链滴

遇见 PlantUML~

作者：zanwen

原文链接：https://ld246.com/article/1565706602548

来源网站：链滴

前言

来到公司实习也快一个月了，最大的体会就是，虽然大部分时间做的是简单的增删该查，但不同于在校时写的Demo，你要充分考虑程序的鲁棒性（健壮性）、可扩展性（可维护性）、时间/空间复杂度。因为是要实际上线的项目，你需要面面俱到，对团队负责。

于是决定在完成组里任务之余，花时间提高自己的的编码规范、多思考程序设计的可扩展性、性能是可观等。我觉得开发工程师和码农之间的区别是，不仅是复制粘贴和以实现功能为最终目标，还要考如何更优雅的编码、如何实现代码复用而避免重复动作，这是一种追求，也能给工作带来一种工艺、心上的体验。

于是我想从设计模式下手，学习前辈们多年沉淀下来的优雅的编码技巧。之前曾看过《Head First设模式》，当时感觉写的挺好，但是急于求成，一味地莽着看完了，到现在回想起来却也不记得几分。觉得实战和重复是学习的重要因素，纸上得来终觉浅，绝知此事要躬行，还是要花时间认真学一遍啊~

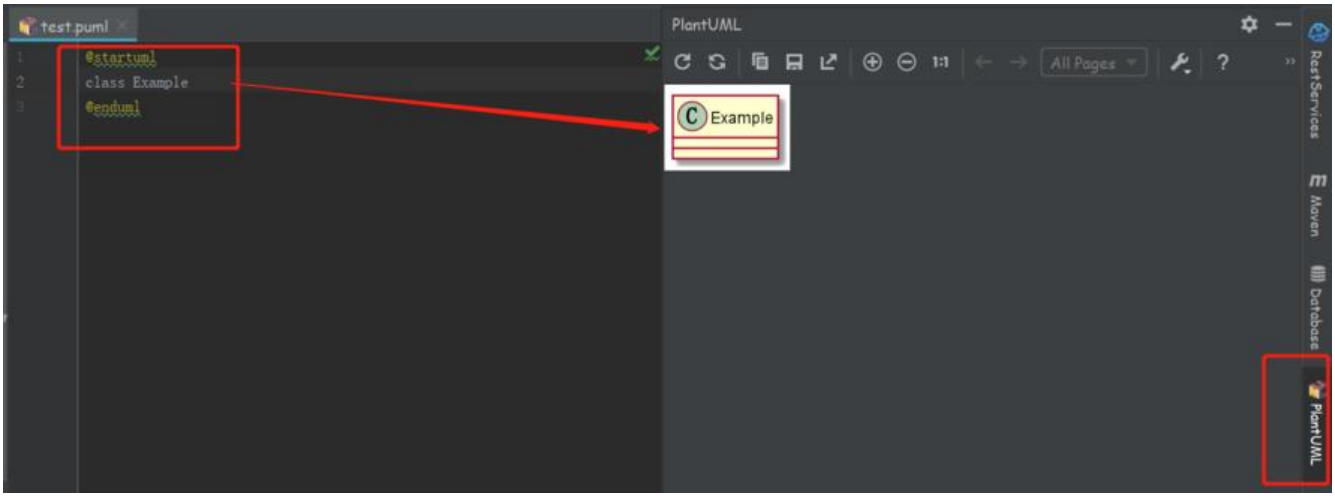
感觉《Head First设计模式》中的内容针对GUI编程的较多，而现在Java主流应用在服务端，再加上过的书除非是像《深入理解Java虚拟机》这种每每“”温故而知新”的，我不太愿意看第二遍，于是目标定在了《图解Java设计模式》上。（本来上当当搜“Java设计模式”发现《Java设计模式及实践好像还不错而且是针对Java服务端的，但奈何太新网盘上还搜不到PDF资源，而我又经济拮据……）

在看设计模式相关的书籍时总有一个感觉，当设计模式中的角色较多时很容易把自己搞混，这时一个晰的Java类图能够高效的帮你理清思路以及如此设计的用意。于是我又折腾的下载了一个IDEA插件PlntUML，自此PlantUML的入坑之路拉开序幕……

