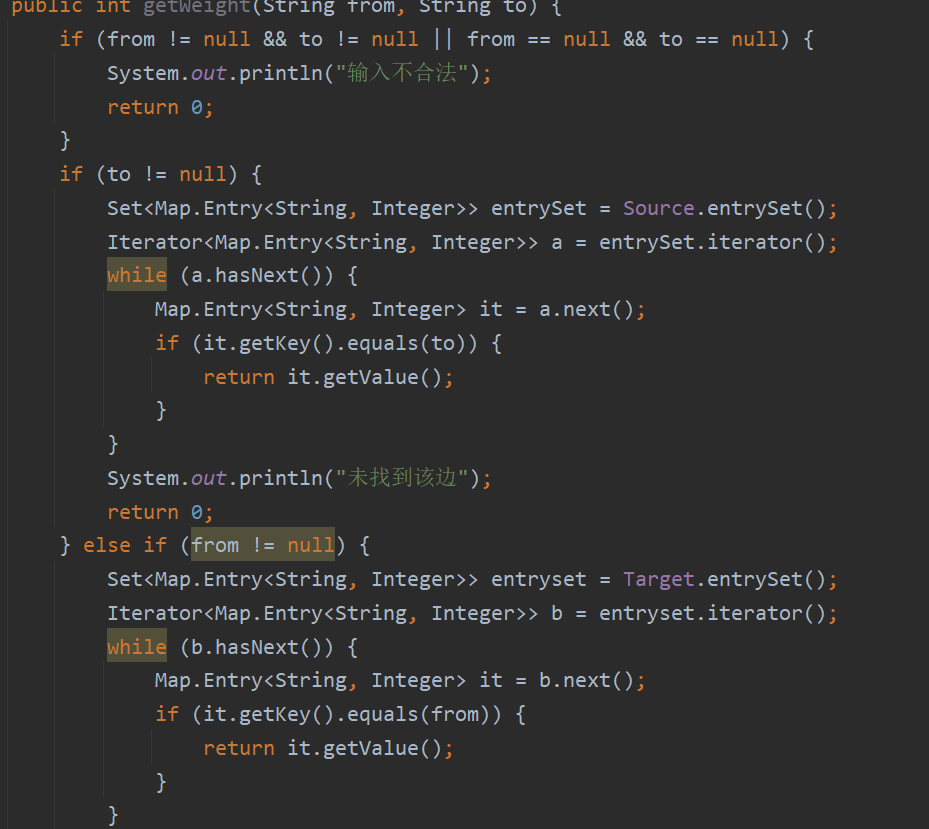
put：添加某条边或者改变某条边的权值

toString:按指定格式打印带有该点所有属性的字符串并返回该字符串

* 1. 确定Vertex的RI,AF,Safety from rep exposure:

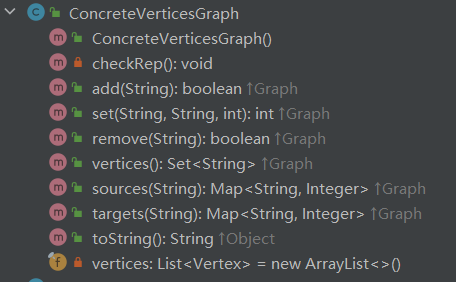


* 1. Vertex的部分代码，分别如下：



2完成ConcreteVerticesGraph类：

2.1确定ConcreteVerticesGraph的field，分别如下:



2.2确定ConcreteVerticesGraph的方法，分别如下：

1、add：简单判断vertices中是否包含vertex点以及vertex的合法性，若包含返回false；若不包含则添加该点入vertices顶点集合。

2、set：首先判断边是否存在，然后对两个端点实施更改，并返回先前边的权值（如果存在，否则返回0）；

3、remove：判断点在vertices中存在后，在vertices中删除该点，并且遍历vertices中其他点，调用set删除与被删点有关的有向边。若修改成功返回true。

4、vertices：返回图中顶点集合的拷贝

5、sources：判断该点是否存在，若存在则返回以该点为target的有向边source顶点集合与边权。

6、targets：判断该点是否存在，若存在则返回以该点为source的有向边target顶点集合与边权。

7、toString：将ConcreteVericesGraph转化为可读的字符串，其中描述了顶点集合信息与各顶点关联有向边信息。

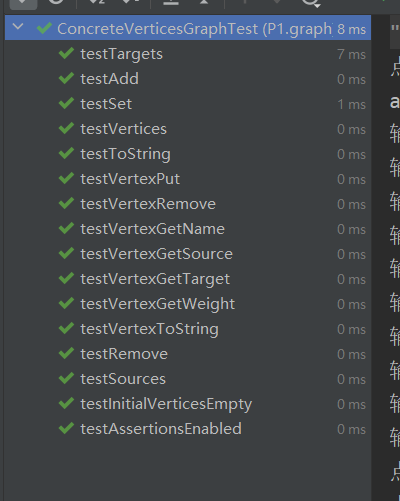
2.3确定ConcreteVerticesGraph的RI,AF,Safety from rep exposure:



