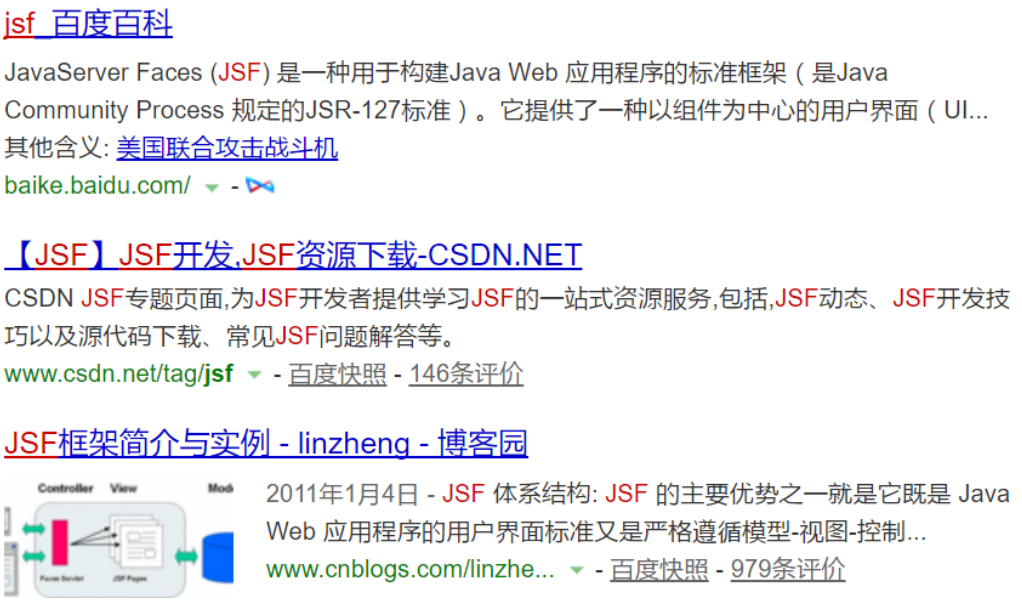
第十一次博客作业

一， 规格化设计的发展历史进程

首先，在查阅资料的过程中，我发现规格化设计这个名词在百度上东西少的可怜，基本没有资料，而查 jsf 出来的东西更是驴头不对马嘴



因此，我认为这个历史指代的英文原词语应该不是 JSF，而是别的某个专有概念。一经查找，我发现了 JML，全称 *Java Modeling Language*，它是一种遵循契约样式的规格化语言，包含了 Hoare style 的前后置条件，不变式等。

The Java Modeling Language (JML) is a specification language for Java programs, using Hoare style pre- and postconditions and invariants, that follows the design by contract paradigm. Specifications are written as Java annotation comments to the source files, which hence can be compiled with any Java compiler.
Various verification tools, such as a runtime assertion checker and the Extended Static Checker (ESC/Java) aid development. [1]

