哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2018 年春季学期

计算机学院大二软件构造课程

Lab 4 实验报告

姓名	穆添愉
学号	1160301008
班号	1603010
电子邮件	1417553133@qq.com
手机号码	15636094072

目录

1 实验目标概述

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

(1)错误处理

(2)异常处理

(3)Assertion 和防御式编程

(4)日志

(5)调试技术

(6)黑盒测试及代码覆盖度

2 实验环境配置

https://github.com/ComputerScienceHIT/Lab4-1160301008

在这里给出你的 GitHub Lab4 仓库的 URL 地址（Lab4-学号）。

3 实验过程

3.1 Error and Exception Handling

针对于所有的异常，有如下的目录

这是我写的一些自定义异常，都是一些需要抛出的异常

捕捉异常处理：

```
while(string != null) {
    try {
    parse(string);
    }
    catch (NoVertexException a) {
        System.out.println(a.getMessage());
        return graph;
    }
    catch (YorNException e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
        return graph;
    }
    catch (IllegalTypeException e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
        return graph;
    }
    catch (NoWeightException e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
        return graph;
    }
    catch (IllegalWeightException e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
        return graph;
    }
    catch (IllegalLabelException e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
```

3.1.1 针对输入文本文件的异常/错误处理

(1) 存在不合法语句，比如说缺少分量，用“N”来代替“No”，有如下的两个异常：

```
package MyException;

public class YorNException extends Exception{
    public YorNException() {
        super();
    }
    @Override
    public String getMessage() {
        return "Please use 'Yes'('No') instead of 'Y'('N'), then change an input file.";
    }
}
```

```
1 package MyException;
2
3 public class LackOfComponentsException extends Exception {
4     public LackOfComponentsException() {
5         super();
6         //System.out.println("The file lack of components, please change an input file.");
7     }
8     @Override
9     public String getMessage() {
0         return "The file lack of components, please change an input file.";
1     }
2 }
3
```

下面是代码中抛出异常的部分：

```
    Matcher matcher6 = Pattern.compile("Edge=<(\\w+),WordNeighborhood,(\\d+),(\\w),(\\w),Y>").matcher(string);
    if(matcher6.matches()) {
        YorNException execption1 = new YorNException();
        throw execption1;
    }
```

可见，当我们输入 Y 而不是 Yes 时，会抛出异常，捕捉到之后，会打印异常信息，从而实现自定义异常处理。

（2）边中使用的节点未定义

```
package MyException;

public class NoVertexException extends Exception{
    public NoVertexException() {
        super();
    }
    @Override
    public String getMessage() {
        return "The edge contains vertex not in the graph, Please change an input file.";
    }
}
```

同样的方法来抛出异常：

```
Matcher matcher5 = Pattern.compile("Edge=<(\\w+),WordNeighborhood,(\\d+),(\\w),(\\w),Yes>").matcher(string);
if(matcher5.matches()) {
    WordEdge edge = new WordEdge(matcher5.group(1), Integer.valueOf(matcher5.group(2)));
    Vertex v1 = null;
    Vertex v2 = null;
    for(Vertex x: graph.vertices()) {
        if(x.getLabel().equals(matcher5.group(3)))
            v1 = x;
        else if(x.getLabel().equals(matcher5.group(4)))
            v2 = x;
    }
    NoVertexException exception = new NoVertexException();
    if(v1 != null && v2 != null) {
        edge.addVertices(Arrays.asList(v1, v2));
        graph.addEdge(edge);
        return;
    } else

            throw exception;

}
```

在加边的时候，去遍历顶点集来寻找边中存在的点，如果两个都能找到，那么进行加边操作，否则就抛出异常。

（3）为节点定义的属性的数目与特定应用的图的约束不符

```
1  package MyException;
2
3  public class LackOfPropertyException extends Exception{
4      public LackOfPropertyException() {
5          super();
6      }
7      @Override
8      public String getMessage() {
9          return "The file lack of properties, please change an input file.";
10     }
11 }
12
```

同样的方法去进行异常的抛出，当输入的字符串缺少属性的时候，就进行异常的抛出。

```
}
Matcher matcher11 = Pattern.compile("Vertex=<(\\w+),(\\w+),<([MF])>>").matcher(stri
if(matcher11.matches()) {
    throw new LackOfPropertyException();
}
```

（4）出现了不应出现的节点类型和边的类型，与上述一样，这个我没有设置正则表达式去进行匹配，因为可能出现太多种不该出现的边和节点类型，所以最后我是如下图来进行异常的抛出：

```
else {
if(string.contains("Person") || string.contains("Computer") || string.contains("Router")
        || string.contains("Server") || string.contains("Movie")
        || string.contains("Director") || string.contains("Actor")
        || string.contains("Connection") || string.contains("Relation"))
    throw new IllegalTypeException();
```

直接看字符串中是不是含有其他种类节点类型（Person，Actor 等）

和边的类型（Connection 等），以此来进行判断，如果含有这些，那么就进行抛出，并打印异常信息。

（5）无向图中出现了有向边，那么就将方向去掉，正常执行加边操作，**这里我觉得这个异常有一点没有必要，因为首先已经不能存在不符合规定的其他类型的边了，也就是说在 GraphPoet 中只能出现 WordNeighbor 这一种边，而 WordNeighbor 已经确定是有向边，那么就不可能同时出现有向边和无向边，所以我觉得这个异常有一点牵强。**

（6）单重图中出现了多重边，只需要在这两个点之间先去进行判断是不是已经存在边了，如果存在的话，忽略掉这条加边的指令，continue。

（7）不该出现边的节点之间连了边，这个由于情况比较少，可以直接根据具体节点来进行判断，例如 Director 和 Actor 之间出现了边。

（8）不该出现 loop 的图中出现了 loop，直接进行判断，如果出现了 loop，跳过就好。

（9）不该出现超边的图中出现了超边，**这个异常我也觉得有一点多余，因为之前已经对于不该在图中出现的边抛出了异常，所以超边也可以进行异常的抛出。**

（10）超边中节点数小于 2，有如下的自定义异常

```
1  package MyException;
2
3  public class HyperEdgeException extends Exception{
4      public HyperEdgeException() {
5          super();
6      }
7      @Override
8      public String getMessage() {
9          return "The HyperEdge is illegal, please change an input.";
10     }
11 }
12
```

一旦超边中的节点数小于 2，就会抛出异常。

```
Matcher matcher7 = Pattern.compile("HyperEdge=<(\\w+),SameMovieHyperEdge,\\{(.*)\\}>").matcher(string);
if(matcher7.matches()) {
    Edge edge = new SameMovieHyperEdge(matcher7.group(1), -1);
    List<Vertex> vertices = new ArrayList<>();
    String allVertices = matcher7.group(2);
    String[] Vertex = allVertices.split(",");
    if(Vertex.length < 2)
        throw new HyperEdgeException();
    for(String x: Vertex) {
        if(!x.matches("(\\w+)"))
            return false;
    }
    for(String x: Vertex) {
        if(x.length() == 0)
            continue;
        boolean flag = false;
        for(Vertex y: graph.vertices()) {
            if(y.getLabel().equals(x)) {
                vertices.add(y);
```

（11）带权边没有给出权重

像上面一样进行正则表达式的匹配，并且进行异常的抛出就好。

（12）权重不符合要求，这里主要选取权重小于 0 的情况，来进行正则匹配，抛出异常。

（13）label 不满足（\w+）的情况，先对 label 进行匹配，如果能匹配（.*）之后，再去匹配（\w+），如果不能匹配，进行异常的抛出，也就意味着含有非法字符。

（14）label 重复，加边加点之前先遍历，不存在重复 label 的话，可以进行加边加点操作，否则忽略掉。

3.1.2 针对输入图操作指令的异常/错误处理（可选）

首先判断指令语法是否正确，如果不正确，直接抛出异常，允许用户输入新的指令。指令语法如下：

```
Pattern p1 = Pattern.compile("vertex --add \"(.+)\" \"(\\w+)\"");
Pattern p2 = Pattern.compile("vertex --delete \"(.+)\"");
Pattern p3 = Pattern.compile(
        "edge --add \"(.+)\" \"(\\w+)\" weighted=(Y|N) (-?[0-9]\\d*(\\.\\d*|0\\.\\d*[1-9]\\d*)?)?\\s?directed=(Y|N) \"(\\w+)\" \"(\\w+)\"");
Pattern p4 = Pattern.compile("edge --delete \"(.+)\"");
Pattern p5 = Pattern.compile("^hyperedge --add \".+\" \"\\w+\" (\"\\w+\" )+\"\\w+\"$");
Pattern p6 = Pattern.compile("vertex --modify oldlabel=(\\w+) newlabel=(.+) (args=(\"\\w+\"\\s?)*)");
```