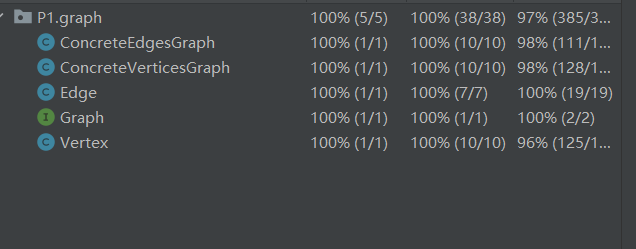
2.4 ConcreteVerticesGraph的部分代码，如下：



3 ConcreteVerticesGraph及Vertex类的测试结果：



覆盖率：



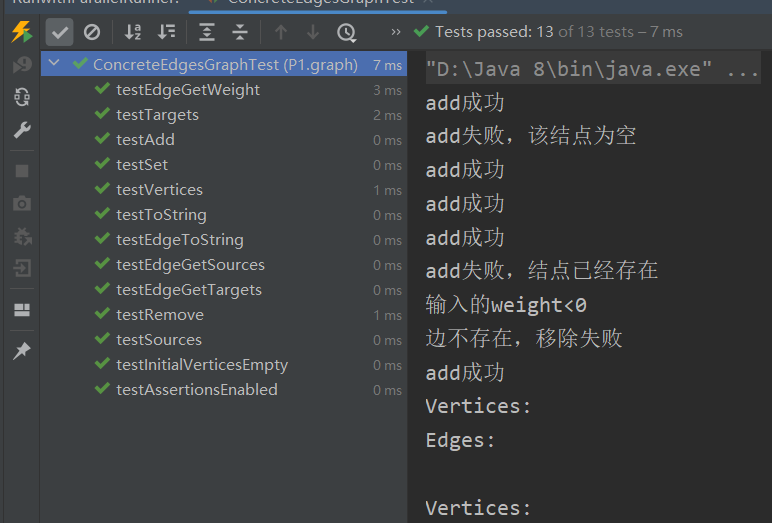
### Problem 3: Implement generic Graph<L>

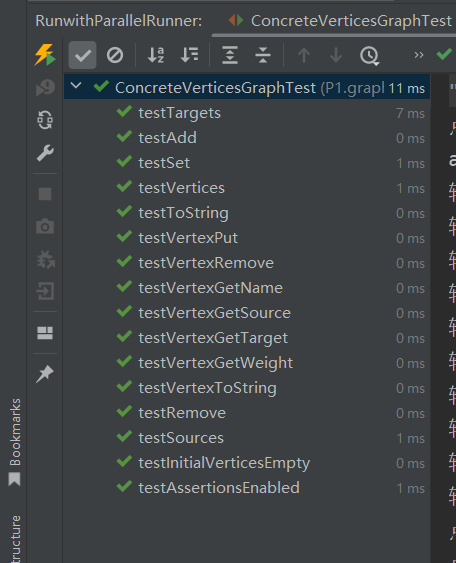
思路：将已有两个Graph<String>的实现类改为基于Graph<L>的实现类，使用IDEA中文本替换工具更改并修复一些错误即可。

#### Make the implementations generic

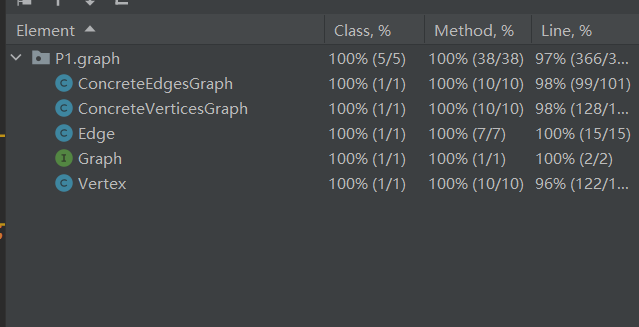
将所有的实现类中的String全部改为泛型实现即可。重新运行ConcreteEdgesGraphTest和ConcreteVerticesGraphTest两个测试，若Graph<String>在两个泛型实现下仍然可以通过，表示更新成功。

