哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Lab Manuals for

Software Construction

Lab-4

Debugging, Exception Handling, and Defensive Programming

School of Computer Science and Technology

Harbin Institute of Technology

Spring 2018

目录

1. 实验目标

本次实验重点训练学生面向健壮性和正确性的编程技能，利用错误和异常处理、断言与防御式编程技术、日志/断点等调试技术、黑盒测试编程技术，使程序可在不同的健壮性/正确性需求下能恰当的处理各种例外与错误情况，在出错后可优雅的退出或继续执行，发现错误之后可有效的定位错误并做出修改。

实验针对 Lab 3 中写好的 ADT（`Graph<L,E>`）代码和基于该 ADT 的四个应用（GraphPoet、社交网络、计算机网络拓扑、电影网络）的代码，使用以下技术进行改造，提高其健壮性和正确性：

- 错误处理
- 异常处理
- Assertion 和防御式编程
- 日志
- 调试技术
- 黑盒测试及代码覆盖度

2. 实验环境

实验环境设置请参见 Lab-0 实验指南。

除此之外，本次实验需要你在 Eclipse IDE 中安装配置 FindBugs（用于 Java 代码静态分析的工具）。请访问 http://findbugs.sourceforge.net，了解它并学习其安装、配置和使用。

本次实验在 GitHub Classroom 中的 URL 地址为：

https://classroom.github.com/a/mnFCZ30Y

请访问该 URL，按照提示建立自己的 Lab4 仓库并关联至自己的学号。

本地开发时，本次实验只需建立一个项目，统一向 GitHub 仓库提交。实验包含的多项任务分别在不同的目录内开发，具体目录组织方式参见各任务最后一部分的说明。请务必遵循目录结构，以便于教师/TA 进行测试。

3. 实验要求

InputFileException

3.1. Error and Exception Handling

针对各种 checked exception，相信你在 Lab 3 的程序中均已经被“强制”考虑过了。本节通过自定义 Exception 类型，表示程序的一些常见错误类型，为代码可能发生的问题提供新的含义，区分代码运行时可能出现的相似问题，或给出程序中一组共性错误的特殊含义。

考虑以下各个场景，使用 exception 和常见的错误机制来处理它们，尽可能 fail fast。为每种错误类型建立新的异常类，在 GraphFactory 及其子类的代码中增加错误和异常处理代码，在特定函数上声明异常（throws）、抛出异常(throw)、捕获并处理异常(try-catch-finally、try-with-resources)。

(1) 输入的文本文件中的内容可能是不合法的或存在数据不一致现象

IllegalGrammerTextException

- 文件中存在不符合语法规则的语句，例如 Vertex = <“TSR”, <“1994”, “USA”, “9.3”>>的最外层<>中缺少了一个分量、Edge = <“SRTR”, “MovieActorRelation”, “2”, “TSR”, “MorganFreeman”, “N”>中最后一个表示是否有向的分量用了 N 而非规定的 No 等：在使用正则表达式进行文本解析的过程中，程序一旦发现这些非法语句，应捕获异常，进行异常处理（提示错误），结束此文件的读取，允许用户选择其他文本文件。

UndefinedVertexException

- 边中使用的节点在节点部分未定义：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

IllegalParamsNumberException

- 为节点定义的属性的数目与特定应用的图的约束不符，例如 Vertex = <“TSR”,“Movie”, <“1994”, “USA”>>只定义了两个属性值 1994 和 USA，缺少第三个属性值：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

在GUI中添加异常

IllegalVertexTypeException

- 在某种类型的图应用中引入了不应出现的节点类型，例如在 GraphPoet 图中出现了类型为 Person 的节点：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

IllegalEdgeTypeException

- 在某种类型的图应用中引入了不应出现的边类型，例如在 GraphPoet 图中出现了类型为 FriendTie 的边：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。
- 在无向图中引入了有向边，例如图 NetworkTopology 中出现了有向边：自动去除这些边的方向，继续向下执行。
- 在有向图中引入了无向边，例如图 SocialNetwork 中出现了有向边：进

