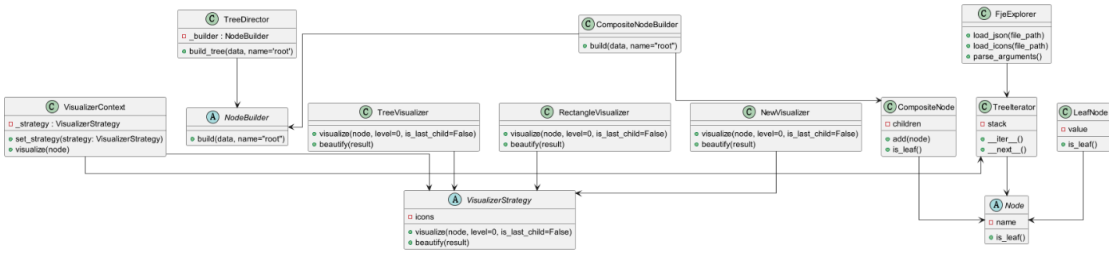
Funny JSON Explorer 设计文档

1.绘制 uml 类图

绘制代码位于`class_pic.puml`。



* `fje.py`: 负责加载 JSON 文件和图标文件以及解析命令行参数。
* `iterator.py`: 定义用于遍历树结构的迭代器。
* `strategy.py`: 定义可视化策略的抽象类以及具体策略的实现类。
* `context.py`: 定义上下文类，用于设置和使用具体的可视化策略。
* `builder.py`: 定义节点建造者的抽象类以及具体的构建器实现类，实现树构建指挥类，用于使用构建器来创建树结构。
* `node.py`: 定义节点的抽象类以及具体的节点实现类。
* `example.json`: 测试可视化结果的代码。
* `icons.json`: 图标族的配置文件。

2.使用迭代器和策略模式来实现 FJE

迭代器模式将用于遍历树结构，策略模式将用于选择和应用不同的可视化风格。迭代器模式：`TreeIterator`类用于遍历树结构，`VisualizerContext`类中使用了这个迭代器来遍历树节点。策略模式：定义了`VisualizerStrategy`接口以及三个具体策略类(`TreeVisualizer`、`RectangleVisualizer`和`NewVisualizer`)，通过`VisualizerContext`类使用不同的策略来实现不同的可视化效果。

迭代器模式用于遍历集合对象的元素，而不需要暴露其底层表示。
在代码中，迭代器模式通过`TreeIterator`类实现，`TreeIterator`通过栈结构深度优先遍历树。`__iter__`方法使`TreeIterator`类成为一个可迭代对象，`__next__`方法则实现了迭代器协议。
策略模式定义了一系列算法，并将每个算法封装起来，使它们可以相互替换。策略模式使得算法可独立于使用它的客户而变化。
在代码中，策略模式通过以下类来实现，这里定义了三个具体策略类`TreeVisualizer`、`RectangleVisualizer`和`NewVisualizer`，它们都实现了`VisualizerStrategy`接口。

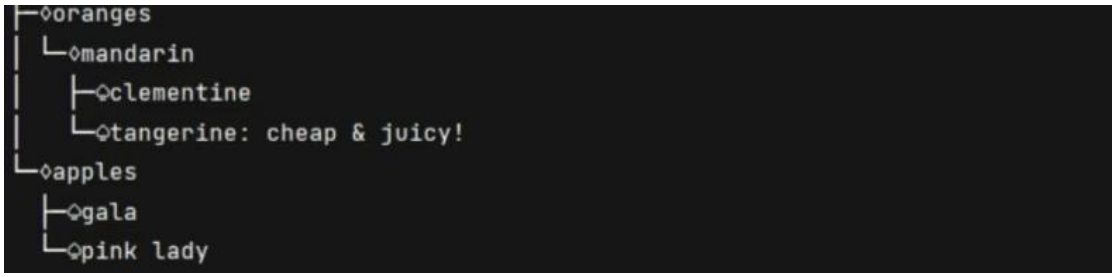
3.结果展示

不改变现有代码，只需添加新的抽象工厂，即可添加新的风格。在`strategy.py`中，通过继承类**VisualizerStrategy**并重写**visualize**方法即可添加新的风格。 示例如下，创建一个新的风格 NewVisulizer，并在具体可视化工厂中添加创建实际的工厂选项，在执行脚本时，参数`-s`后跟新添加的可视化风格名称即可使用新的可视化风格。通过配置文件，可添加新的图标族。在`icons.json`中添加新的图标，在执行脚本时，参数`-i`后跟

新添加的图标组名称即可使用新的图标。

Tree

poker



star

Rectangle

poker

star