测试结果：



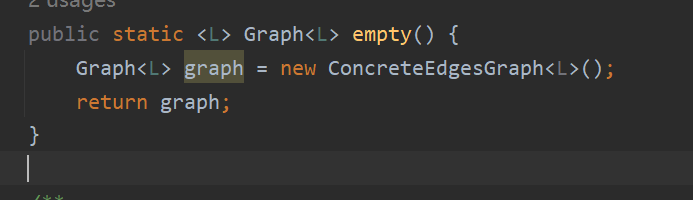


覆盖率：



#### Implement Graph.empty()

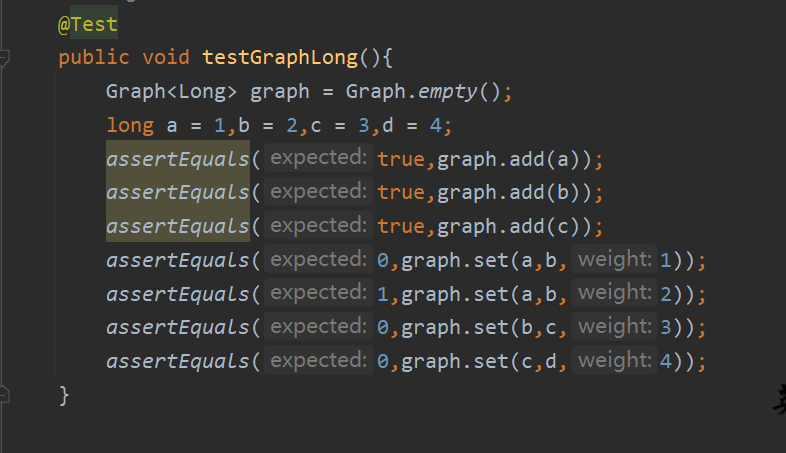
1、empty（）使用ConcreteEdgesGraph<L>类



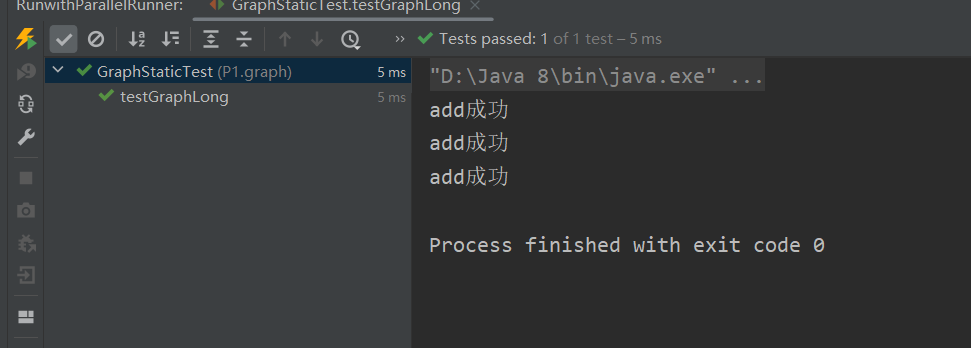
2、使用不同类型的name进行测试

1）使用Long 类测试：

代码：

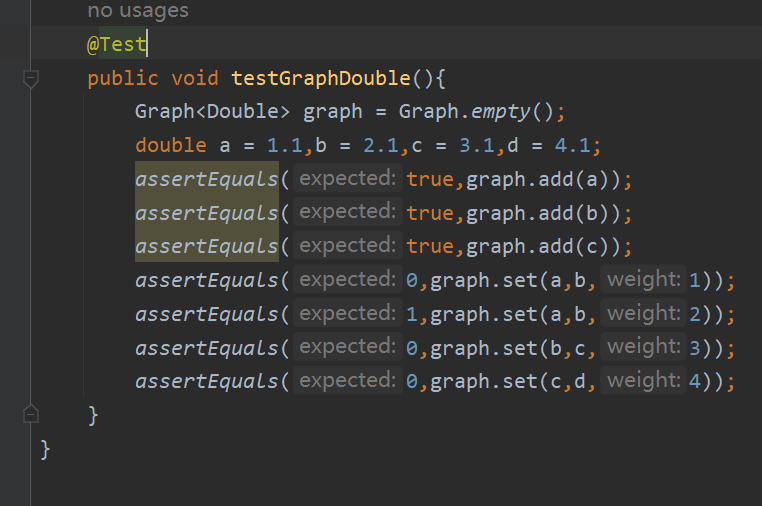


结果：

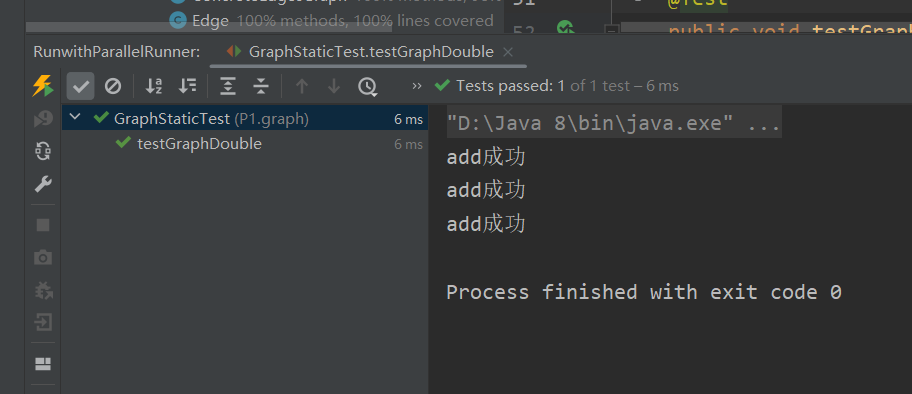


1. 使用Double类测试：

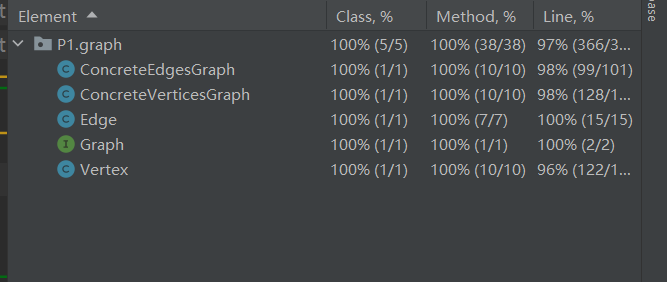
代码：



结果：



3、覆盖率：

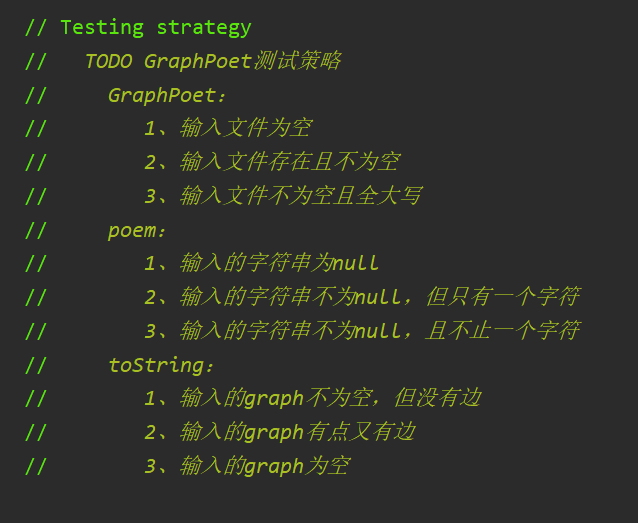


### Problem 4: Poetic walks

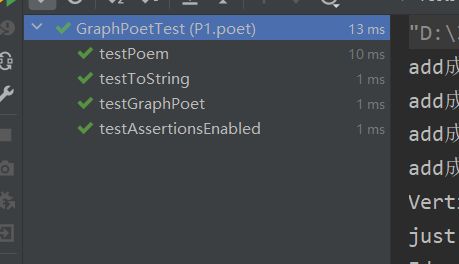
该任务要求我们完成一个GraphPoet类：功能为将输入的词句转化为语料库，然后通过在输入的每个相邻单词对之间插入一个桥接词，得到并输出一首小诗。