1. Vertex –add label type。

   如果该语法匹配成功，则提取出 label、type 信息。首先判断 label 是否符合\w+或者输入的 label 已经在图中出现过，否则抛出异常，允许用户输入新的指令。根据不同类型的图，判断 type 对于该类图是否合法。

```
vertex --add "illegalLabel+++" "Word"
Exception.IllegalCommandException: vertex --add "illegalLabel+++" "Word" : new vertex label is illegal
        at helper.ParseCommandHelper.parseAndExecuteCommand(ParseCommandHelper.java:69)
        at application.GraphPoetApp.main(GraphPoetApp.java:54)

vertex --add "label" "Person"
Exception.IllegalCommandException: vertex --add "label" "Person" : the new vertex's type is illegal for the graph
        at helper.ParseCommandHelper.addVertex(ParseCommandHelper.java:474)
        at helper.ParseCommandHelper.parseAndExecuteCommand(ParseCommandHelper.java:81)
        at application.GraphPoetApp.main(GraphPoetApp.java:54)

vertex --add "Kevin" "Person"
Exception.IllegalCommandException: vertex --add "Kevin" "Person" : the vertex label has existed in the graph
        at helper.ParseCommandHelper.parseAndExecuteCommand(ParseCommandHelper.java:77)
        at application.SocialNetworkApp.main(SocialNetworkApp.java:51)
```

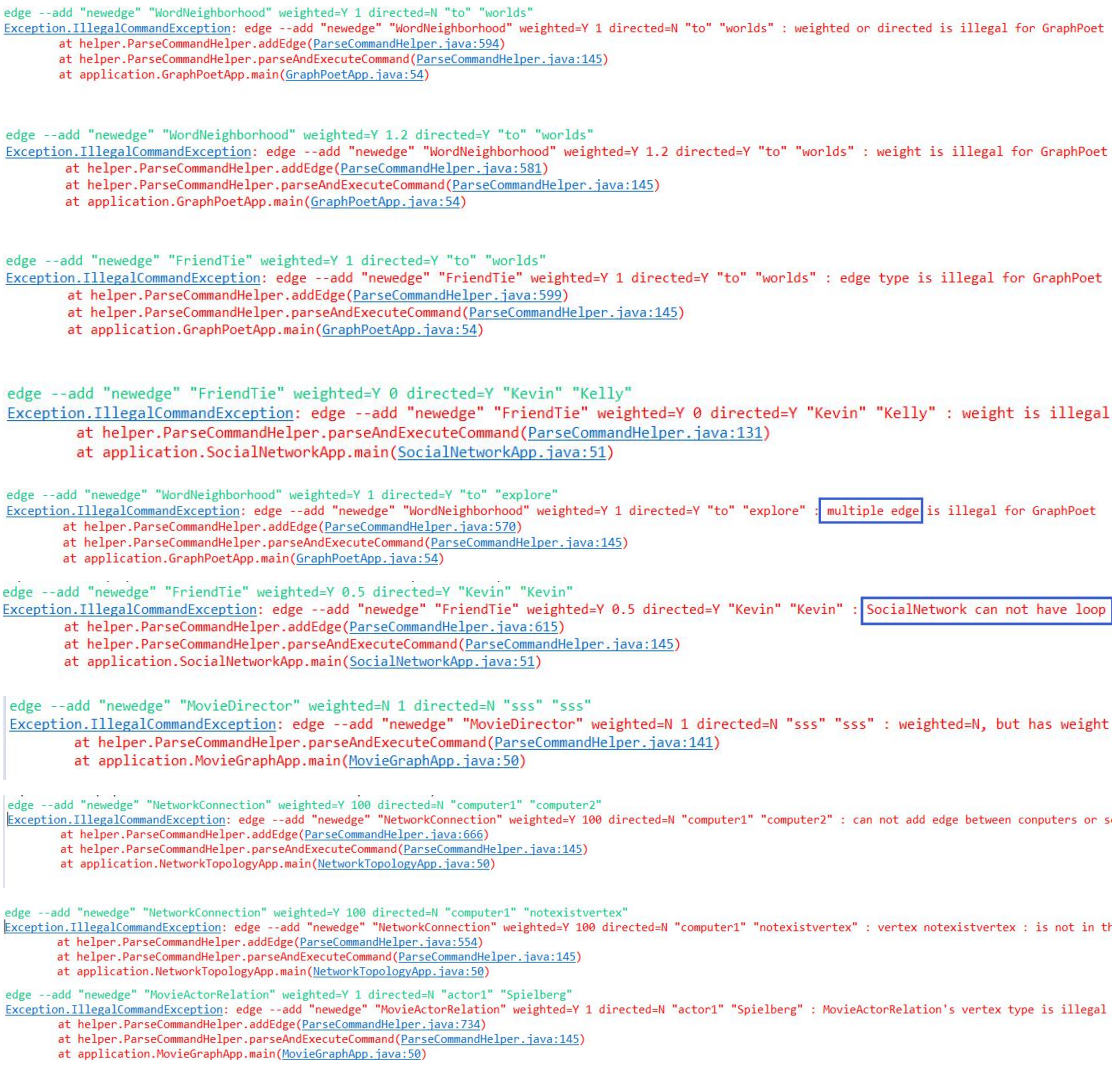
2. Vertex –delete regex

   如果匹配成功该指令，则不需要抛出异常，只需要找到 label 符合该 regex 的点删除即可。

3. Edge –add label type weighted=Y|N weight directed=Y vertex1 vertex2

   如果该语法匹配成功，提取出 label，type，这两个属性的判断方法与 vertex 的一致。首先判断两个顶点是否都在图中。根据不同类型图或者不同类型边来判断 weighted 与 directed 是否合法。对于 weighted=Y 的边，需要判断所输入的 weight 对于该类型的边来说是否是合法的。如果是无权边，可以允许不输入权值，默认为-1，但是带权边必须手动输入权值。还需要判断是否在单重图中出现了多重边。对于不允许出现自环的图，如要判断新加的边是否为自环。对于无向边，

判断是否给出了非零的权值。对于某些边，要判断顶点类型是否合法。如 NetworkConnection 要求两个顶点不能同时为 Computer 或 Server.如果上述情况出现不合法的情况，均抛出异常，允许用户输入新的指令。



4. Edge –delete regex

与 vertex –delete 相同，不需要抛出异常。

5. HyperEdge – add label type vertex1 vertex2……

如果匹配该语句，首先判断 label 是否合法。然后判断该图中是否含有超边，对于 GraphPoet、SocialNetwork、

NetworkTopology 不允许超边存在。如果 MovieGraph，判断超边的类型是否正确。然后判断加入顶点的个数，是否大于等于 2。如果上述合法，判断所加入的顶点是否都在图中，否则抛出异常。判断所加入的点的类型是否合法。最终判断所加入的一条超边上的顶点，是否出演同一个电影。如果图中已经有该 label 的超边，只需要将顶点加入到原有的超边上。

6. Vertex –modify oldlabel=xxxx newlabel=xxxx args=

首先判断原 label 是否在图中，然后判断所给的 newlabel 是否符合\w+。两个 label 可以相同。根据不同类型的图来判断所给的属性是否合法。

7. Edge –modify oldlabel label=Y|N newlabel weight=Y|N newweight direct=Y|N

首先判断 oldlabel 是否在图中，否则抛出异常如果在 label=Y 或者 weight=Y 后面没有 label 或 weight，抛出异常

```
edge --modify sss label=Y newlabel weight=N direct=N
Exception.IllegalCommandException: edge --modify sss label=Y newlabel weight=N direct=N : edge not in graph
        at helper.ParseCommandHelper.parseAndExecuteCommand(ParseCommandHelper.java:248)
        at application.SocialNetworkApp.main(SocialNetworkApp.java:51)
```

```
edge --modify friend1 label=Y weight=N direct=N
Exception.IllegalCommandException: edge --modify friend1 label=Y weight=N direct=N : command error, try "help"
        at helper.ParseCommandHelper.parseAndExecuteCommand(ParseCommandHelper.java:278)
        at application.SocialNetworkApp.main(SocialNetworkApp.java:51)
```