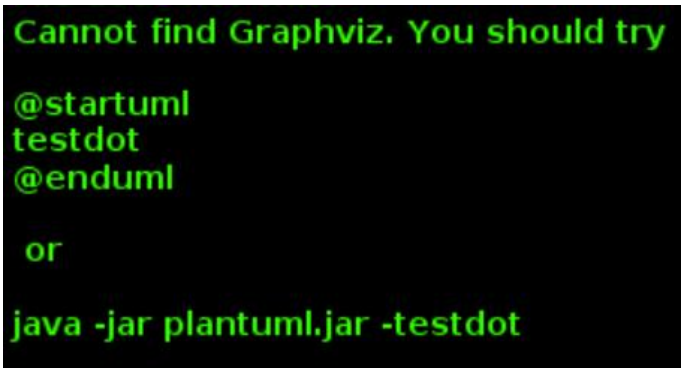
安装及配置

IDEA插件

在IDEA中安装好PlantUML插件（和markdown插件类似，有自己的UML描述语言，在编写后能事实应到UML图中）后，按照官方文档的提示File->New->PlantUML File开始我的第一个类图

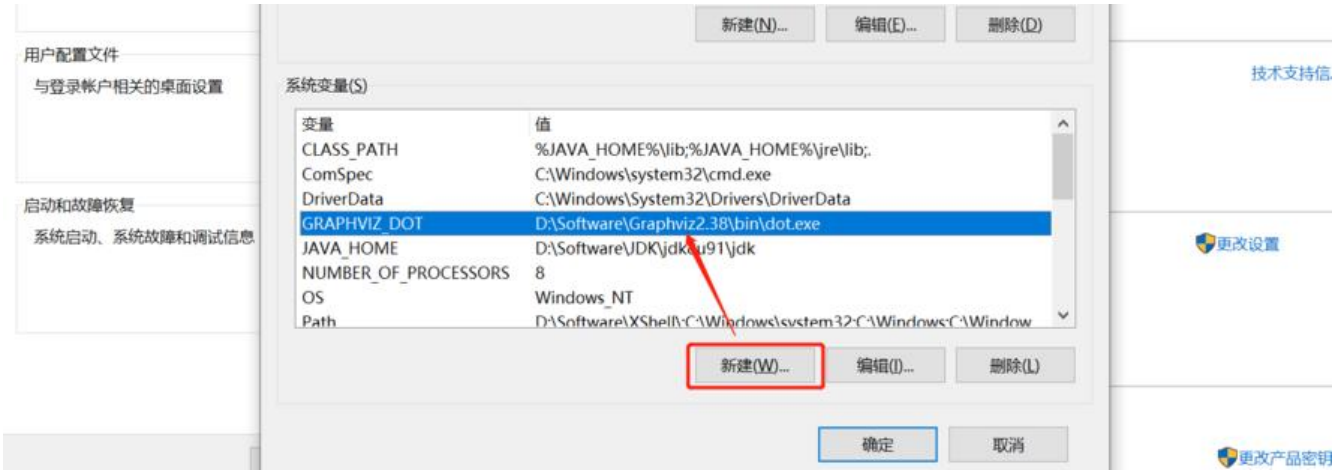


然而，我在编写如上右侧的代码后，右边显示的确实一个异常：

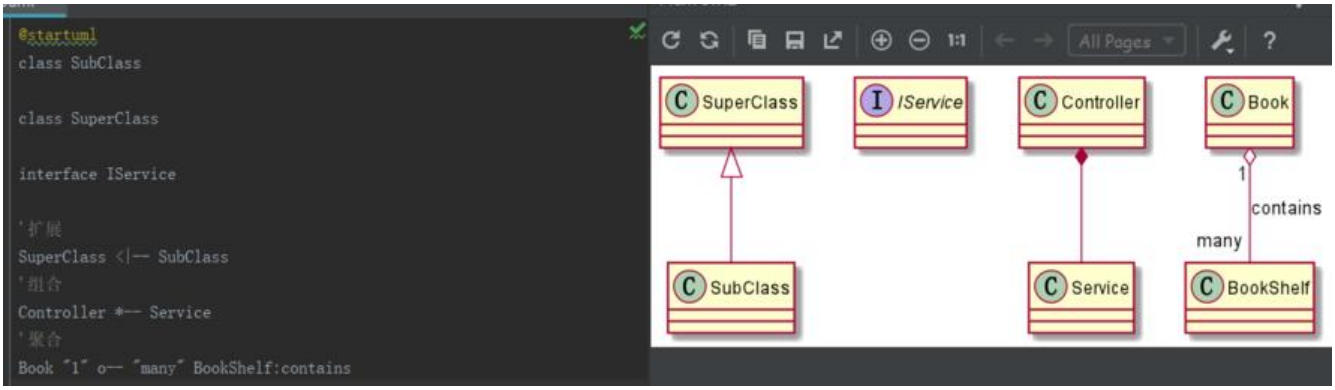