#### Test GraphPoet

1. 测试策略：

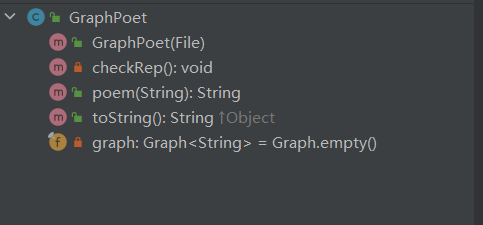


1. 测试结果：

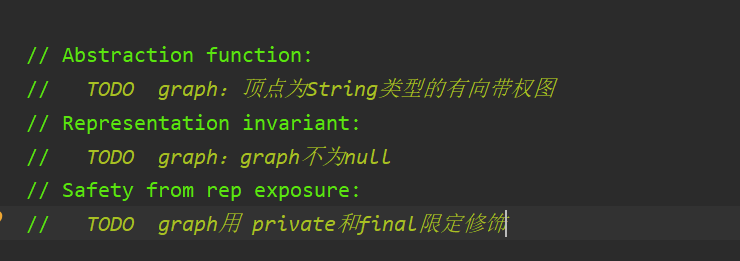


#### Implement GraphPoet

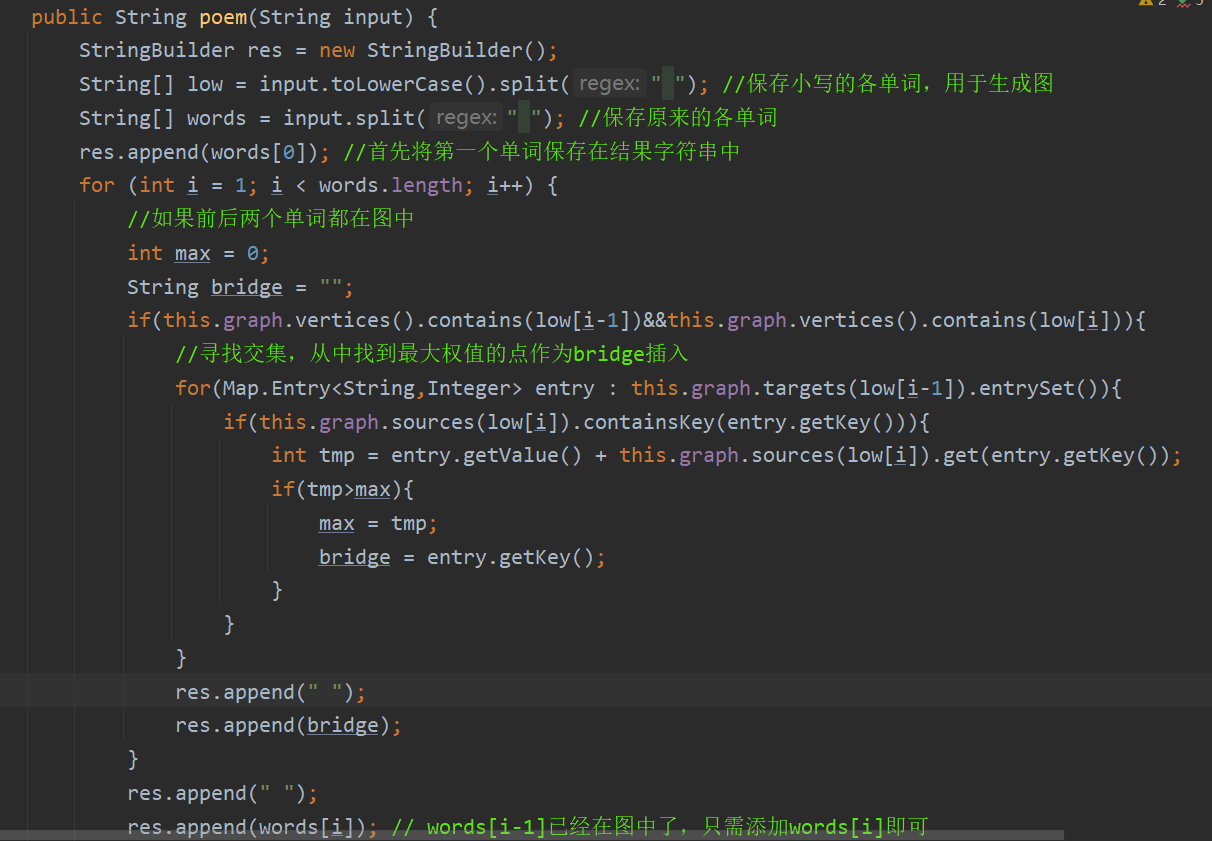
1、GraphPoet的field：



2、GraphPoet的RI,AF,Safety from rep exposure：



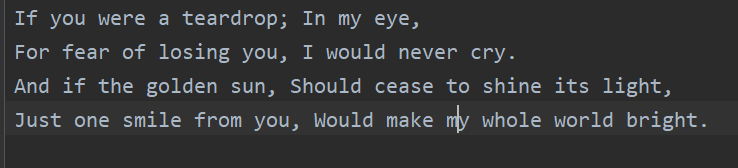
1. GraphPoet的部分代码：



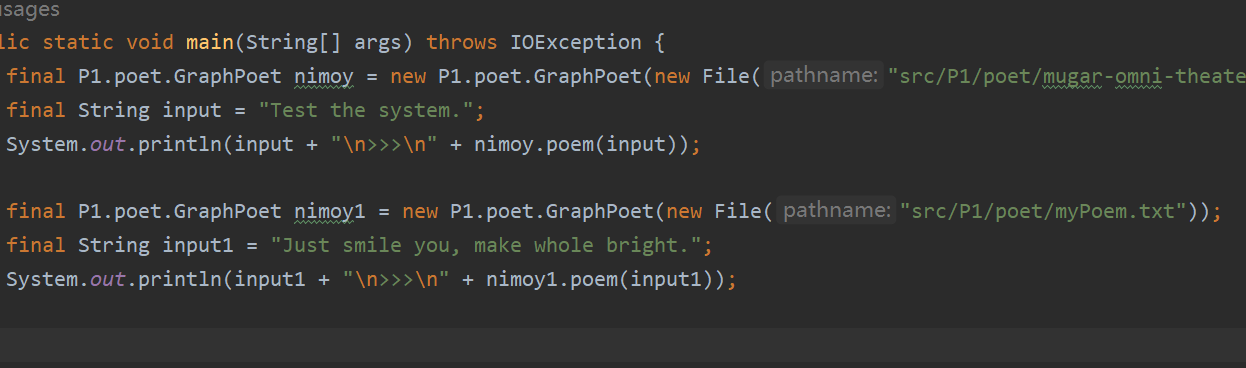