3.2 Assertion and Defensive Programming

3.2.1 checkRep()检查 invariants

（1）在 concreteGraph.java 中有如下的 checkRep（）：

```
public void checkRep() {
    assert allVertices.size() >= allEdges.size() * 2;
    assert allEdges != null;
    assert allVertices != null;
}

/**
```

针对于超边，没有额外的去写 checkRep（），因为我在进行加超边的时候已经进行了判断，一定要节点数大于 2，

```
while (it.hasNext()) {
    Edge edge = (Edge) it.next();
    if(edge.vertices().contains(v)) {
        if(edge instanceof HyperEdge) {
            if(edge.vertices().size() >= 2) {
                ((HyperEdge) edge).removeVertex(v);
                continue;
            }
                                    // hyper edge is already legal.
            else it.remove();
        }
        else it.remove();
    }
}
```

另外，针对于 SocialNetwork，需要满足所有边权重之和为 1，所以要重写 checkRep（）如下：

```java
@Override
public void checkRep() {
    super.checkRep();
    double sum = 0;
    for(Edge x: allEdges) {
        sum += x.getWeight();
    }
    assertEquals(1, sum, 0.01);
}
```

（3）对于 movie 来说，要保证 IMDB 得分在 0~10 分之间，这个已经在正则表达式的地方进行限制，也就是限制一位小数，还有拍摄年份限制为四位整数

```
Matcher matcher4 = Pattern.compile("Vertex=<(\\w+),Movie,<(\\d{4}),(\\w+),([\\d\\.]+)>>").matcher(string);
if(matcher4.matches()) {
    Movie movie = new Movie(matcher4.group(1));
    movie.fillVertexInfo(new String[]{matcher4.group(2), matcher4.group(3), matcher4.group(4)});
    graph.addVertex(movie);
    return true;
}
```

（4）对于网络拓扑图来说，要严格限制 IP 地址的格式，这个也已经在正则表达式中进行了限制：

```
Matcher matcher3 = Pattern.compile("Vertex=<(\\w+),(\\w+),<(\\d+\\.\\d+\\.\\d+\\.\\d+)>>").matcher(string);
if(matcher3.matches()) {
    if(matcher3.group(2).equals("Server")) {
        Server server = new Server(matcher3.group(1));
        server.fillVertexInfo(new String[] {matcher3.group(3)});
        graph.addVertex(server);
        return;
    }
    else if(matcher3.group(2).equals("Computer")) {
        vertex.Computer computer = new vertex.Computer(matcher3.group(1));
        computer.fillVertexInfo(new String[] {matcher3.group(3)});
        graph.addVertex(computer);
        return;
    }
    else if(matcher3.group(2).equals("Router")) {
        Router router = new Router(matcher3.group(1));
        router.fillVertexInfo(new String[] {matcher3.group(3)});
        graph.addVertex(router);
        return;
    }
    else if(matcher3.group(2).equals("WirelessRouter")) {
        vertex.Computer computer = new vertex.Computer(matcher3.group(1));
        computer.fillVertexInfo(new String[] {matcher3.group(3)});
        graph.addVertex(computer);
        return;
    }
```

3.2.2 Assertion 保障 pre-/post-condition

（1）针对于 label，要保证所有的 label 都是（\w+），所以有如下

的 assert：

```
/**
 * Constructor of Vertex.
 * @param label, String object, which can not be null.
 */
public Vertex(String label) {
    Matcher matcher = Pattern.compile("(\\w+)").matcher(label);
    assertTrue(matcher.matches());
        this.label = label;
}
```

（2）要保障给节点传入信息的时候，数组的长度和要求属性的数量一致，这个就要每个类分别考虑了

```
/**
 * Filling the info of the vertex.
 */
@Override
public void fillVertexInfo(String[] args) {
    assertTrue(args.length == 2);
    gender = args[0];
    age = Integer.valueOf(args[1]).intValue();

}
```

3.3 Logging

3.3.1 写日志

这里采用老师推荐的第三方库 log4j，下载安装之后，有如下的配置文件：

Main.java Main.java SocialNetworkAp GraphPoetApp.ja MovieGraphApp.j NetworkTopology log4j2.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<Configuration status="info">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n" />
        </Console>
        <File name="log" fileName="log/graph.log" append="true">
            <PatternLayout pattern="%d{yyyy.MM.dd.HH:mm:ss} %p %C %M %m%n"/>
        </File>
    </Appenders>
    <Loggers>
        <Root level="INFO">
            <AppenderRef ref="Console" />
            <AppenderRef ref="log" />
        </Root>
    </Loggers>
</Configuration>
```

这里规定了日志的输出格式，以及输出的位置：src/graph/log.txt

在每个异常的位置和加边加点的位置我加入了记录日志，最终，日志效果如下：

```
2018.05.20.20:48:59 INFO factory.GraphPoetFactory parse Addedge: ac, which contains vertices: a, c, and the weight is 1
2018.05.20.20:48:59 INFO factory.GraphPoetFactory parse Addedge: bd, which contains vertices: b, d, and the weight is 1
2018.05.20.20:48:59 INFO factory.GraphPoetFactory parse Addedge: cd, which contains vertices: c, d, and the weight is 1
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse GraphType=MovieGraph
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse GraphName is MyFavoriteMovies
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse VertexType are Movie, Actor, Director.
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addmovie: TheShawshankRedemption
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addmovie: TheShawshankRedemption2
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Adddirector: FrankDarabont
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addactor: MorganFreeman
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addactor: TimRobbins
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addmovie: TheGreenMile
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addactor: TomHanks
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse EdgeType are MovieActorRelation, MovieDirectorRelation, SameMovieHyperEdge.
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addedge, MovieActorRelation, SRFD
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addedge, MovieActorRelation, GMFD
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addedge, MovieActorRelation, SRMF
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addedge, MovieActorRelation, SRTR
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse Addedge, MovieActorRelation, GMTH
2018.05.20.20:49:20 INFO factory.MovieGraphFactory parse AddHyperEdge, ActorsInSR
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse GraphType=NetworkTopology
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse GraphName is LabNetwork
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse VertexType are Computer, Router, Server, Server.
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse Addvertex, Server: Server1
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse Addvertex, Router: Router1
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse Addvertex, Computer: Computer1
2018.05.20.20:49:48 INFO factory.NetworkTopologyFactory parse EdgeType is NetworkConnection
2018.05.20.20:49:48 ERROR factory.NetworkTopologyFactory createGraph NoVertexException, The edge contains vertex not in the graph, Please change an input file.
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse GraphType=SocialNetwork
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse GraphName is MySocialNetwork
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse VertexType is Person
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addvertex, Person: Wang
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addvertex, Person: Li
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addvertex, Person: Liu
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addvertex, Person: Zhao
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse EdgeType are ForwardConnection, CommentConnection, FriendConnection.
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addedge, ForwardConnection: LiForwardLiu
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addedge, ForwardConnection: ZhaoForwardLiu
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addedge, CommentConnection: WangCommentZhao
2018.05.20.20:50:08 INFO factory.SocialNetworkFactory parse Addedge, FriendConnection: LiFriendLiu
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse GraphType=GraphPoet
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse VertexType is Word
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addvertex: a
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addvertex: b
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addvertex: c
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addvertex: d
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse EdgeType is WordNeighborhood
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addedge: ab, which contains vertices: a, b, and the weight is 1
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addedge: ac, which contains vertices: a, c, and the weight is 1
2018.05.20.20:50:31 INFO factory.GraphPoetFactory parse Addedge: bd, which contains vertices: b, d, and the weight is 1
```

3.3.2 日志查询

比较时间的部分比较麻烦，要考虑很多种情况，具体如下：