这是因为PlantUML依赖Graphviz，我们需要先下载它（该链接是Windows的.msi安装包，若是其它台，可在参考https://www.graphviz.org/download/）

安装之后需要为其配置环境变量GRAPHVIZ_DOT，为安装目录下的/bin/dot.exe，如图



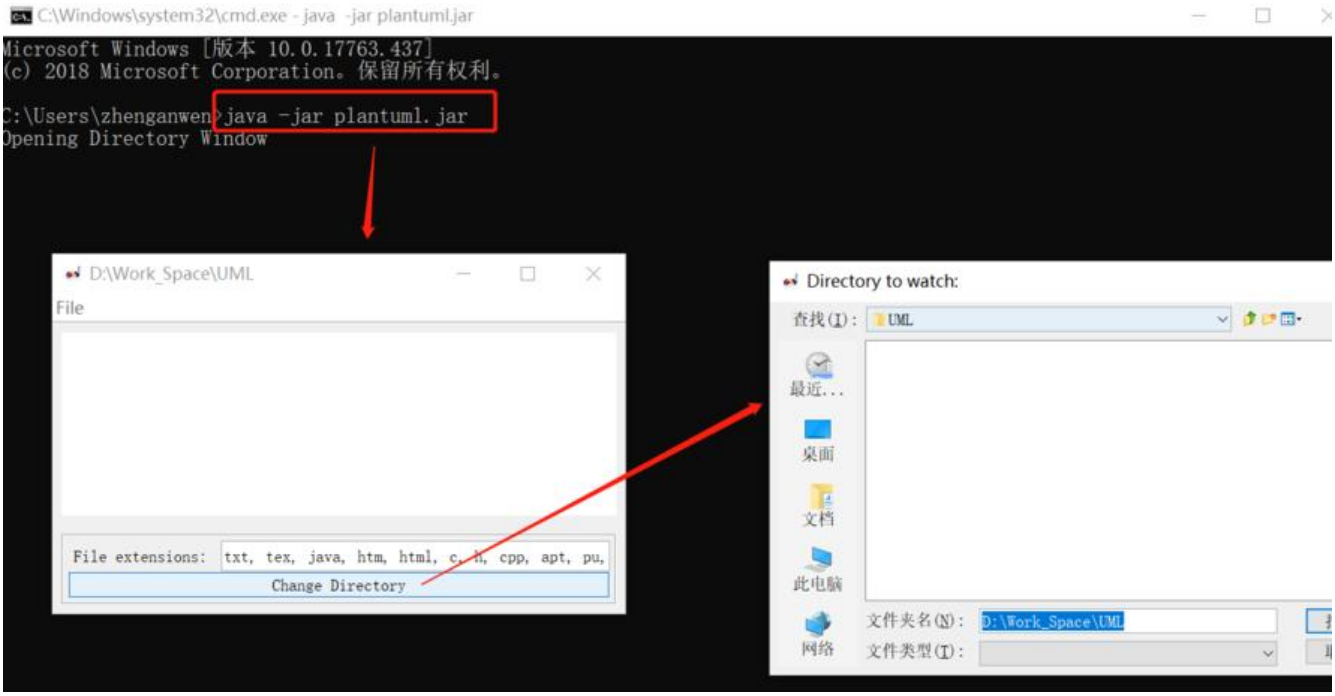
然后重启IDEA，PlantUML标签窗口就能根据你输入的PlangUML language实时绘图了