#### Graph poetry slam

功能：使用一个新的语料库和输入新的诗词，进行Main函数的测试。

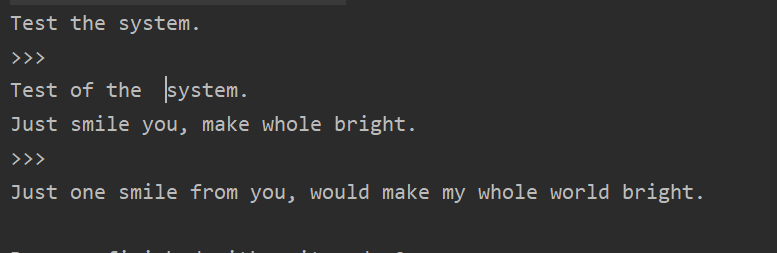
语料库：



部分代码：

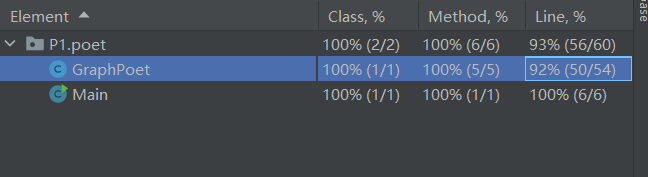


测试结果：



### 使用Eclemma检查测试的代码覆盖度

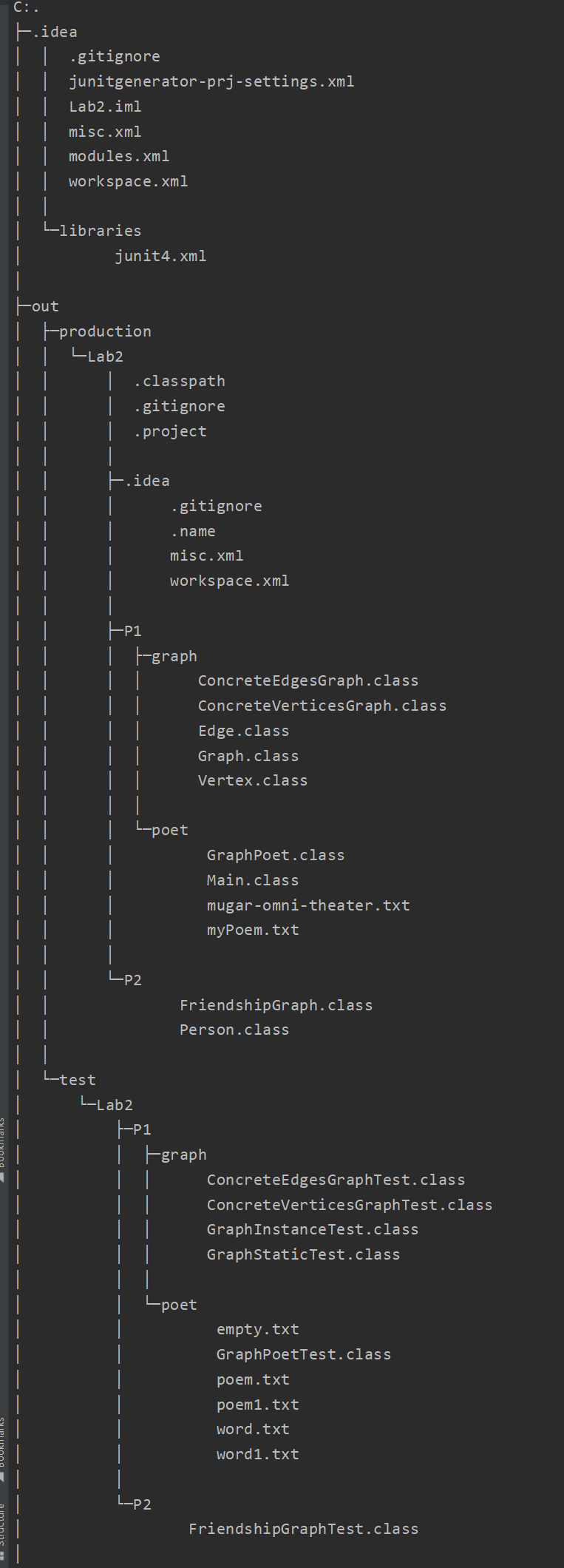
覆盖率：



### Before you’re done

在该文件夹下打开GitBash,使用git add指令，git commit指令添加到本地，最后使用git push<仓库地址> master 将当前的文件上传到远程仓库里去。

