

# JClassDiagram 迭代二设计文档

第31组

## 团队介绍

姓名	学号	QQ号	职责
刘畅	221900071	1053432766	组长
梅天豪	221900056	2242974635	组员
潘智杰	221900313	3160400570	组员
庞鸿博	221900314	3231417980	组员

## 项目目标

在迭代二中，进一步完善类图解析工具，增强对复杂Java元素（泛型、容器、数组、抽象类及枚举）的规范化处理能力，并实现关联与依赖关系的粗略分析。

同时，设计并实现三类基于类图的代码质量分析器，支持检测类规模异常（过大/过小类、数据类）、继承结构缺陷及循环依赖问题，最终输出符合PlantUML规范的文本形式类图和代码质量分析报告。

## 系统设计

### 总体设计

```
1  ./src/main/java
2  |─ Main.java
3  |
4  |─ diagram #解析源代码并生成类图的plantUML语句
5  |   |─ ClassDiagram.java      #存储类图中的元素
6  |   |─ ClassDiagramGenerator.java  #生成类图
7  |   |─ ClassParser.java      #生成类图
8  |   |─ SmellAnalyzer.java     #分析软件工程坏味道
9  |
10 |─ graph   #管理和分析java类之间的关系
11 |   |─ ClassMap.java          #解析类级元素之间的关系
12 |   |─ Graph.java            #基于ClassMap进行封装的全局关系管理器
13 |
14 |─ model   #负责定义和封装类、字段和方法的模型数据结构
```