PlantUML

上节是PlantUML作为插件在IDEA中的使用，作为独立的产品，自然也能够独立运行，本节就介绍通运行jar的方式使用PlantUML

首先下载该产品对应的jar，然后在该jar所在路径运行java -jar plantuml.jar即可启动该应用：

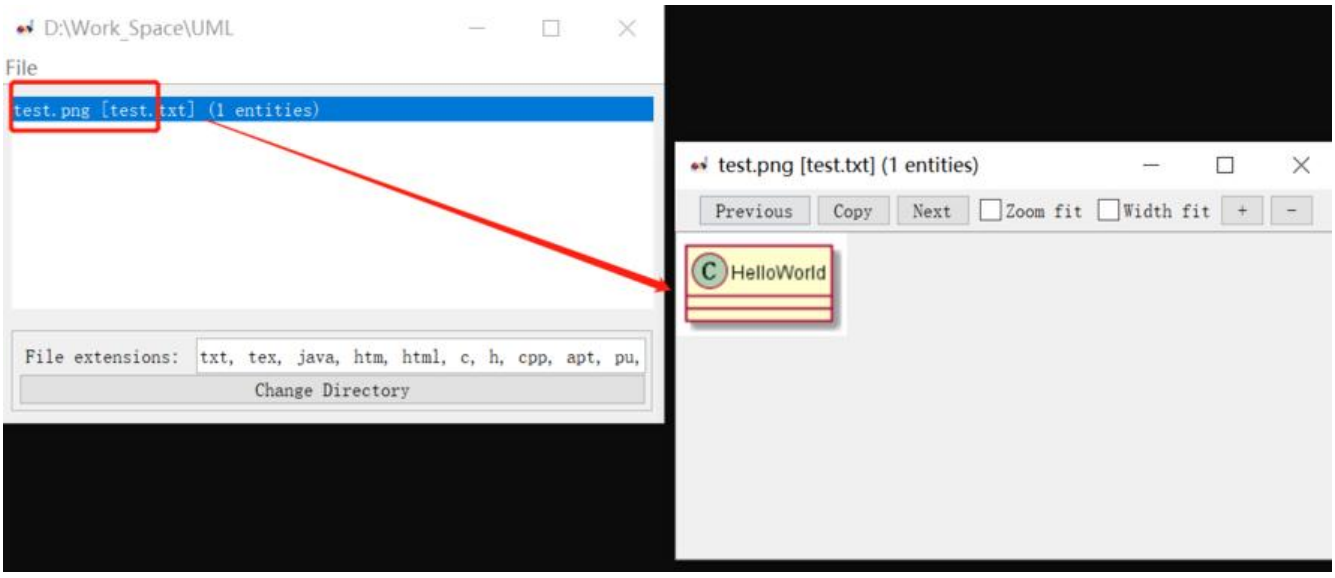


然后点击Change Directory选择一个目录作为workspace，这里我选的是D:\Work_Space\UML，后续lantUML源文件和对应生成的图片都将放在此目录。

在workspace中新建源文件，如test.txt，键入如下代码：

```
@startuml
class HelloWorld
@enduml
```

保存后，在GUI中双击该文件即可打开对应的绘图窗口，并且每次保存源文件都会自动触发重新绘图：



类图

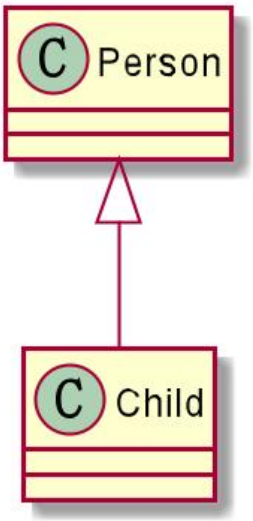
使用PlantUML可以画出很多中图，可参考官方文档，本文仅介绍IDEA中类图的画法。

类之间的关系

泛化/IS-A

IS-A通常用来表继承关系，用空心三角形加实线来表示，语法为Parent <|-- Child：

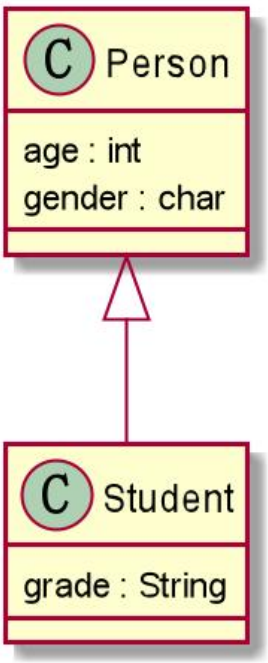
```
@startuml
Person <|-- Child
@enduml
```