项目的目录结构树状示意图。

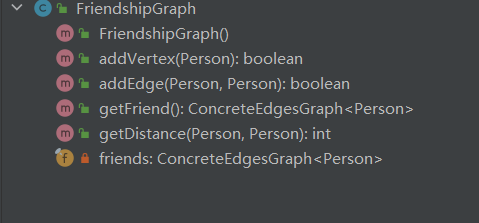


## Re-implement the Social Network in Lab1

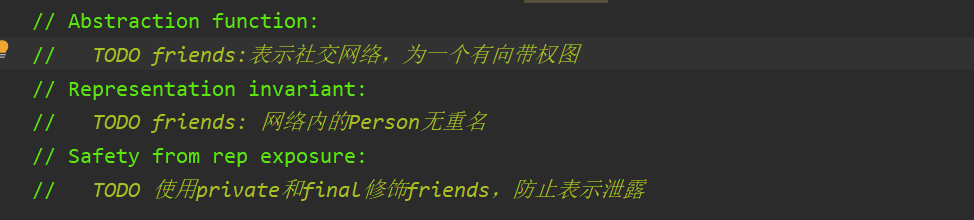
基于Graph<L>的两种实现ConcreteEdgeGraph<L>和ConcreteVerticesGraph<L>，重新实现Lab1中的社交网络，并对其进行测试。