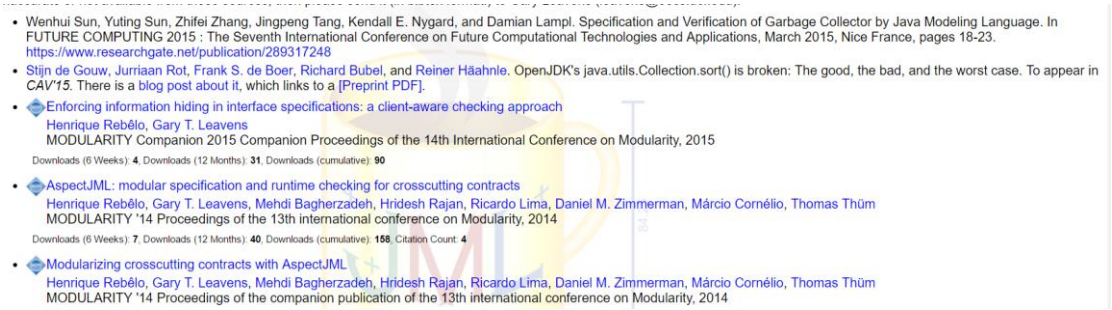
进一步地，分析 JML 的历史渊源。最早关于 JML 的文章已经不可考，但是早在 2002 年就已经出现了自动校验 JML 规范性和根据 JML 检查代码的自动化程序。根据一篇综述性文章 Design by Contract with JML [2] ，JML 起源于 DBC 设计思想（最早描述并将 DBC 规范化变成可实现状态的是 Hoare 在 1969 年的一篇关于形式化验证的文章[3]），也就是 **D**esign **B**y **C**ontract，这种契约式思想可以说规范了类和调用者的权利和义务[5]，一则明确了实现者的管辖范围，二则引导了调用者的正确使用。JML 继承了遵循 DBC 设计的多种规范化语言的设计思想精髓——比如面向模型规范化语言的丰富表意性和，包括 Eiffel[6], Larch, the Refinement Calculus 等，在以上语言的基础上，JML 构建了一套完整的语义学模型来对 Java 模型的行为进行描述，以避免代码行为与设计者的意图出现含混不清的情况。之后，人们不断开发出了自动检查 JML 格式和根据 JML 自动在代码中

插入 assertion 检查，形式化证明检查，不变式和状态检查，运行时检查等[4]的相关工具包括 jmlc，jmlunit，jmldoc 等等。

至于人们为什么重视 JML，从根本原因上来说，相对于 java 语言，JML 是一种更加高级的语言。它屏蔽了部分细节，进行了进一步抽象，使得设计中调用者需要关心的和可以期望得到的部分得以精确地描述和定义，这一点，不仅满足了对调用者权利和义务的范围界定，也方便了设计者实现。同时，JML 引入量词等形式化语言拓展了 JAV 表达式，规范化的语言格式在满足了表意丰富的情况下，提供了多种工具检查的支持，这也是以往 DBC 形式化语言不能做到的。

举个例子来说，如果一个开平方的函数，计算出来小数位保留到七位，那么我们直接阅读代码可能会让调用者迷惑，为什么是 7 位，以后可以改变吗？我只需要 4 位可以吗？而这些问题如果放大到更加复杂的例子中，会很影响实际工程中上下游的调用关系设计，因此，在使用 JML 以后，我们可以定义这个函数就是计算开平方的，位数最高（而不是只能）提供 7 位精确度（给以后代码重写留下解释余地）

因此，JML 也越来越受到人们的重视。下面是一张 JML 相关论文近年来的研究情况，可以看到热度不减当年。



- Wenhui Sun, Yuting Sun, Zhifei Zhang, Jingpeng Tang, Kendall E. Nygard, and Damian Lampl. Specification and Verification of Garbage Collector by Java Modeling Language. In FUTURE COMPUTING 2015 : The Seventh International Conference on Future Computational Technologies and Applications, March 2015, Nice France, pages 18-23. https://www.researchgate.net/publication/289317248
- Stijn de Gouw, Jurriaan Rot, Frank S. de Boer, Richard Bubel, and Reiner Hähnle. OpenJDK's java.utils.Collection.sort() is broken: The good, the bad, and the worst case. To appear in *CAV'15*. There is a blog post about it, which links to a [Preprint PDF].
- Enforcing information hiding in interface specifications: a client-aware checking approach
  Henrique Rebêlo, Gary T. Leavens
  MODULARITY Companion 2015 Companion Proceedings of the 14th International Conference on Modularity, 2015
  Downloads (6 Weeks): 4, Downloads (12 Months): 31, Downloads (cumulative): 90
- AspectJML: modular specification and runtime checking for crosscutting contracts
  Henrique Rebêlo, Gary T. Leavens, Mehdi Bagherzadeh, Hridesh Rajan, Ricardo Lima, Daniel M. Zimmerman, Márcio Cornélio, Thomas Thüm
  MODULARITY '14 Proceedings of the 13th international conference on Modularity, 2014
  Downloads (6 Weeks): 7, Downloads (12 Months): 40, Downloads (cumulative): 158, Citation Count: 4
- Modularizing crosscutting contracts with AspectJML
  Henrique Rebêlo, Gary T. Leavens, Mehdi Bagherzadeh, Hridesh Rajan, Ricardo Lima, Daniel M. Zimmerman, Márcio Cornélio, Thomas Thüm
  MODULARITY '14 Proceedings of the companion publication of the 13th international conference on Modularity, 2014