```
public static boolean compareTime(String s1, String s2) {
    String[] a1 = s1.split("\\."), a2 = s2.split("\\.");
    if(Integer.valueOf(a1[0]) > Integer.valueOf(a2[0])) return false;
    else if (Integer.valueOf(a1[0]) < Integer.valueOf(a2[0])) return true;
    if(Integer.valueOf(a1[1]) > Integer.valueOf(a2[1])) return false;
    else if(Integer.valueOf(a1[1]) < Integer.valueOf(a2[1])) return true;
    if(Integer.valueOf(a1[2]) > Integer.valueOf(a2[2])) return false;
    else if(Integer.valueOf(a1[2]) < Integer.valueOf(a2[2])) return true;
    a1 = a1[3].split(":"); a2 = a2[3].split(":");
    if(Integer.valueOf(a1[0]) > Integer.valueOf(a2[0])) return false;
    else if (Integer.valueOf(a1[0]) < Integer.valueOf(a2[0])) return true;
    if(Integer.valueOf(a1[1]) > Integer.valueOf(a2[1])) return false;
    else if(Integer.valueOf(a1[1]) < Integer.valueOf(a2[1])) return true;
    System.err.println(a1[2] + "\t" + a2[2]);
    if(Integer.valueOf(a1[2]) > Integer.valueOf(a2[2])) return false;
    else if(Integer.valueOf(a1[2]) < Integer.valueOf(a2[2])) return true;
    return false;
}
public static void printLog(String str) {
    String[] a = str.split(" ");
    System.out.print("Time: " + a[0] + "\t" +
            "Class: "  + a[2] + "\t" +
            "Method: " + a[3] + "\t" +
            "Operation: " + a[4]);
    str = "";
    for(int i=5; i<a.length; i++) {
        str += a[i] + " ";
```

最终输出效果如下：

（1）按照时间段进行查询

```
Please choose a function(1 - 9) and input the number.
9
1 By time
2 By class
3 By method
4 By operation
Please input the number
1
Please input the time, from(such as 2018.05.19.18:41:07):
2018.05.20.20:49:20
Please input the time, to(such as 2018.05.19.18:43:07):
2018.05.20.20:50:08
Here are logs founded:
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: GraphType=NetworkTopo
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: GraphNameis LabNetworl
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: VertexTypeare Computer
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: Addvertex,Server: Serv
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: Addvertex,Router: Rout
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: Addvertex,Computer: C
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: parse   Operation: EdgeTypeis NetworkConn
20      48
Time: 2018.05.20.20:49:48      Class: factory.NetworkTopologyFactory   Method: createGraph     Operation: NoVertexExcep
08      08
```

（2）按照类名进行查询

```
Please choose a function(1 - 9) and input the number.
9
1 By time
2 By class
3 By method
4 By operation
Please input the number
2
Please input the class name
factory.SocialNetworkFactory
Here are logs founded:
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: GraphType=SocialNetwo
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: GraphNameis MySocialNe
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: VertexTypeis Person
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex, Person: Wan
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex, Person: Li
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex, Person: Liu
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex, Person: Zha
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: EdgeTypeare ForwardCon
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addedge, ForwardConnec
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addedge, ForwardConnec
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addedge, CommentConnec
Time: 2018.05.20.20:03:20       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addedge, FriendConnect
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: GraphType=SocialNetwo
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: GraphNameis MySocialNe
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: VertexTypeis Person
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex,Person: Wang
```

（3）按照方法名进行查询

```
Please choose a function(1 - 9) and input the number.
9
1 By time
2 By class
3 By method
4 By operation
Please input the number
3
Please input the method name
parse
Here are logs founded:
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: GraphType=GraphPoet
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: VertexTypeis Word
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: a
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: b
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: c
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: d
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: EdgeTypeis WordNeighborhood
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addedge: ab, which contains
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addedge: ac, which contains
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addedge: bd, which contains
Time: 2018.05.20.20:01:30       Class: factory.GraphPoetFactory Method: parse   Operation: Addedge: cd, which contains
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: GraphType=GraphPoet
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: GraphNameis MyGraphPoet
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: VertexTypeis Word
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: a
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: b
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: c
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: Addvertex: d
Time: 2018.05.20.20:01:39       Class: factory.GraphPoetFactory Method: parse   Operation: EdgeTypeis WordNeighborhood
Time: 2018.05.20.20:01:50       Class: factory.GraphPoetFactory Method: parse   Operation: GraphType=GraphPoet
```

（4）按照操作进行查询

```
Please choose a function(1 - 9) and input the number.
9
1 By time
2 By class
3 By method
4 By operation
Please input the number
4
Please input the operation name
Addvertex,
Here are logs founded:
Time: 2018.05.20.20:49:48       Class: factory.NetworkTopologyFactory   Method: parse   Operation: Addvertex,Server: Serv
Time: 2018.05.20.20:49:48       Class: factory.NetworkTopologyFactory   Method: parse   Operation: Addvertex,Router: Rout
Time: 2018.05.20.20:49:48       Class: factory.NetworkTopologyFactory   Method: parse   Operation: Addvertex,Computer: Co
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex,Person: Wang
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex,Person: Li
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex,Person: Liu
Time: 2018.05.20.20:50:08       Class: factory.SocialNetworkFactory     Method: parse   Operation: Addvertex,Person: Zhao
```

至此，四种查询方式已经全部实现。

3.4 Testing for Robustness and Correctness

3.4.1 Testing strategy

```
//  -Test strategy
//   the method in the factory which can throw my exceptions is parse(),
//   so I just call it to parse and execute the string which can appear
//   in the input files, and then input the false string to make sure it
//   can throw the expected exceptions.
```

总之就是模仿各种异常输入，然后去判断是不是进行了期望的异常的抛出。

3.4.2 测试用例设计

（1）Exception 测试

首先先要明确一点是 factory 中的一个方法 parse（）来进行字符串的解析，从而进行加边加点的操作，所以，只需要让 parse（）方法去解析不同的字符串去模仿加边加点，然后判断抛出的异常类型即可。示例如下：

```
/**
 * Test for the illegal input for "Y"("N") instead of "Yes"("No")
 * @throws Exception
 */
@Test(expected = YorNException.class)
public void testYorNException() throws Exception {
    //Graph<Vertex, Edge> g = new GraphPoetFactory().createGraph("src/GraphPoet.txt");
    GraphPoetFactory graphPoetFactory = new GraphPoetFactory();
    graphPoetFactory.createGraph("src/GraphPoet.txt");
    new GraphPoetFactory().parse("Edge=<ab,WordNeighborhood,1,a,b,Y>");
}
```

```
/**
 * Test for the exception that the edge contains vertex which is not in the graph yet.
 * @throws Exception
 */
@Test(expected = NoVertexException.class)
public void testNoVertexException() throws Exception {
    GraphPoetFactory graph = new GraphPoetFactory();
    graph.parse("GraphName=PoetGraph");
    graph.parse("EdgeType=WordNeighborhood");
    graph.parse("GraphType=GraphPoet");
    graph.parse("VertexType=Word");
    graph.parse("Vertex=<a,Word>");
    graph.parse("Vertex=<b,Word>");
    graph.parse("Edge=<ab,WordNeighborhood,1,a,b,Yes>");
    graph.parse("Edge=<ac,WordNeighborhood,1,a,c,Yes>");
}
```

以下是针对于 GraphPoet 来进行所有异常的测试

```
/**
 * Test for the robustness
 */
@Test
public void testExceptions() {
    GraphPoetFactory graphPoetFactory1 = new GraphPoetFactory();
    graphPoetFactory1.createGraph("src/GraphPoet1.txt");
    GraphPoetFactory graphPoetFactory2 = new GraphPoetFactory();
    graphPoetFactory2.createGraph("src/GraphPoet2.txt");
    GraphPoetFactory graphPoetFactory3 = new GraphPoetFactory();
    graphPoetFactory3.createGraph("src/GraphPoet3.txt");
    GraphPoetFactory graphPoetFactory4 = new GraphPoetFactory();
    graphPoetFactory4.createGraph("src/GraphPoet4.txt");
    GraphPoetFactory graphPoetFactory5 = new GraphPoetFactory();
    graphPoetFactory5.createGraph("src/GraphPoet5.txt");
    GraphPoetFactory graphPoetFactory6 = new GraphPoetFactory();
    graphPoetFactory6.createGraph("src/GraphPoet6.txt");
    GraphPoetFactory graphPoetFactory7 = new GraphPoetFactory();
    graphPoetFactory7.createGraph("src/GraphPoet7.txt");
    GraphPoetFactory graphPoetFactory8 = new GraphPoetFactory();
    graphPoetFactory8.createGraph("src/GraphPoet8.txt");
}
```

其中这些测试文件就是模仿各种不同的错误输入。

示例如下：

这是用“Y”来代替 “Yes”

