

JClassDiagram 迭代一设计文档

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

git仓库地址

https://git.nju.edu.cn/software_group_31/software_group_31

项目目标

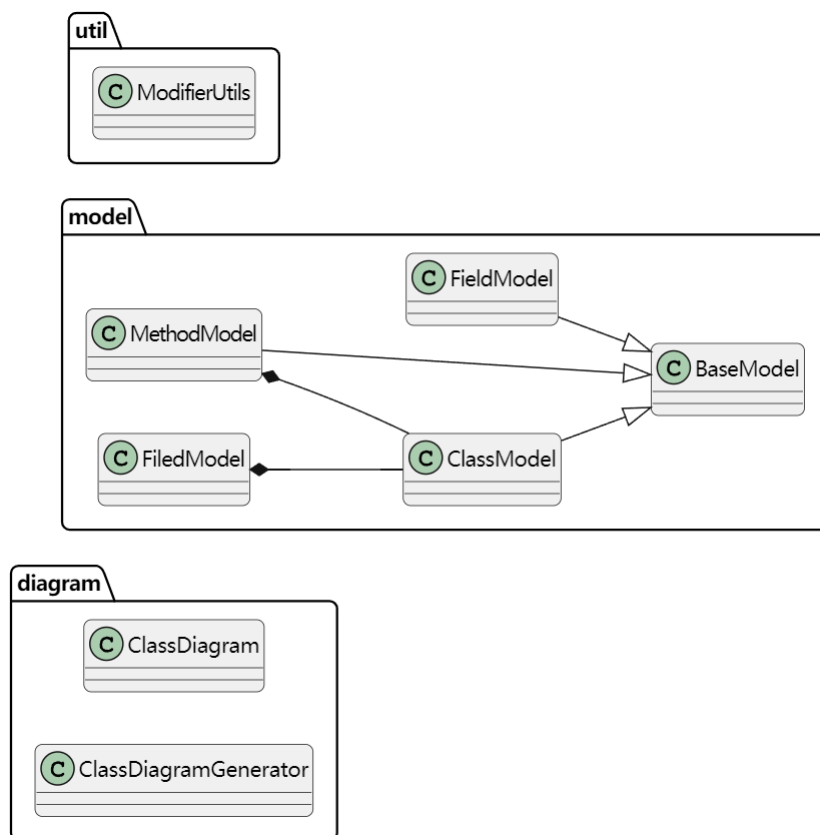
设计并用 Java 代码实现一个 为 Java 代码自动生成 plantUML 类图的工具

系统设计

总体设计

- Diagram 包:解析源代码并生成类图的 plantUML 语句
- Model 包:负责定义和封装类、字段和方法的模型数据结构
- Utils 包: 提供辅助工具函数

类设计



设计模式

- **组合模式**: `ClassModel` 是一个复合对象，它由多个 `FieldModel` 和 `MethodModel` 组成。客户端在处理 `ClassModel` 时不需要关心它内部是如何组织字段和方法的。
- **模板方法模式**: `BaseModel` 类作为一个基类，提供了基本的框架和实现，`ClassModel`、`FieldModel` 和 `MethodModel` 继承 `BaseModel` 并实现了具体的细节

弹性设计

- **模块化编程**: 项目中每个包关注不同的职责，体现了模块化的设计思路。
- **面向接口编程**: `ClassModel`、`FieldModel` 和 `MethodModel` 类都继承自 `BaseModel`，这种设计允许其他类型的模型继承 `BaseModel`，而不破坏现有系统。

数据结构与算法设计

代码解析: 系统使用 `JavaParser` 库来解析源代码,将 Java 源代码解析成抽象语法树(AST)。在解析过程中，源代码会被转换成 `CompilationUnit` 对象,作为 AST 的根节点。对于不同的数据模型，递归地传入对应的语法树节点。

关系分析: 使用 `ClassOrInterfaceDeclaration` 的 `getExtendedTypes` 与 `getImplementedTypes` 方法得到继承与依赖方法。

结果输出: 目前采用 `StringBuilder`,对于模型类覆写了 `ToString()`方法。后续考虑在 `utils` 中添加新的输出工具类以统一输出模式。

结果排序: 通过哈希表对优先级预先映射，按照

附录

实际工作安排

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

运行结果

Test 运行结果

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running MixTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.223 s -- in MixTest
[INFO] Running MultiClassTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.015 s -- in MultiClassTest
[INFO] Running RelationshipTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.056 s -- in RelationshipTest
[INFO] Running SingleClassTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.020 s -- in SingleClassTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.534 s
[INFO] Finished at: 2025-03-20T14:34:24+08:00
[INFO] -----
```

oj 运行结果

[View or edit group](#)

Total Points

- / 100 pts

Autograder Score

100.0 / 100.0

Passed Tests

MixTest.MixTest.testAnimal (5/5)

MixTest.MixTest.testInheritanceOverride (5/5)

MixTest.MixTest.testInterfacePolymorphism (5/5)

MixTest.MixTest.testDeepInheritance (5/5)

MixTest.MixTest.testAccessModifierMix (5/5)

MixTest.MixTest.testMultiInterfaceWithInheritance
(5/5)

MixTest.MixTest.testDeepHierarchy (5/5)

MixTest.MixTest.testInterfaceInheritance (5/5)

RelationshipTest.RelationshipTest.testInterfaceInheritance
(5/5)

RelationshipTest.RelationshipTest.testInheritance
(5/5)

RelationshipTest.RelationshipTest.testImplement
(5/5)

RelationshipTest.RelationshipTest.testProduct (5/5)

SingleClassTest.SingleClassTest.testEmpty (5/5)

SingleClassTest.SingleClassTest.testField (5/5)

SingleClassTest.SingleClassTest.testMultiField (5/5)

SingleClassTest.SingleClassTest.testMethod (5/5)

SingleClassTest.SingleClassTest.testSingleClass (5/5)

MultiClassTest.MultiClassTest.testMultiEmpty (5/5)

MultiClassTest.MultiClassTest.testMultiClass (5/5)

MultiClassTest.MultiClassTest.testMultiInterface (5/5)

运行实例(以 Animal.java 为例)

```
1 @startuml
2 class Animal {
3     - name: String
4     + {static} count: int
5     + getName(): String
6     + setName(name: String): void
7 }
8 class Dog {
9     + gender: String
10    + wolf(): void
11 }
12 class Bird {
13     + color: String
14     + fly(): void
15 }
16 interface Flyable {
17     + fly(): void
18 }
19 Animal <|-- Dog
```

```
20 Animal <|-- Bird
21 Flyable <|.. Bird
22 @enduml
```