UndirectedEdgeDirectedGraphException

入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

- 在单重图中存在了多重边，例如图 NetworkTopology 中出现了多重边：保留两个节点之间的第 1 条边并去除第 2、3、…条边，继续向下执行。
- 在不应存在边的两个节点之间存在了边，例如图 MovieGraph 中出现了 Actor 和 Director 之间的边：直接忽略这种非法边，继续向下执行。
- 在不应存在超边的图中加入了超边，例如图 NetworkTopology 中出现了超边：直接忽略这种非法边，继续向下执行。
- 在不应出现 loop 的图中出现了 loop，例如图 NetworkTopology 中出现了 loop：直接忽略这种非法边，继续向下执行。
- 某超边中包含的节点数小于 2，例如 HyperEdge = <“ActorsInSR”, “SameMovieHyperEdge”, {“TimRobbins”}> 中只有一个节点 TimRobbins：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。
- 带权边却未能给出权值：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。
- 带权边的权值不符合应用要求，例如在 GraphPoet 图中出现权值非正整数的边、在 SocialNetwork 图中出现权值不在(0,1]范围内的边：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。
- 图、节点、边的 label 不满足正则表达式（\w+）的要求：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。
- 多个节点的 label 重复、多个边的 label 重复：应通过自动修改某些 label 来消除重复（例如在重复的 label 后面增加 001、002 等序号），提示用户做出的所有修改，并继续执行。
- 请仔细阅读 Lab3 实验手册中的说明，在解析输入文件为 Graph<L,E>时考虑其他不合法/不一致的情况，并自行设计错误和异常处理机制。

如果红色字表述的处理策略要求你的程序自动为某些错误进行修复（例如忽略非法边、自动修改重复的 label 等），**需在生成图 Graph<L,E>之后将所有做出的自动修复汇总起来提示给用户（请考虑相应的实现策略，例如利用日志或 Memento 设计模式）。**

如果处理策略要求你的程序不自动修复错误，结束当前文件的读取，并提示用户选择其他文本文件，也**需要将所有遇到的不合法之处汇总起来提示给用户**。

(2) 输入的图操作指令可能不合法或存在数据不一致现象（选做，额外计分）

- 与上一节类似，用户输入的图操作指令可能不符合规约（具体规约请参见 Lab 3 实验手册的 3.4 节），例如参数数目不正确、参数值不符合语法要求等。当你的程序解析指令之后发现错误，则进入错误处理或异常处理，本次输入的指令不执行，程序需保持执行，接收用户输入其他指令。
- 指令中包含的参数与程序当前正在处理的图的数据不一致，导致指令无法执行。例如在 `MovieGraph` 图中，若删除某些节点之后，导致某个超边中包含的节点数量降至 1 或 0，则进入异常处理，自动删除超边，程序继续执行。

3.2. Assertion and Defensive Programming

基于你在 Lab 3 里为 ADT `Graph<L,E>`、`Vertex`、`Edge` 和它们的所有子类型所撰写的 rep invariants (RI)、Abstraction Function (AF)，以及它们的每个方法的 spec（pre-condition 和 post-condition），对代码进行防御式改造。

针对每个类，它们有明确的 RI（如果你没有在 Lab 3 的 ADT 代码中阐述清楚这些 invariants，需对照 Lab 3 实验手册仔细补充），请据此撰写 `checkRep()` 并加入各方法。例如（但你的实现不应仅限于这两个例子）：

- 在 `SocialNetwork` 图中，不管用户对图如何修改，所有边的权值之和应始终等于 1；
- 在 `MovieGraph` 图中，任意超边中包含的节点数量一定大于 1。

针对每个方法，它们有明确的 spec，请据此在各方法中通过 assertion 或异常机制分别处理 pre-condition 和 post-condition。例如：

- 在 `Vertex` 的构造函数中，其参数 label 必须满足`\w+`正则表达式的要求；
- 在 `Vertex` 的 `fillConcreteInformation(String[] args)`方法中，`args` 中包含的参数数量应与 `Vertex` 子类型中规定的参数数量一致；

