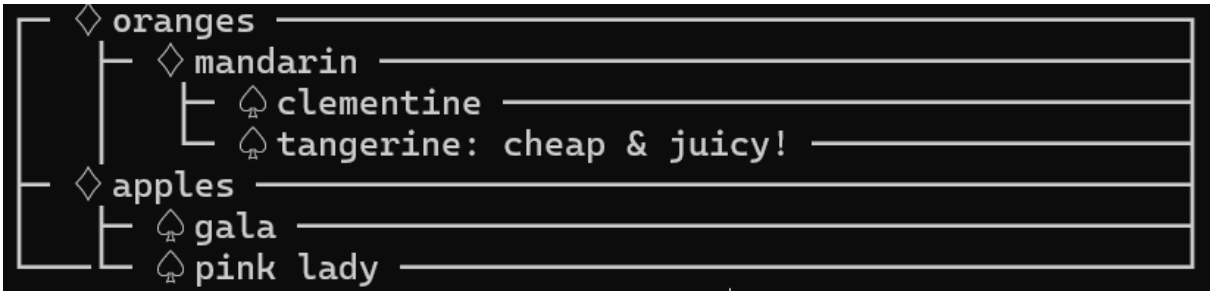


Funny JSON Explorer

21307259 彭璇

开发一个名为Funny JSON Explorer（FJE）的命令行小工具，以可视化JSON文件。我们将实现以下需求：

1. **可视化风格切换**：支持树形（tree）和矩形（rectangle）两种风格。
2. **图标族切换**：支持不同的图标族，例如poker-face-icon-family。
3. **命令行参数**：用户可以通过命令行指定JSON文件、风格和图标族。

📢 目录

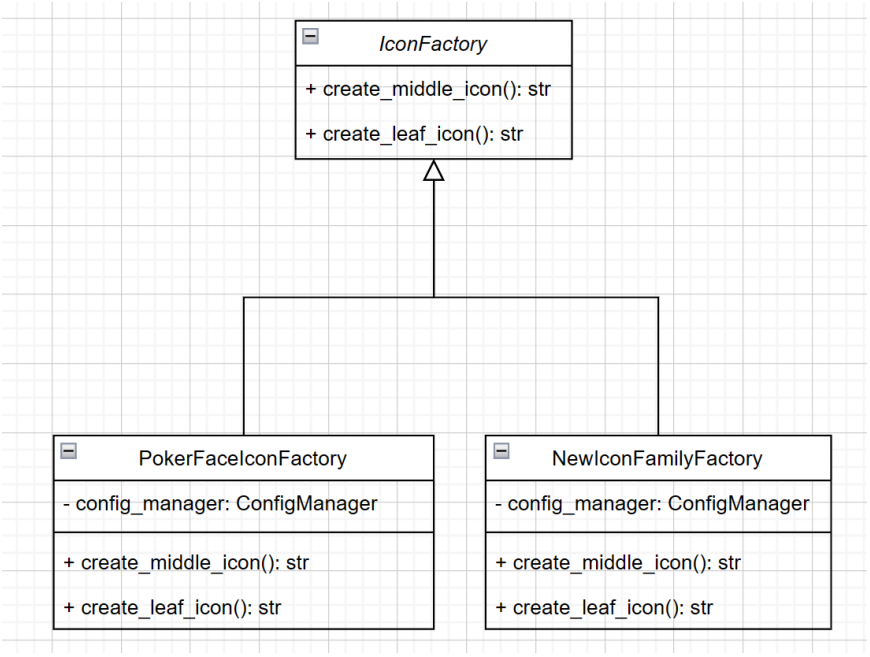
设计模式

1. 工厂方法（Factory Method）：
    - 用于创建不同风格的可视化器。
2. 抽象工厂（Abstract Factory）：
    - 用于创建不同图标族的图标。

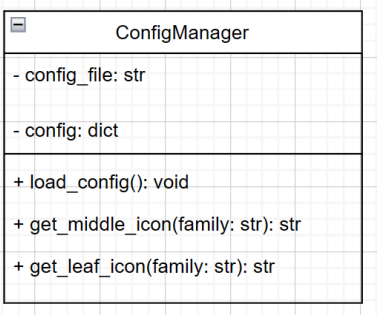
3. 建造者（Builder）模式：
    - 用于构建复杂的可视化结构。
4. 组合模式（Composite Pattern）：
    - 用于表示树形结构的节点和叶子。