你也可以像下面写代码一样表述关系，extends将被识别

```
@startuml
class Person{
    age : int
    gender : char
}

class Student extends Person{
    grade : String
}
@enduml
```



原文链接：遇见 PlantUML~

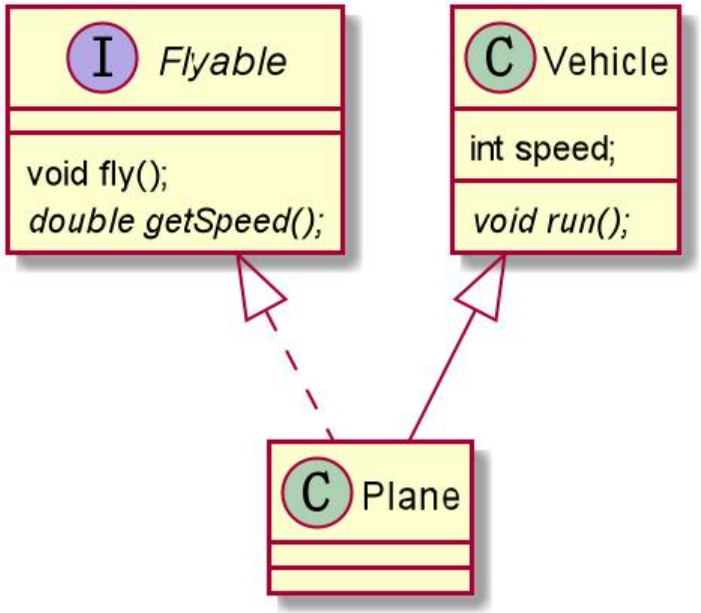
实现

实现一般针对接口，符号为三角形加虚线，语法为Flyable <|.. Plane

```
@startuml
interface Flyable{
    void fly();
    {abstract} double getSpeed();
}

class Vehicle{
    int speed;
    {abstract} void run();
}

Flyable <|.. Plane
Vehicle <|-- Plane
@enduml
```



聚合Aggregation

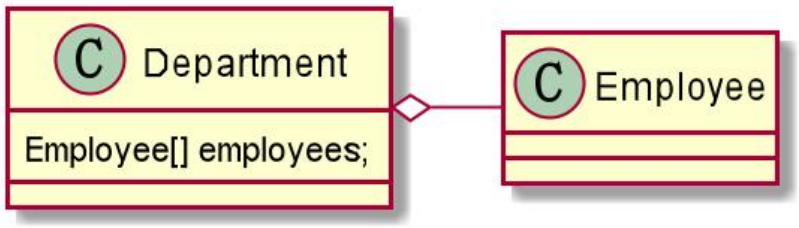
语法：空心菱形加实线Department o-- Employees

Aggregation聚合关系用于表示实体对象之间的关系，表示整体由部分构成的语义，例如一个部门由个员工组成。与组合关系不同的是，整体和部分不是强依赖的，即使整体不存在了，部分仍然存在，如：部门撤销了，人员不会消失，他们依然存在

```
@startuml
class Employee{
}

class Department{
    Employee[] employees;
}
```

```
Department o- Employee
@enduml
```



一个-会画横线，两个-会画竖线，如o--会画竖线

组合Composition

语法：实心菱形+实线ArrayList *- Element

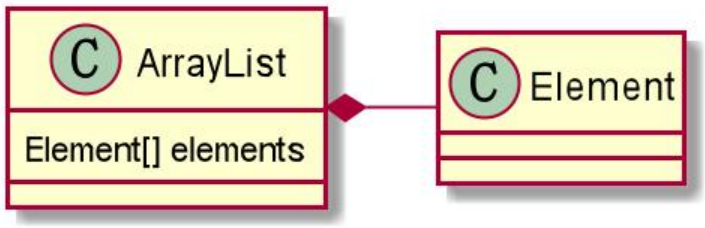
Composition组合关系是一种强依赖的特殊聚合关系，如果整体不存在了，则部分也不存在了，例如公司不存在了，部门也将不存在了

```
@startuml
class Element{

}

class ArrayList{
    Element[] elements
}

ArrayList *- Element
@enduml
```



由于在Java中，内存管理对于我们是透明的。不像C语言，可能我们调用free(company)释放结构体象company的内存，那么其成员departmentList也会被销毁。Java的自动内存管理机制是只要该对到GC Roots是可达的（即能够与引用链连上）那就不会被回收。因此，在Java中，这两种关系的界并不分明。