### FriendshipGraph类

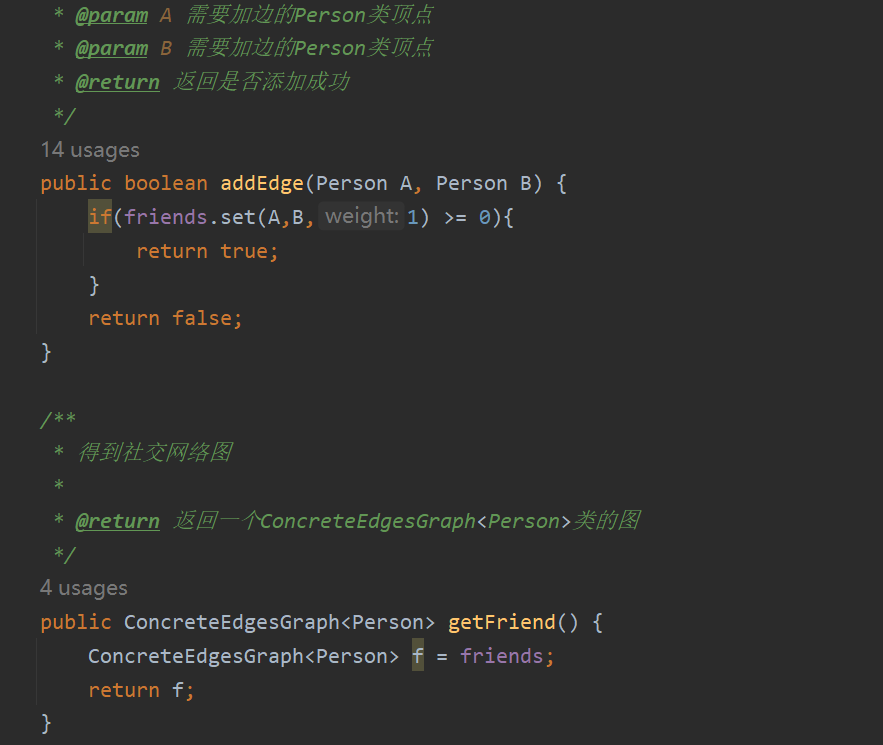
1. FriendshipGraph类的field



1. FriendshipGraph类的RI,AF,Safety from rep exposure



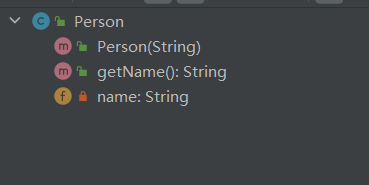
1. FriendshipGraph类的部分代码



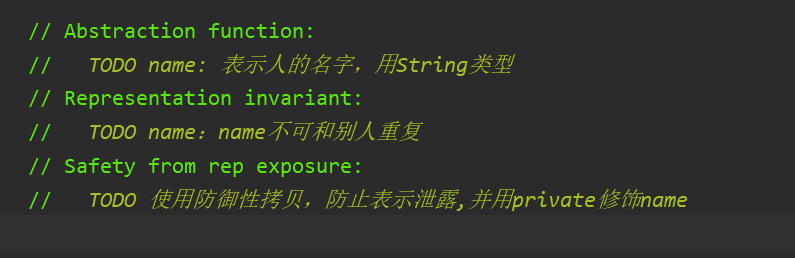
### Person类

思路：Person类私有属性只有Name。其他只要添加类中的构造方法即可。

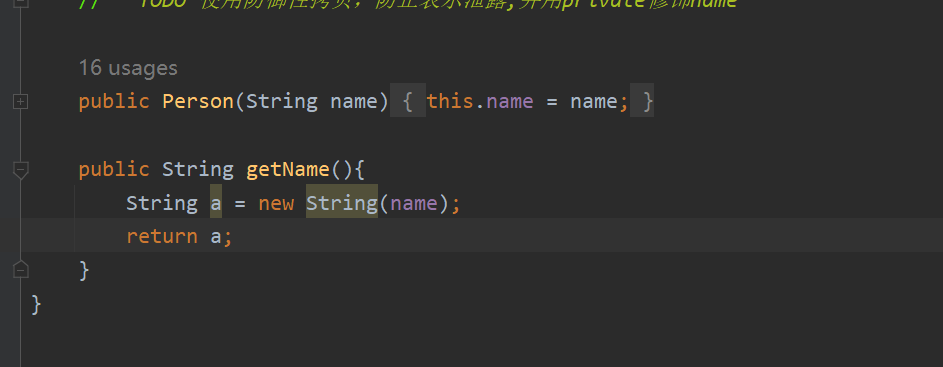
1. Person类的field



1. Person类的RI,AF,Safety from rep exposure



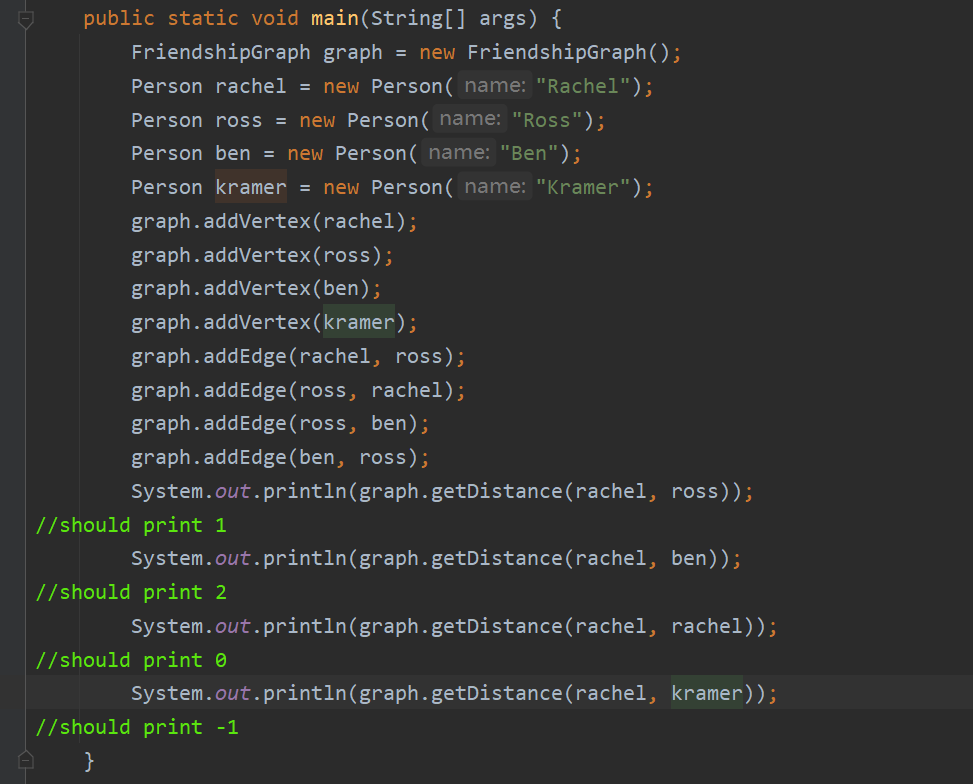
1. Person类的部分代码：



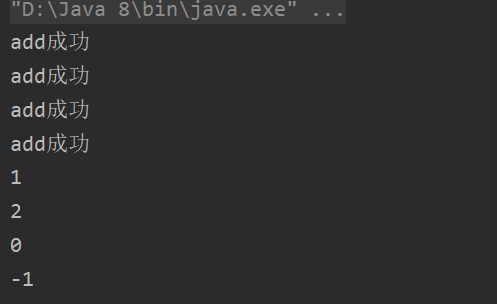
### 客户端main()