注意：上述两个列表只是在举例，并非全集，你需要对所有的类和方法进行防御式改造，并以注释的形式说清楚你的“防御策略”。

另外：如果你在 Lab3 中已经做了足够多的“防御工作”，该任务只需在实验报告中说清楚你的每一个防御策略和相应的实现即可。

3.3. Logging

使用 `java logging`（或其他第三方 java 日志库如 `log4j`、`SLF4J`），为 3.1

和 3.2 节经过异常处理、错误处理、断言处理的程序增加日志功能。日志需要记录以下信息：

- 所有的异常/错误：发生的时间、异常/错误类型、异常/错误发生的类名和方法名，异常/错误的具体信息、异常/错误处理的结果；
- 对图的所有操作，包括：读取文件、增加节点、增加边、删除节点、删除边、修改节点或边的 label 等。

为应用添加日志查询功能，用户可输入过滤条件（例如按时间段、按类型、按类、按方法、按操作类型）进行日志查询。注意：查询结果不要直接显示原始日志记录，需要具有良好的可理解性。

3.4. Testing for Robustness and Correctness

使用等价类和边界值的测试思想，为 ADT `ConcreteGraph`、`Vertex`、`Edge` 和它们的所有子类型中添加 testing strategy。

考虑 3.1 节中出现的多种非法情形，设计一组测试用例，人为制造非法输入的文件和非法输入的图操作指令，对程序进行健壮性和正确性测试，想方设法让程序崩溃（即验证程序是否有容错能力）。

使用 JUnit 为上述测试用例编写代码，并运行测试。

使用 EclEmma 查看测试的覆盖度，若覆盖度过低，继续增加测试用例，直到覆盖度逼近 100%。生成 EclEmma 的测试覆盖度报告。

注意：对所有的方法进行测试！

另外：如果你在 Lab3 中已经做了足够多的“测试工作”，该任务只需在实验报告中说清楚你的测试策略和相应的实现即可。

3.5. Using FindBugs tool（可选，额外计分）

使用 FindBugs 工具，检查你到目前为止的代码，根据工具所提示的潜在 bug 及其危害程度，逐项消除，并在后续编程中避免此类问题。

3.6. Debugging

该节任务不针对 Lab 3 的背景，而是针对已有的程序，通过测试和调试发现其中存在的 bug 并将其修复。

请从 https://github.com/rainywang/Spring2018_HITCS_SC_Lab4 下载待修复的程序源文件，每个程序中蕴含错误，无法按期望执行得到结果。一共有四个独立的程序：calculator、geometryProcessor、textProcessor、webDownloader。从中选择 1 个（若选多于 1 个，额外计分），利用工具进行 debug，发现问题所在，将代码修改正确，给出测试用例，并提交。

提交目录：

```
项目名称：Lab4_学号
目录：    src
子目录：      debug
                四个程序之一的子目录
                    (修复后的源文件).java
          test
              debug
                四个程序之一的子目录
                        ...Test.java
```

4. 实验报告

针对上述任务，请遵循 CMS 上 Lab4 页面给出的**报告模板**，撰写简明扼要的实验报告。

实验报告的目的是记录你的实验过程，尤其是遇到的困难与解决的途径。不需要长篇累牍，记录关键要点即可，但需确保报告覆盖了本次实验所有开发任务。

注意：

- 实验报告不需要包含所有源代码，请根据上述目的有选择的加入关键源代码，作为辅助说明。
- 请确保报告格式清晰、一致，故请遵循目前模板里设置的字体、字号、行间距、缩进；
- 实验报告提交前，请“目录”上右击，然后选择“更新域”，以确保你的目录标题/页码与正文相对应。

5. 提交方式

截止日期：第 12 周周日（2018 年 5 月 20 日）夜间 23:55。截止时间之后通过 Email 等其他渠道提交实验报告和代码，均无效，教师和 TA 不接收，学生本次实验无资格。

源代码：从本地 Git 仓库推送至个人 GitHub 的 Lab4 仓库内。

实验报告：除了随代码仓库（doc）目录提交至 GitHub 之外，还需手工提交至 CMS 实验 4 页面下。

6. 评分方式

TA 在第 9 周和第 11 周实验课上现场验收：学生做完实验之后，向 TA 提出验收申请，TA 根据实验要求考核学生的程序运行结果并打分。现场验收并非必需，由学生主动向 TA 提出申请。

Deadline 之后，教师使用持续集成工具对学生在 GitHub 上的代码进行测试。教师和 TA 阅读实验报告，做出相应评分。