```
GraphType=GraphPoet
GraphName=MyGraphPoet
VertexType=Word
Vertex=<a,Word>
Vertex=<b,Word>
Vertex=<c,Word>
Vertex=<d,Word>
EdgeType=WordNeighborhood
Edge=<ab,WordNeighborhood,1,a,b,Y>
Edge=<ac,WordNeighborhood,1,a,c,Yes>
Edge=<bd,WordNeighborhood,1,b,d,Yes>
Edge=<cd,WordNeighborhood,1,c,d,Yes>

```

这是带权边没有输入权值：

输入边的 label 不符合（\w+）

```
GraphType=GraphPoet
VertexType=Word
Vertex=<a,Word>
Vertex=<b,Word>
Vertex=<c,Word>
Vertex=<!d,Word>
EdgeType=WordNeighborhood
Edge=<ab,WordNeighborhood,1,a,b,Yes>
Edge=<ac,WordNeighborhood,1,a,c,Yes>
Edge=<bd,WordNeighborhood,1,b,d,Yes>
Edge=<cd,WordNeighborhood,1,c,d,Yes>

```

边中的点不在图中：

```
GraphType=GraphPoet
VertexType=Word
Vertex=<a,Word>
Vertex=<b,Word>
Vertex=<c,Word>
Vertex=<d,Word>
EdgeType=WordNeighborhood
Edge=<ab,WordNeighborhood,1,a,e,Yes>
Edge=<ac,WordNeighborhood,1,a,c,Yes>
Edge=<bd,WordNeighborhood,1,b,d,Yes>
Edge=<cd,WordNeighborhood,1,c,d,Yes>

```

输入的权值不合法：

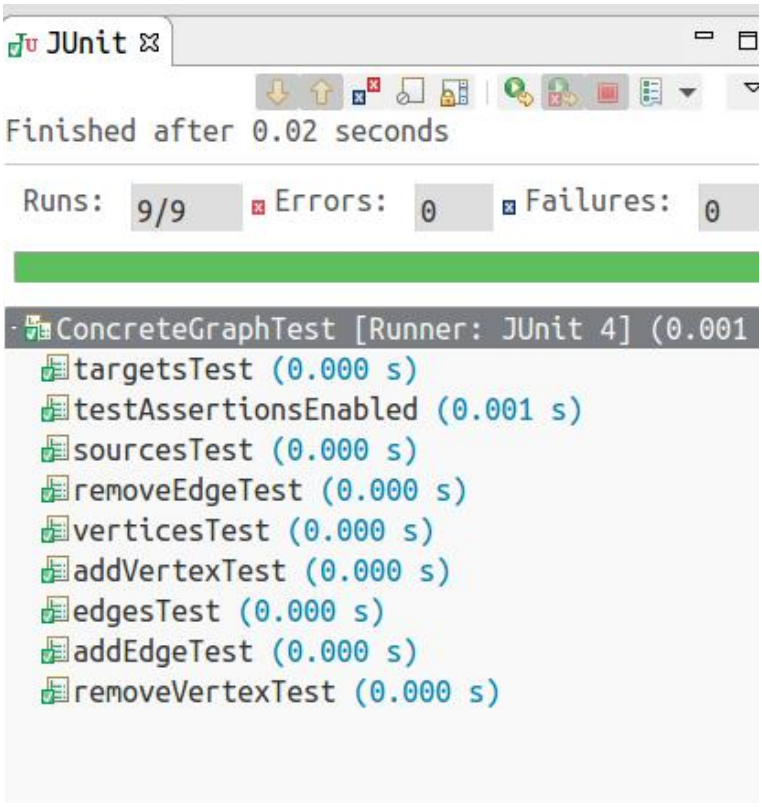
```
GraphType=GraphPoet
VertexType=Word
Vertex=<a,Word>
Vertex=<b,Word>
Vertex=<c,Word>
Vertex=<d,Word>
EdgeType=WordNeighborhood
Edge=<ab,WordNeighborhood,-1,a,b,Yes>
Edge=<ac,WordNeighborhood,1,a,c,Yes>
Edge=<bd,WordNeighborhood,1,b,d,Yes>
Edge=<cd,WordNeighborhood,1,c,d,Yes>

```

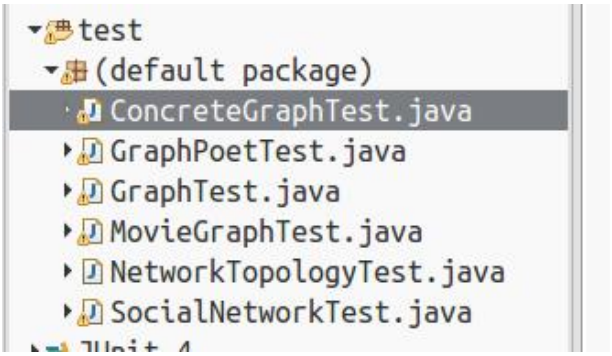
（2）关于点和边的测试

已经在 lab3 中的代码中有体现，所以这里只给出测试结果截图

如下的 test strategy：

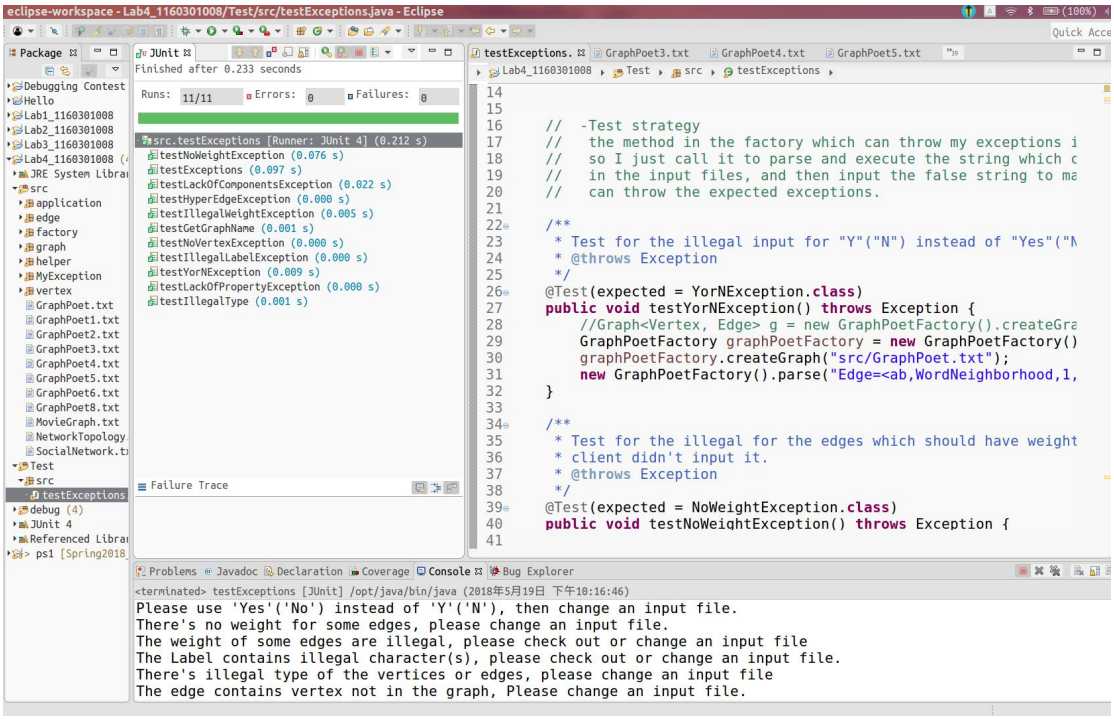
```
public class ConcreteGraphTest {