关联关系Association

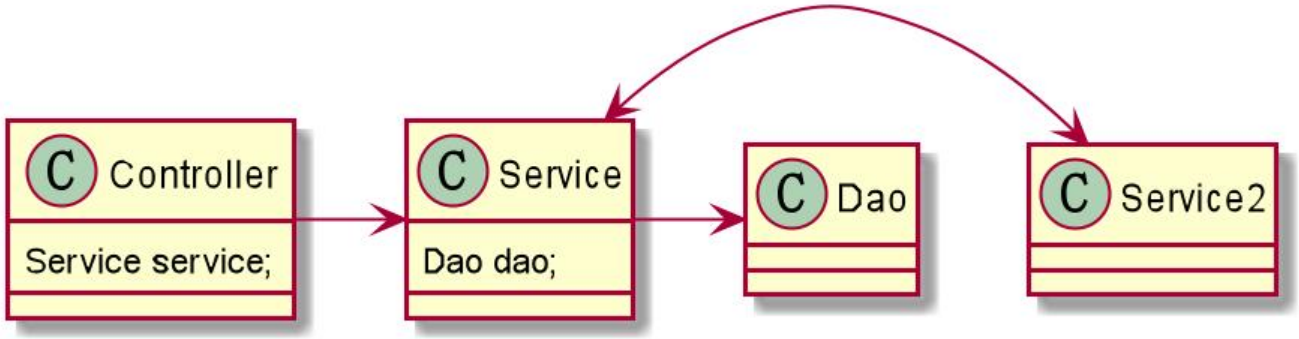
语法：实线+箭头，->/<-或<--/-->或<->/<-->

关联关系是用一条直线表示的；它描述不同类的对象之间的结构关系；它是一种静态关系， 通常与运状态无关，一般由常识等因素决定的；它一般用来定义对象之间静态的、天然的结构； 所以，关联关是一种“**强关联**”的关系；

比如，乘车人和车票之间就是一种关联关系；学生和学校就是一种关联关系；

关联关系默认不强调方向，表示对象间相互知道；如果特别强调方向，如下图，表示A知道B，但 B不道A；

```
@startuml
class Controller{
    Service service;
}
class Service{
    Dao dao;
}
class Service2
class Dao
Controller -> Service
Service <-> Service2
Service -> Dao
@enduml
```



依赖

语法：

依赖也容易和关联弄混，但与关联关系不同的是，它是一种**临时性**的关系，通常在运行期间产生，并随着运行时的变化， 依赖关系也可能发生变化

显然，依赖也有方向，双向依赖是一种非常糟糕的结构，我们总是应该保持单向依赖，杜绝双向依赖产生；

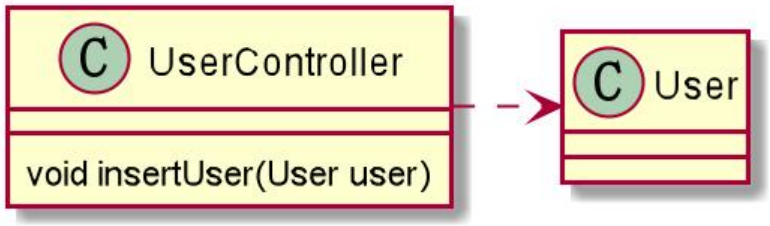
在最终代码中，依赖关系体现为类构造方法及类方法的传入参数，箭头的指向为调用关系；依赖关系了临时知道对方外，还是“使用”对方的方法和属性；

```
@startuml
class UserController{
    void insertUser(User user)
}

class User{

}

UserController .> User
@enduml
```



关于类图的入门介绍到此为止，有兴趣的同学可参考官网进一步学习，并在学设计模式之后借助类图理流程和思路，以达到事半功倍的效果~

参考资料

- 《Head First设计模式》

链接: https://pan.baidu.com/s/1mc2ZmKEpGJwTpna5MbJvzw 提取码: e6a7

- 《图解Java设计模式》

链接: https://pan.baidu.com/s/11p4wRMEpE7OWBrn9oWZ1rw 提取码: 6j9r

- PlantUML语法手册

http://plantuml.com/zh/class-diagram

- 博客

https://design-patterns.readthedocs.io/zh_CN/latest/read_uml.html