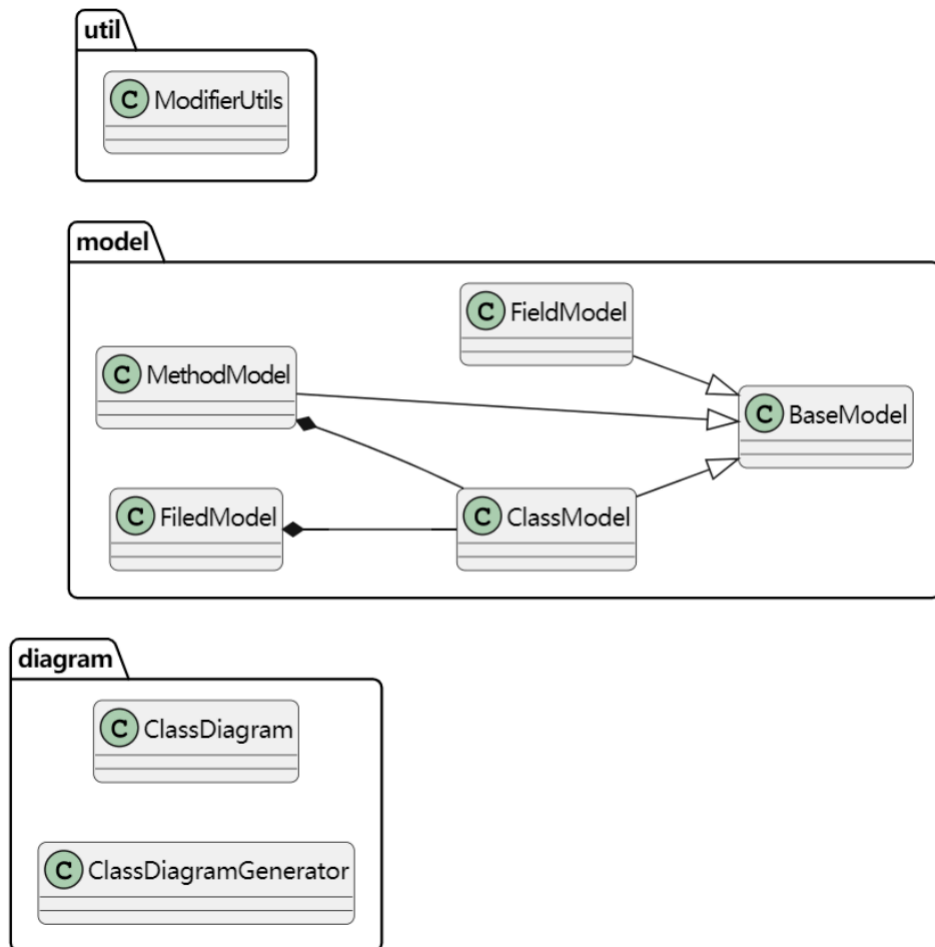
```

15 | |   └─ BaseModel.java          #基础解析模型
16 | |   └─ AbstractClassModel.java #类级元素解析模型
17 | |   └─ ClassModel.java        #提取类的信息
18 | |   └─ EnumModel.java         #提取枚举类的信息
19 | |   └─ InterfaceModel.java    #提取接口的信息
20 | |   └─ FieldModel.java        #提取类图元素的属性
21 | |   └─ MethodModel.java       #提取类图元素的方法
22 | |
23 | └─ utils    #提供辅助工具函数
24 |     └─ AnalyzerUtil.java      #分析软件工程坏味道的工具函数
25 |     └─ CommonUtil.java       #常规工具函数
26 |     └─ Factory.java          #创建类模型对象的工具函数
27

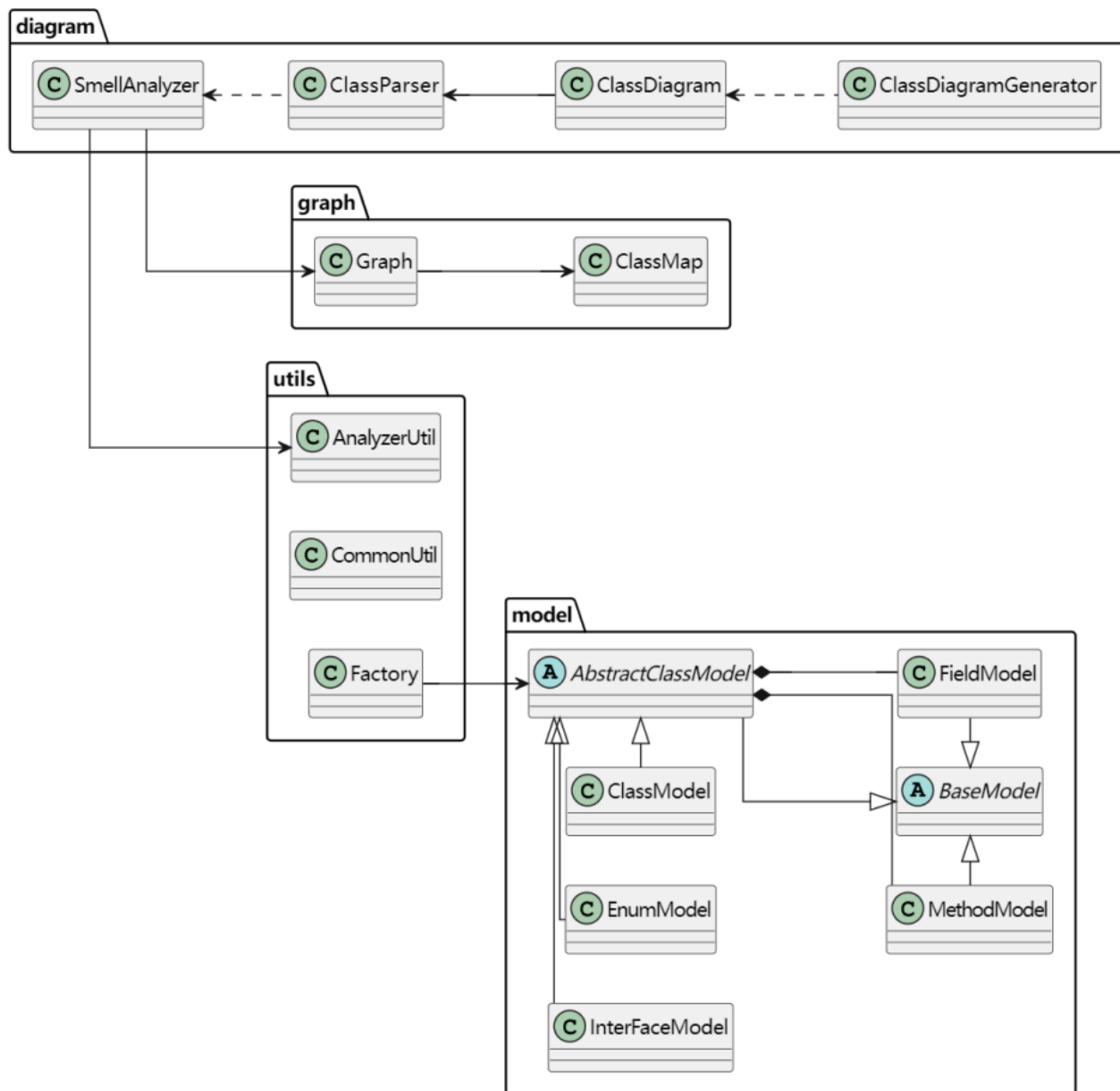
```