    // Test strategy:
    //  -Previous remarks: create a graph through class ConcreteGraph, and then create some edges
    //   and vertices. It is worth mentioning that, the edges cannot be "Edge", but some other instance,
    //   such as WordEdge, the vertices is the same. So I used WordEdge and Word here.
    //  -addVertexTest(): create some vertices, first add them into the graph, assert true here, then add
    //   a vertex which has already exists in the graph, assert false here.
    //  -removeVertexTest(): create some vertices and edges, when we call the method removeVertex, we should
    //   pay attention to two things: the vertex is disappeared and the edges which have the vertex we want
    //   delete in its vertices().
    //  -verticesTest(): it is easy to be tested, we can just test the Set which method edges() returned,
    //   test the size and the elements in it.
    //  -sourcesTest() / targetsTest(): this two methods are similar, so them have the similar strategy,
    //   for directed edges, (here I am coding according to directed graph) if the edge is like that:
    //   a ---6---> b, the source is "a" and the target is "b".
    //  -addEdgeTest() / removeEdgeTest(): similarly as the addVertexTest() and the removeVertexTest(),
    //  -edgesTest(): determine the size and elements of the Set which method edges() returned.
    //
    /**
     * Tests that assertions are enabled.
     */
```



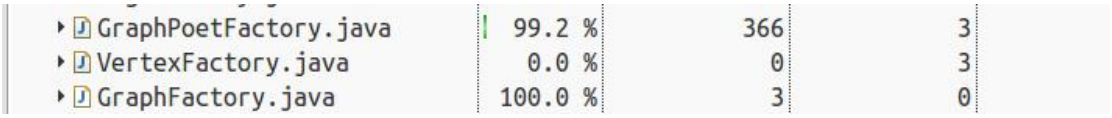


3.4.3 测试运行结果与覆盖度报告

测试结果如下：



覆盖度如下（保证 GraphPoet 为 100%）：



然后看一下 GraphPoetFactory 中的代码，确保都是绿色

```
 1 package factory;
 2
 3 import helper.*;
25
26
27 public class GraphPoetFactory extends GraphFactory {
28     private Graph<Vertex, Edge> graph;
29     private String graphName = new String();
30     private List<String> vertexTypes = new ArrayList<>();
31     private List<String> edgeTypes = new ArrayList<>();
32     public void parse(String string) throws Exception{
33
34         if(string.equals("GraphType=GraphPoet")) {
35             graph = new GraphPoet();
36             //return true;
37         }
38         Matcher matcher = Pattern.compile("GraphName=(\\w+)").matcher(string);
39         if(matcher.matches()) {
40             graphName = matcher.group(1);
41             //return true;
42         }
43         Matcher matcher2 = Pattern.compile("VertexType=(\\w+)").matcher(string);
44         if(matcher2.matches()) {
45             vertexTypes.add(matcher2.group(1));
46             //return true;
47         }
48         Matcher matcher3 = Pattern.compile("Vertex=<(\\w+),Word>").matcher(string);
49         if(matcher3.matches()) {
50             Word word = new Word(matcher3.group(1));
51             graph.addVertex(word);
52             return;
53         }
54         Matcher matcher4 = Pattern.compile("EdgeType=(\\w+)").matcher(string);
55         if(matcher4.matches()) {
56             edgeTypes.add(matcher4.group(1));
57             //return true;
58
```

```
@Override
public Graph<Vertex, Edge> createGraph(String filePath) {
    BufferedReader bf = null;
    String string;
    try {
        bf = new BufferedReader(new FileReader(filePath));
    } catch (Exception e) {
        System.out.println("Wrong path!");
    }
    if(bf != null) {
    try {
        string = bf.readLine();
        while(string != null) {
            try {
            parse(string);
            }
            catch (NoVertexException a) {
                System.out.println(a.getMessage());
                return graph;
            }
            catch (YorNException e) {
                // TODO: handle exception
                System.out.println(e.getMessage());
                return graph;
            }
            catch (IllegalTypeException e) {
                // TODO: handle exception
                System.out.println(e.getMessage());
                return graph;
            }
            catch (NoWeightException e) {
                // TODO: handle exception
                System.out.println(e.getMessage());
```

3.5 FindBugs tool（可选）

首先，发现的第一个 bug，就是在进行文件读取的时候，有可能会有空指针，bugInfo 如下：

```
186                 try {
187
188                     string = bf.readLine();
189                     while(string != null) {
190                         try {
191                         parse(string);
192                         }
193                         catch (NoVertexException e) {
194                             System.out.println(e.getMessage());
195                             return graph;
196                         }
197
```

Problems Javadoc Declaration Coverage Console Bug Explorer Bug Info

NetworkTopologyFactory.java: 188

Navigation

```
Possible null pointer dereference of bf in factory.NetworkTopologyFactory.createGraph(String) on exception path
Dereferenced at NetworkTopologyFactory.java:[line 188]
Value loaded from bf
Null value at NetworkTopologyFactory.java:[line 178]
Known null at NetworkTopologyFactory.java:[line 182]
Known null at NetworkTopologyFactory.java:[line 184]
```

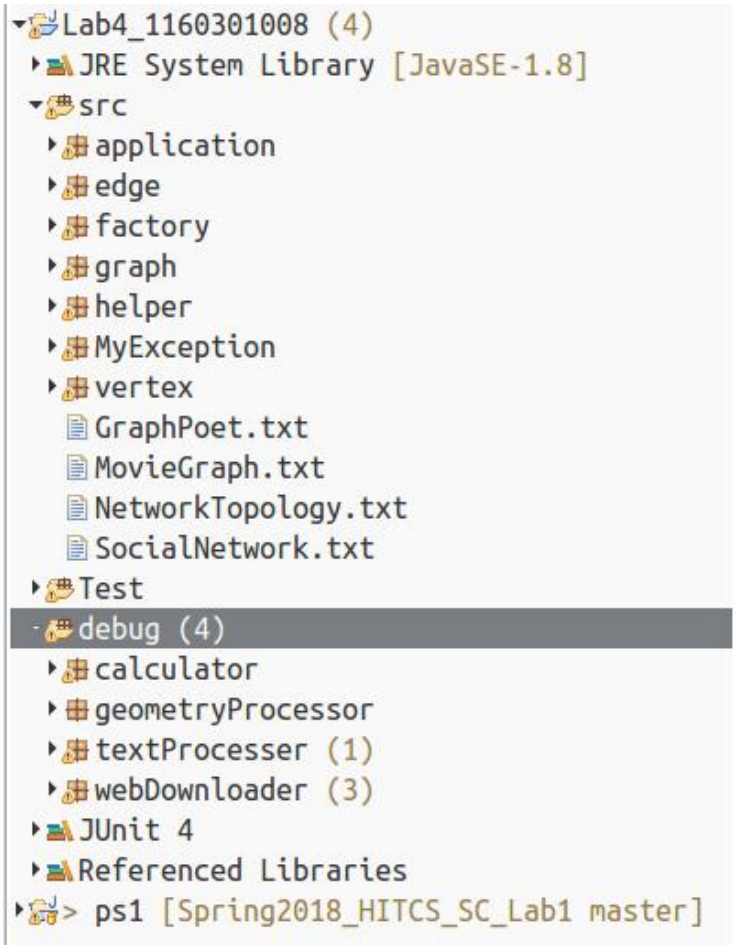
Bug: Possible null pointer dereference of bf in factory.NetworkTopologyFactory.createGraph(String) on exception path

也就是说，我将 bf 初始化为 null，但是其实是不允许为 null 的，所以，为了改正，我进行了如下的修改