二， 仅仅只是 *自以为* 不错的类规格*3 和数据规格*3

a) 类规格

```
/*Overview
 * 这个类实现了模拟出租车运行的各种基本功能，包括模拟运动，寻路，状态转换，输出记录信息，
 * 提供接口供外界检查状态等。
 * 抽象函数：将出租车的当前以及历史地理位置映射到对应的运动状态，记录输出信息等
 */

public class Taxi{
```

```
public class Readin implements Runnable{
    /*Overview
     * 这个类主要职能是处理各种输入。
     * 包括了读取控制台的出租车请求，读取改路请求，读取各种地图，并对以上输入的合法性做出初步判断
     * 抽象函数：控制台输入的请求映射到出租车的任务空间，并进行分配
     */
```

```
public class PinkTaxi extends Taxi implements Iterable<String>{
/*Overview
 * 继承父类Taxi并扩展出新的返回双向迭代器查询历史运行记录的功能
 * 抽象函数：该类出租车isnew为真
 */
```

b) 数据规格

```java
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (point == null||next_point == null||current_dst == null||last_state == null
            ||info == null)
        return false;
    return true;
}
```

```java
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (gps == null||flow == null||gui == null||editYourCode == null
            ||taxis == null||stateToint.isEmpty()||blockq == null
            ||request_queue == null||traffic_lights == null)
        return false;
    return true;
}
```

```java
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (order_takenTaxis == null||start_taxis == null||info == null
            ||ranges == null||dst == null||src == null)
        return false;
    return true;
}
```

三， 写的*确实不咋地*的过程规格*5 和数据规格*5 及他们的改进

a) 类规格

来自于 EditYourCode

```java
/*Overview
 * 测试线程的相关函数都在这个类里进行填写和实现。
 * 抽象函数：测试方法调用和init出租车
 */
```

```java
/*Overview
 * 测试线程的相关函数都在这个类里进行填写和实现。
 * 抽象函数：封装测试方法并将初始化的出租车对象映射到调度器出租车队列中
 */
```

来自于 Light

```
/*Overview
 * 读取红绿灯地图，初步检查红绿灯地图合法性（不负责检查是否符合岔路口和30%数量条件，请测试者保证）
 * 初始化信号灯系统
 * 模拟红绿灯运行，提供接口查询红绿灯状态
 * 抽象函数：映射红绿灯地图的01状态到红绿灯实例化对象
 */
```

```
/*Overview
 * 读取红绿灯地图，初步检查红绿灯地图合法性（不负责检查是否符合岔路口和30%数量条件，请测试者保证）
 * 初始化信号灯系统
 * 模拟红绿灯运行，提供接口查询红绿灯状态
 * 抽象函数：映射红绿灯地图中记录的0/1数据到红绿灯实例化对象并模拟红绿灯灯组的运行
 */
```

来自于 Request

```
/*Overview
 * 记录出租车请求的各种信息
 * 输出该请求被处理的过程
 * 协助调度系统进行抢单调度。
 * 抽象函数：从读取请求的阻塞队列映射到出租车的分配队列
 */
```

```
/*Overview
 * 记录出租车请求的各种信息
 * 输出该请求被处理的过程
 * 协助调度系统进行抢单调度。
 * 抽象函数：将新的请求从控制台和测试线程读取请求的阻塞队列映射到出租车的分配队列
 */
```

来自于 FindPath

```
/*Overview
 * 实例化GPS对象的类。保存了地图并承担出租车的找路方法调用接口，记录了地图的邻接矩阵等信息，并提供查询方法。
 * 抽象函数：从两个坐标点映射到一条路径
 */
```

```
/*Overview
 * 实例化GPS对象的类。保存了地图并承担出租车的找路方法调用接口，记录了地图的邻接矩阵等信息，并提供查询方法。
 * 抽象函数：从两个坐标点映射到一条路径，从一个输入的地图映射到地图信息的邻接矩阵和每个点的邻接点HashMap
 */
```