## 类设计

### 迭代一类图



## 迭代二类图



## 设计模式

**工厂模式:** `Factory` 类用于统一创建模型对象，避免直接使用构造函数，有利于集中管理对象创建逻辑

**抽象工厂 / 策略模式:** 在 `model` 包中，`AbstractClassModel` 提供统一的抽象接口，子类分别实现不同的行为逻辑。  
`SmellAnalyzer` 可以基于不同子类选择对应策略

**外观模式:** `SmellAnalyzer` 封装了类图解析、模型分析等多个子系统，对外提供简化的接口

弹性设计

工厂机制:

所有模型对象的创建统一通过 Factory 类完成，调用方无需关心具体实例化逻辑。

松耦合依赖关系:

模块之间依赖抽象类或工具类，而非具体实现，例如 SmellAnalyzer 依赖 ClassParser 和 AnalyzerUtil 等工具类，而不是耦合到具体模型类。

支持策略注入或功能扩展:

系统中的分析器、工厂类、工具类等设计支持策略模式和参数化行为扩展。

重要类:

SmellAnalyzer

属性

类型	名称	描述
Graph	graph	表示类之间的关系图结构
List<AbstractClassModel>	classModels	存储所有待分析的类模型
List<String>	output	存储分析后输出的结果内容
AnalyzerUtil	util	工具类

方法

返回类型	方法名	描述
构造方法	SmellAnalyzer(List<AbstractClassModel>, Graph)	构造函数，初始化类模型列表和图结构
List<String>	generateOutput()	执行全部分析流程，返回分析结果列表
void	ClassAnalyze()	检测类的类型异味
void	InheritanceTreeAnalyze()	分析继承结构中的问题
void	CircularDependencyAnalyze()	检测类图中的循环依赖问题

ClassMap

属性

类型	名称	描述
HashMap<String, HashSet<String>>	map	存储类与其关联类之间的映射关系

方法

返回类型	方法名	描述
构造方法	<code>ClassMap()</code>	初始化 map 为空的类图结构
<code>boolean</code>	<code>hasRelation(String src, String dst)</code>	判断是否存在 src 指向 dst 的关系
<code>void</code>	<code>add(String src, String dst)</code>	添加从 src指向 dst 的关系
<code>void</code>	<code>addAll(String src, HashSet&lt;String&gt; dsts)</code>	批量添加 src 指向多个 dst 的关系
<code>HashSet&lt;String&gt;</code>	<code>get(String className)</code>	获取直接父类
<code>HashSet&lt;String&gt;</code>	<code>getReverse(String className)</code>	获取子类
<code>String</code>	<code>generateString(String connectionSymbol)</code>	生成完整关系图,并转成字符串
<code>String</code>	<code>generateStringWithFilter(String connectionSymbol, ClassMap filterMap)</code>	生成过滤指定关系后的关系图
<code>List&lt;String&gt;</code>	<code>getKeys()</code>	获取所有存在映射关系的类名集合
<code>ClassMap</code>	<code>MergeWith(ClassMap other)</code>	将另一个 ClassMap 的内容合并到当前对象中

Factory

方法

返回类型	方法名	描述
<code>AbstractClassModel</code>	<code>classFactory(BodyDeclaration declaration)</code>	创建对应的类模型对象

数据结构与算法设计

**类型怪味检测：**遍历所有类模型，调用各自的方法识别是否为God Class、Lazy Class或Data Class，并分别将结果加入输出列表。

**过深继承树检测：**自根开始DFS遍历，若继承深度过大即标记。

**过宽继承树检测：**遍历每个类，若该类被其他类继承的数量达到10，则标记为“Too Many Children”。

**循环依赖检测：**构建类关系图后，使用DFS遍历各节点，记录访问路径，一旦发现回到起点即判断存在循环依赖，并输出循环路径。

## 附录

## 实际工作安排

姓名	任务	备注
刘畅	编码	
梅天豪	设计与文档	
庞鸿博	编码	
潘智杰	调试与分析	

## 运行结果