```
public Graph<Vertex, Edge> createGraph(String filePath) {
    BufferedReader bf = null;
    String string;
    try {
        bf = new BufferedReader(new FileReader(filePath));
    } catch (Exception e) {
        System.out.println("Wrong path!");
    }
    if(bf != null) {
    try {
        string = bf.readLine();
        while(string != null) {
            try {
            parse(string);
            }
            catch (NoVertexException a) {
```

这样就可以避免 bf 为 nu11，，避免抛出异常。今后我会注意到这一点的。

至此，之前的代码已经将 bug 全部消除。（显示为 4 是 debug 部分的 bug）



3.6 Debugging

3.6.1 待调试程序

（1）CalculatorGUI

（2）GeometryProcessor

（3）textProcessor

（4）webDownloader

3.6.2 理解待调试程序的过程

（1）这个文件其实是一个计算器的 GUI 可视化，首先，发现了不能将数字显示在上面，同时运算符也只能显示“-”，由此意识到应该是将代码修改到可以正常运行为止。

（2）首先实现接口的类发现有代码没有实现，并且在初始化对象的时候没有调用实现类，而是调用了接口类，并根据 SampleOutput 文件，知道应该是要输出一些面积的计算之类的操作。

（3）字典树，在维基百科中查找了字典树的定义，理解了字典树，才知道该怎么去进行 debug。

（4）文件下载，看了 SampleOutput，大概知道了是类似于脚本的东西，自动的去进行文件的下载。

3.6.3 发现并定位错误的过程

（1）首先，先去思考为什么数字和运算符不会显示在 GUI 界面，后来仔细读了一遍代码，大概了解了逻辑，发现了最终的错误原因

```
// buttons
for (int i = 0; i < btnNum.length; i++) {
    //btnNum[i] = new JButton(1 + "");
    btnNum[i] = new JButton(i + "");                    //correct here.
    btnNum[i].setFont(largeFont);
    btnNum[i].addActionListener(new DigitActionListener());
}
```

```
for (int i = 0; i < btnOp.length; i++)
    //pnlOp.add(btnOp[1]);
    pnlOp.add(btnOp[i]);                                //correct here.
```

这两个错误都是将循环变量“i”写成了“1”，导致无法循环的去加入值。

其次，发现了有的方法显示没有被调用，例如下面的方法：

```
// Action listener for digit buttons
class DigitActionListener extends JFrame implements ActionListener {


    /**
     *
     */
    private static final long serialVersionUID = 1L;
```

然后，我看了一下 eclipse 的提供的修改方式，采用自动修改方法，为每一个操作加上了 serialVersionUID，还有很多很细小的 bug，比如“+” “-”写反，“×” “/”逻辑写反，init 没有设置为“true”等等。

（2）这个比较简单，首先就是 eclipse 的报错（语法错误），实例化一个对象的时候，采用了接口类而不是实现了接口的实现类，所以，首先将这里改好

```
for (int i = 0; i < shapes.length; i++) {
    for (int n = 0; n < shapes[i].length; n++) {
        if (n % 3 == 0)
            shapes[i][n] = new Circle(Math.random() * 10, "Red", "Circle" + i + n);

        else if (n % 3 == 1)
            shapes[i][n] = new Square(Math.random() * 10, "Green", "Square" + i + n);
        else if (n % 3 == 2)
            shapes[i][n] = new Triangle(Math.random() * 10, "Blue", "Triangle" + i + n);

    }
}
```

其次，是逻辑上的一些问题，我先读了一边代码，先弄清他的逻辑在实现接口的时候，我发现有的方法没有实现，先将其实现，然后，我运行了一下，发现面积那里算的不太对，最终，我发现了是计算面积的公式写错了，然后进行改正

即可，第二个还是比较好写的。

（3）首先通读代码，发现了在初始化树的时候没有为 pointer 指针初始化，所以改正如下：

```
// Adds a word to the Trie
public void addWord(String s) {
    char[] arr = s.toCharArray();
    pointer = root;                                    //correct here
    for (char c : arr) {
        pointer.addChild(c);
    }
```

然后我运行了一下代码，发现在打印树的信息的时候，打印的是节点的地址值，然后仔细看代码，发现了问题，修改如下：

```
// Displays the current state of the Trie
public void display() {
    pointer = root;
    for (Node n : nodeSet) {
        System.out.println();
        System.out.print("Node Value: " + n.c + ", Children: ");

        for(int i = 0; i < n.childValues().size(); i++) {
            System.out.println(n.childValues().get(i));
        }

        if (n.isLeaf) {
            System.out.print(", isLeaf: " + n.isLeaf + ", hashcode: " + n.hashCo
        }
    }
}
```

然后，发现了一个小的逻辑错误，在执行操作的时候，没有修改计数器的值

```
Scanner in = new Scanner(new File(filename));
// add all words in text to an array
String word;
while (in.hasNext()) {
    word = in.next();

    // page delimeter reached, increment page counter for storing values
    if (word.equals(pageDelimiter)) {
        pageCounter++;
        System.out.println("Page Delimiter Reached, on page" + pageCounter);
        continue;
    } else if (!map.containsKey(word)) {
        trie.addWord(word);
        map.put(word, new LinkedList<Integer>());
        map.get(word).add(pageCounter);
    } else if (map.containsKey(word)) {
        map.get(word).add(pageCounter);
    }
}
```

（4）通读代码，发现了和（2）类似的问题，也是在进行实例化对象的时候采用了接口类，修改之后，发现了一些逻辑问题，将“&&”写成了“||”如下：

```
}

while (choice != 1 && choice != 2 && choice != 3) {

    System.out.println("Choose a file type to download: \n1).pdf\n2).
    choice = in.nextInt();

    switch (choice) {

    case 1:
```

然后发现了打印文件信息的时候不能正常输出文件后缀名，仔细的去看代码之后，发现是边界处理问题，修改如下：

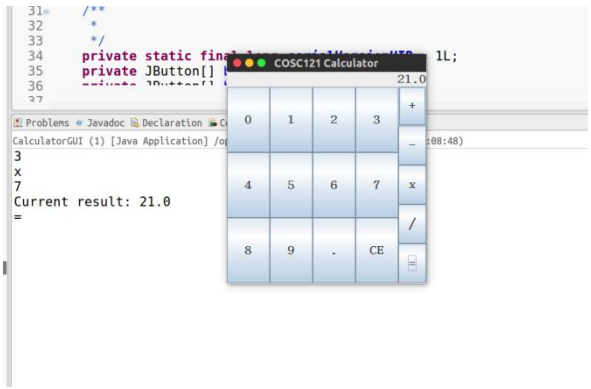
```
    // System.out.println(html.substring(startIndex, endIndex))
    // System.out.println(startIndex + ", " + endIndex);