思路：添加人名、人与人之间的关系，并调用方法计算距离，并与预期进行对比，看结果是否相同，主函数代码沿用Lab1。

部分代码：

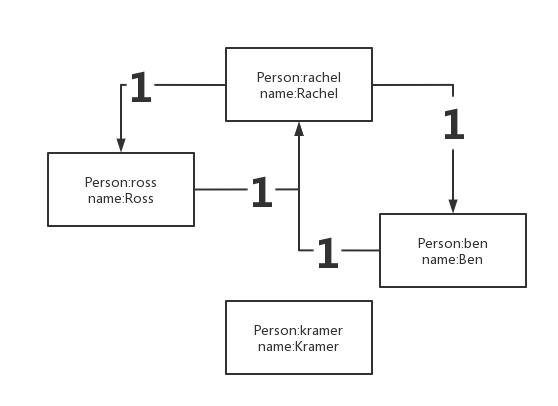


运行结果：



### 测试用例

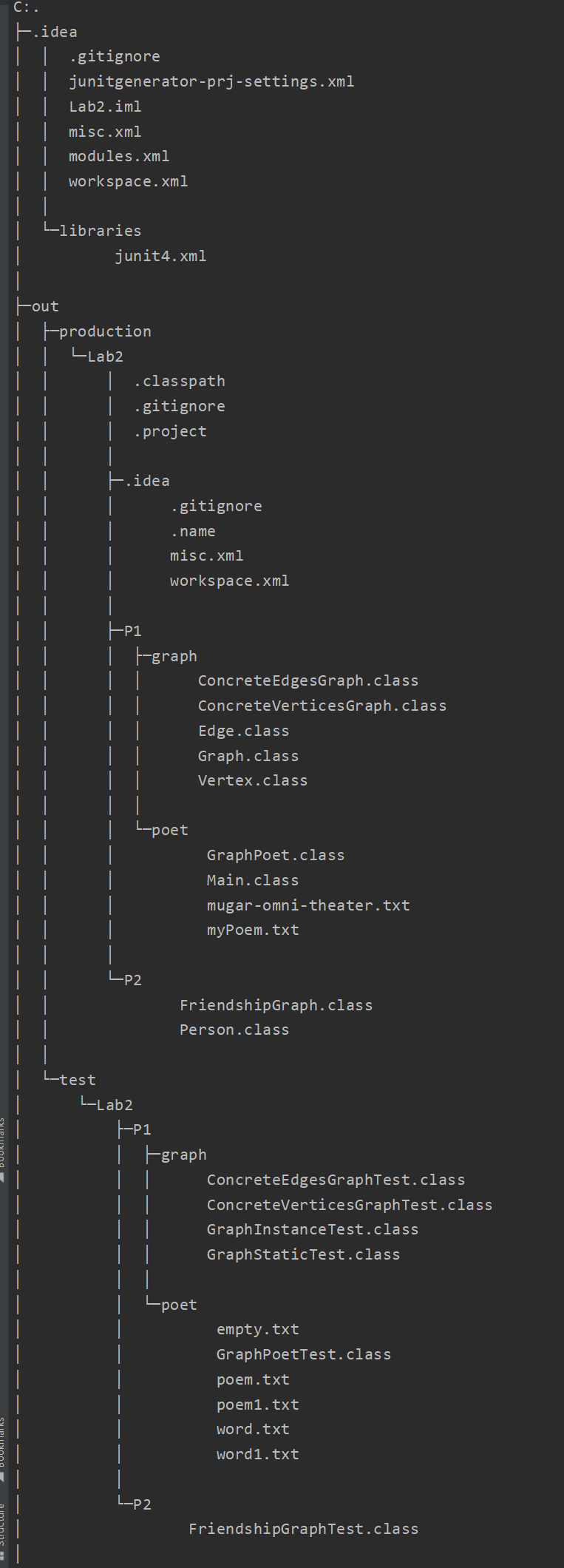
测试用的社交网络图如图所示：



### 提交至Git仓库

在该文件夹下打开GitBash,使用git add指令，git commit指令添加到本地，最后使用git push<仓库地址> master 将当前的文件上传到远程仓库里去。

项目的目录结构树状示意图:



# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2023-03-29 | 18:45-21:20 | 完成P1的第一个任务，并且初步完成Edge类的代码，并填写相关报告 | 按计划完成 |
| 2023-03-30 | 20:00-22:00 | 完成ConcreteEdgesGraph的类功能实现，并完成相关报告 | 按计划完成 |
| 2023-04-01 | 20:00-22:00 | 完成ConcreteEdgesGraph的测试类并测试，完成vertex的相关代码和报告 | 按计划完成 |
| 2023-04-03 | 19:30-22:00 | 完成ConcreteVerticesGraph的类功能实现，并完成相关报告 | 按计划完成 |
| 2023-04-04 | 20:30-22:00 | 完成将ConcreteEdgesGraph和ConcreteVerticesGraph改为泛型实现，并完成相关报告 | 按计划完成 |
| 2023-04-06 | 20:00-22:00 | 完成GraphPoet类的相关代码及报告 | 按计划完成 |
| 2023-04-07 | 20:00-22:00 | 完成P1剩余全部内容，并填写相关报告 | 按计划完成 |
| 2023-04-08 | 18:00-23:00 | 完成P2代码部分及其报告 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 对HashSet类和HashMap类的防御性拷贝不知道如何进行 | 上网查找资料，借鉴思路，看视频，看各种解决方法并总结 |
| 拿到题目P1的Poet开始没有头绪 | 慢慢看提供的spec中的描述，借助翻译软件慢慢理解 |
| Java的一些语法操作不会、不熟悉 | 上网观看教学视频，去csdn等论坛看博客 |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训（必答）

在完成项目之前，对ADT的设计要慎重，要考虑之后设计过程中的方方面面，不然在后续使用ADT的时候会出现许多errors，或者程序运行出现bug，或者运行结果不正确。

## 针对以下方面的感受（必答）

1. 面向ADT的编程和直接面向应用场景编程，你体会到二者有何差异？

面向ADT的编程更加注重复用性，完成的ADT还可以用于其他项目的编写，极大地降低了编程的成本。而直接面向应用场景编程，虽然可能更加简单直接，但是复用性很差，导致完成其他类似项目时依然要付出极大的时间成本。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

使用泛型的编程对我来说更加困难，无法使用具体类型的一些已经写好的方法去完成代码编写，还要考虑不同类型的各种情况，但复用性高。

不使用泛型的编程更加简单，但是复用性比较一般。

1. 在给出ADT的规约后就开始编写测试用例，优势是什么？你是否能够适应这种测试方式？

测试用例可以不依赖于ADT的具体实现对ADT的各项功能进行测试，确保了ADT对spec的遵循，同时将测试与具体实现隔离，使得具体实现以测试为优先，更能在开发过程中准确实现spec中规定的功能。有利于版本的进一步开发。

刚开始是十分不适应的，但是我相信在进一步地编程过后，我能熟练运用这种编程方法。

1. P1设计的ADT在多个应用场景下使用，这种复用带来什么好处？

完成的ADT还可以用于其他项目的编写，极大地降低了编程的时间成本和经济成本等。且更加有利于程序的维护。

1. 为ADT撰写specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后编程中坚持这么做？

意义：防止程序运行出错，保证程序的安全性、可靠性、有效性，极大地提高程序的运行效率，并且方便程序员的后续维护工作。

我愿意在以后的编程中继续这么做。

1. 关于本实验的工作量、难度、deadline。

本实验的工作量较大，难度较大，deadline较紧。

1. 《软件构造》课程进展到目前，你对该课程有何收获和建议？

帮助我进一步了解项目编写过程，让我明白了如何使用ADT编程以提高程序的复用性，对我的编程能力的培养影响巨大，潜移默化地培养我的思维方式和习惯，作用巨大，受益匪浅。