来自于 TestOperation

```
/*Overview
 * 封装了各个测试方法
 * 抽象函数：可以记录各个测试方法被调用产生的信息
 */
```

```
/*Overview
 * 封装了各个测试方法
 * 抽象函数：可以记录各个测试方法被调用产生的信息，
 * 将一条条单独的字符串类型记录映射到链表中方便输出
 */
```

b) 数据规格

来自于 PinkTaxi 类

```
public boolean repOK(){
    /*
     * @REQUIRES:   None;
     * @MODIFIES:   None;
     * @EFFECTS:    Checks if "this" is legal.
     */
    if (!isnew)
        return false;
    return true;
}
```

```
public boolean repOK(){
    /*
     * @REQUIRES:   None;
     * @MODIFIES:   None;
     * @EFFECTS:    Checks if "this" is legal.
     */
    if (!isnew||!super.repOK())
        return false;
    return true;
}
```

来自于 EditYourCode

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (testOperation == null)
        return false;
    return true;
}
```

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal and the initialization is successful.
     */
    if (testOperation == null || pink_count < 30)
        return false;
    return true;
}
```

来自于 Readin

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (map == null||request_count == null||block_rq == null||block_setroad == null)
        return false;
    return true;
}
```

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if the requests reader is working properly
     */
    if (map == null||request_count == null||block_rq == null||block_setroad == null)
        return false;
    return true;
}
```

来自于 TestOperation 类

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (testSys == null||testlog == null)
        return false;
    return true;
}
```

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if the log list and the sys
     *               are initialized properly and "this" is legal.
     */
    if (testSys == null||testlog == null)
        return false;
    return true;
}
```

来自于 Light

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal.
     */
    if (lightmap == null||time_lapse>=200||time_lapse<=500||gui == null)
        return false;
    return true;
}
```

```
public boolean repOK(){
    /*
     * @REQUIRES:    None;
     * @MODIFIES:    None;
     * @EFFECTS:     Checks if "this" is legal and the lapse time is in range.
     */
    if (lightmap == null||time_lapse>=200||time_lapse<=500||gui == null
            ||time_left<0)
        return false;
    return true;
```

以上数据规格和类规格写的不够完善，经分析，主要原因我认为有以下几点：

（1） 先写了代码后补写规格

（2） 表示对象实现概念不够清晰，导致抽象函数写得不好，不知道管理的东西是什么

四， bug 中反映出的过程规格质量问题

（三次作业一共两个 bug，另外一个纯粹个人笔误）

a) 由于出租车随机走动算法问题，一段时间后车辆聚集到地图上方

这个 bug 最主要的原因是前期设计细节没有考虑周全，在循环当前坐标点的邻接点时直接找了其中第一个流量最小的方向，一般就是上。然而在规格书写中由于先写代码后写规格导致完全忽略了这个设计逻辑上的错误。之后将所有流量最小且相同的点都加入队列随机返回，遂解决。

附录：引用文章来源

[1] https://en.wikipedia.org/wiki/Java_Modeling_Language

[2] Leavens G T, Cheon Y. Design by contract with JML[J]. URL, 2004, 32(8):571 - 586.

[3] C. A. R. Hoare. An axiomatic basis for computer programming. Communications of the ACM, 12(10):576–583, October 1969.

[4] Gary T. Leavens, Yoonsik Cheon, Curtis Clifton, Clyde Ruby, and David R. Cok. How the design of JML accommodates both runtime assertion checking and formal verification. Science of Computer Programming, 55(1-3):185– 208, March 2005.

[5] Bertrand Meyer. Applying “design by contract”. Computer, 25(10):40–51, October 1992.

[6] Bertrand Meyer. Eiffel: The Language. Object Oriented Series. Prentice Hall, New York, NY, 1992.