    // get the substring of the array corresponding to the
    // file's url
    files.add(html.substring(startIndex, endIndex + 1));
    startIndex = 0;
    endIndex = 0;
```

3.6.4 如何修正错误

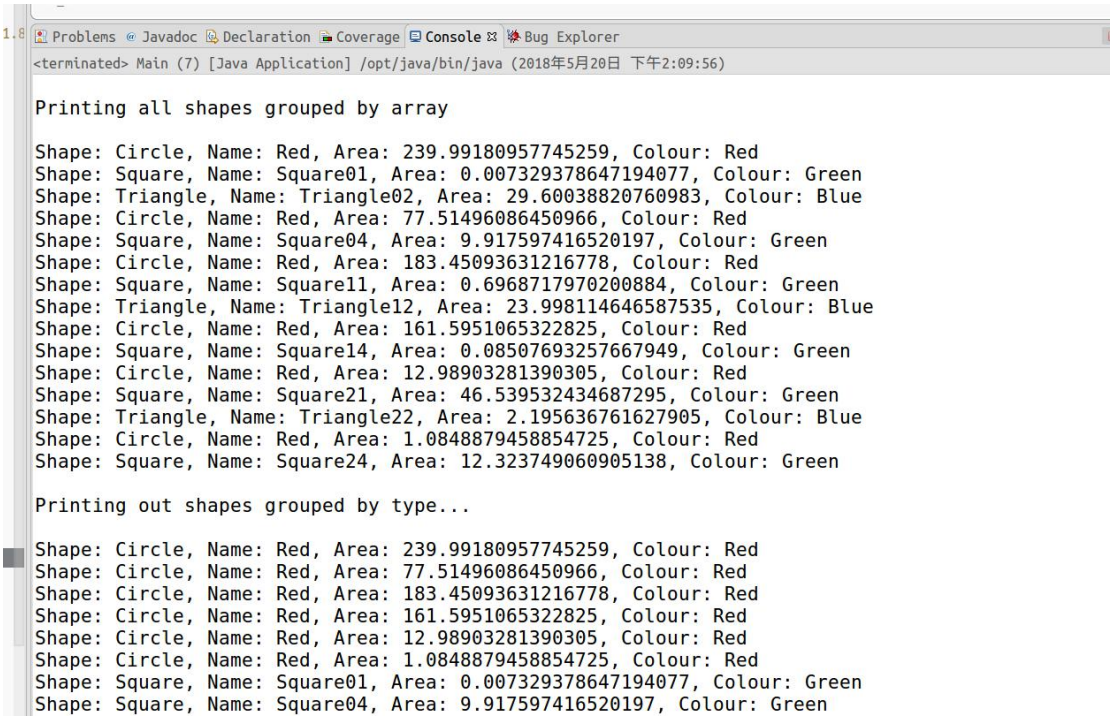
见 3.6.3

3.6.5 结果

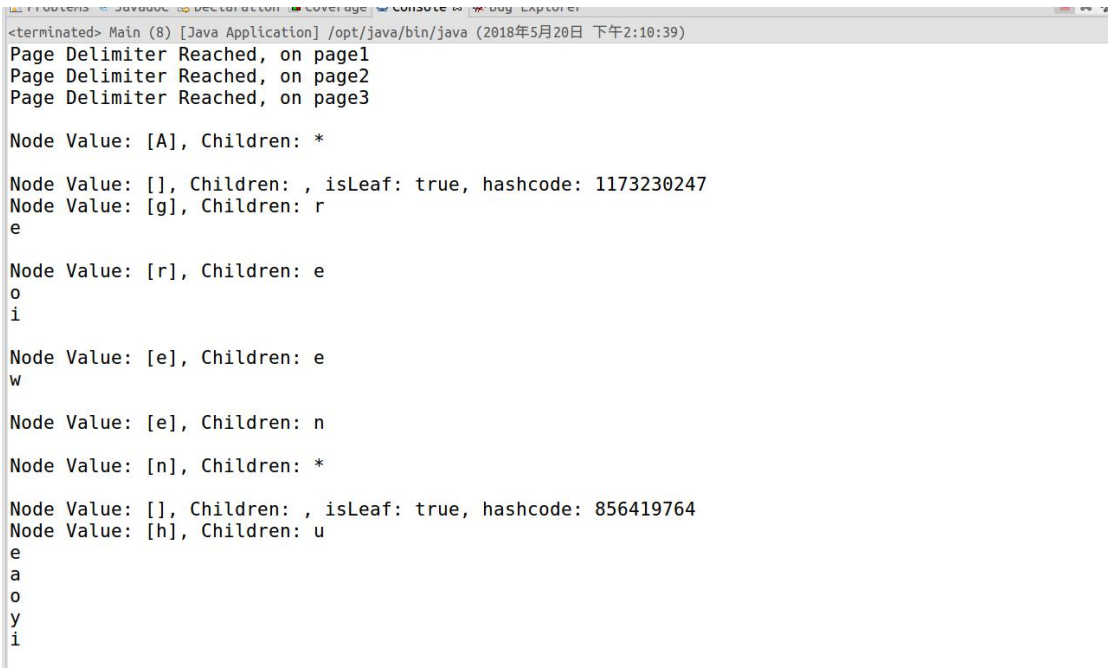
（1）



（2）


Problems Javadoc Declaration Coverage Console Bug Explorer
<terminated> Main (7) [Java Application] /opt/java/bin/java (2018年5月20日 下午2:09:56)
Printing all shapes grouped by array
Shape: Circle, Name: Red, Area: 239.99180957745259, Colour: Red
Shape: Square, Name: Square01, Area: 0.007329378647194077, Colour: Green
Shape: Triangle, Name: Triangle02, Area: 29.60038820760983, Colour: Blue
Shape: Circle, Name: Red, Area: 77.51496086450966, Colour: Red
Shape: Square, Name: Square04, Area: 9.917597416520197, Colour: Green
Shape: Circle, Name: Red, Area: 183.45093631216778, Colour: Red
Shape: Square, Name: Square11, Area: 0.6968717970200884, Colour: Green
Shape: Triangle, Name: Triangle12, Area: 23.998114646587535, Colour: Blue
Shape: Circle, Name: Red, Area: 161.5951065322825, Colour: Red
Shape: Square, Name: Square14, Area: 0.08507693257667949, Colour: Green
Shape: Circle, Name: Red, Area: 12.98903281390305, Colour: Red
Shape: Square, Name: Square21, Area: 46.539532434687295, Colour: Green
Shape: Triangle, Name: Triangle22, Area: 2.195636761627905, Colour: Blue
Shape: Circle, Name: Red, Area: 1.0848879458854725, Colour: Red
Shape: Square, Name: Square24, Area: 12.323749060905138, Colour: Green
Printing out shapes grouped by type...
Shape: Circle, Name: Red, Area: 239.99180957745259, Colour: Red
Shape: Circle, Name: Red, Area: 77.51496086450966, Colour: Red
Shape: Circle, Name: Red, Area: 183.45093631216778, Colour: Red
Shape: Circle, Name: Red, Area: 161.5951065322825, Colour: Red
Shape: Circle, Name: Red, Area: 12.98903281390305, Colour: Red
Shape: Circle, Name: Red, Area: 1.0848879458854725, Colour: Red
Shape: Square, Name: Square01, Area: 0.007329378647194077, Colour: Green
Shape: Square, Name: Square04, Area: 9.917597416520197, Colour: Green

（3）


<terminated> Main (8) [Java Application] /opt/java/bin/java (2018年5月20日 下午2:10:39)
Page Delimiter Reached, on page1
Page Delimiter Reached, on page2
Page Delimiter Reached, on page3
Node Value: [A], Children: *
Node Value: [], Children: , isLeaf: true, hashcode: 1173230247
Node Value: [g], Children: r
e
Node Value: [r], Children: e
o
i
Node Value: [e], Children: e
w
Node Value: [e], Children: n
Node Value: [n], Children: *
Node Value: [], Children: , isLeaf: true, hashcode: 856419764
Node Value: [h], Children: u
e
a
o
y
i

（4）

```
Main (9) [Java Application] /opt/java/bin/java (2018年5月20日 下午2:11:15)
Enter your desired file destination: /home/muty
Choose a file type to download:
1).pdf
2).mp3
3).txt
Choice:
1
Print files before downloading?
1) Yes
2) No
1
304_All_Notes_by6.pdf
304_1_Intro_by6.pdf
304_1_Intro.pdf
304_2_DBIntro_by6.pdf
304_2_DBIntro.pdf
304_3_Relational_by6.pdf
304_3_Relational.pdf
304_4_SQL_DDL_by6.pdf
304_4_SQL_DDL.pdf
304_5_SQL_by6.pdf
304_5_SQL.pdf
304_5_SQL_answers.pdf
304_6_DBDesign_by6.pdf
304_6_DBDesign.pdf
304_7_ER_by6.pdf
304_7_ER.pdf
304_7_ER_by6.pdf
```

4 实验进度记录

请尽可能详细的记录你的进度情况。

日期	时间段	计划任务	实际完成情况

5 实验过程中遇到的困难与解决途径

6 实验过程中收获的经验、教训、感想

本节除了总结你在实验过程中收获的经验和教训，也可就以下方面谈谈你的感受（非必须）：

(1) 健壮性和正确性，二者对编程中程序员的思路有什么不同的影响？

(2) 为了应对 1%可能出现的错误或异常，需要增加很多行的代码，这是否划算？

(3) “让自己的程序能应对更多的异常情况”和“让客户端/程序的用户承担确保正确性的职责”，二者有什么差异？你在哪些编程场景下会考虑遵循前者、在哪些场景下考虑遵循后者？

(4) 过分谨慎的“防御”（excessively defensive）真的有必要吗？

(5) 通过调试发现并定位错误，你自己的编程经历中有总结出一些有效的方法吗？请分享之。Assertion 和 log 技术是否会帮助你更有效的定位错误？

(6) 怎么才是“充分的测试”？代码覆盖度 100%是否就意味着 100%充分的测试？

(7) 关于本实验的工作量、难度、deadline。

(8) 到目前为止你对《软件构造》课程的评价和建议。