类图

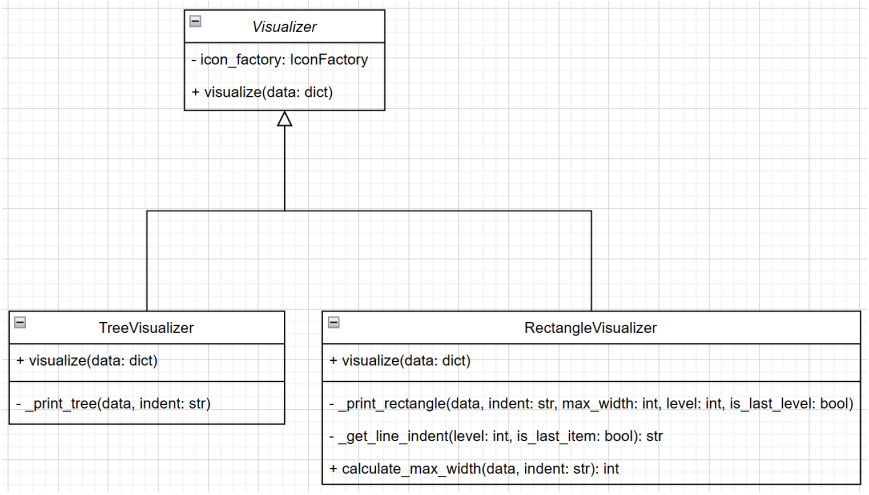
工厂方法和抽象工厂模式的类图如下：



建造者模式的类图如下：



组合模式的类图如下：



设计解析

工厂方法（Factory Method）

工厂方法模式用于定义一个创建对象的接口，但由子类决定要实例化的类是哪一个，我设计在 `IconFactory` 和其子类中体现：

- **IconFactory (抽象类)** 定义了两个抽象方法：

```
from abc import ABC, abstractmethod

class IconFactory(ABC):
    @abstractmethod
    def create_middle_icon(self) -> str:
        pass

    @abstractmethod
    def create_leaf_icon(self) -> str:
        pass
```

- **PokerFaceIconFactory 和 NewIconFamilyFactory (具体工厂类)** 实现了这些方法：

```
class PokerFaceIconFactory(IconFactory):
    def __init__(self, config_manager: ConfigManager):
        self.config_manager = config_manager

    def create_middle_icon(self) -> str:
        return self.config_manager.get_middle_icon("poke
```

```
r-face")

    def create_leaf_icon(self) -> str:
        return self.config_manager.get_leaf_icon("poker-
face")

class NewIconFamilyFactory(IconFactory):
    def __init__(self, config_manager: ConfigManager):
        self.config_manager = config_manager

    def create_middle_icon(self) -> str:
        return self.config_manager.get_middle_icon("new-
icon-family")

    def create_leaf_icon(self) -> str:
        return self.config_manager.get_leaf_icon("new-ic
on-family")
```

抽象工厂（Abstract Factory）

抽象工厂模式用于提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类，`IconFactory` 可以被视为抽象工厂，定义了创建相关图标的方法。

- **IconFactory (抽象工厂)**
- **PokerFaceIconFactory 和 NewIconFamilyFactory** 是具体工厂，分别创建不同风格的图标。

建造者（Builder）

建造者模式用于将一个复杂对象的构建过程与它的表示分离，使得同样的构建过程可以创建不同的表示。`ConfigManager` 为建造者负责从配置文件中加载并提供具体图标。

- **ConfigManager (建造者)** 负责加载配置并提供图标：

```
class ConfigManager:
    def __init__(self, config_file):
        self.config_file = config_file
        self.config = {}

    def load_config(self):
```

```
        with open(self.config_file, 'r') as file:
            self.config = json.load(file)

    def get_middle_icon(self, family: str) -> str:
        return self.config[family]['middle_icon']

    def get_leaf_icon(self, family: str) -> str:
        return self.config[family]['leaf_icon']
```

组合（Composite）

组合模式用于将对象组合成树形结构以表示“部分-整体”的层次结构。组合模式使得用户对单个对象和组合对象的使用具有一致性，这个模式在 `TreeVisualizer` 和 `RectangleVisualizer` 中体现。

- **TreeVisualizer 和 RectangleVisualizer** 通过递归调用，处理字典和列表，形成树状结构，仅展示TreeVisualizer 的实现：

```
class TreeVisualizer(Visualizer):
    def _print_tree(self, data, indent=""):
        if isinstance(data, dict):
            total_items = len(data)
            for index, (key, value) in enumerate(data.it
ems()):
                icon = self.icon_factory.create_middle_i
con()
                if isinstance(value, dict):
                    if index == total_items - 1:
                        print(f"{indent}└─ {icon} {ke
y}")
                        self._print_tree(value, indent +
"    ")
                    else:
                        print(f"{indent}├─ {icon} {ke
y}")
                        self._print_tree(value, indent +
"│   ")
                else:
                    icon = self.icon_factory.create_leaf
```

```
_icon()
                    if value is not None:
                        if index == total_items - 1:
                            print(f"{indent}└─ {icon} {k
ey}: {value}")
                        else:
                            print(f"{indent}├─ {icon} {k
ey}: {value}")
                    else:
                        if index == total_items - 1:
                            print(f"{indent}└─ {icon} {k
ey}")
                        else:
                            print(f"{indent}├─ {icon} {k
ey}")
        elif isinstance(data, list):
            for item in data:
                self._print_tree(item, indent)
        else:
            if data is not None:  # Skip displaying null
nodes
                icon = self.icon_factory.create_leaf_ico
n()
                print(f"{indent}└─ {icon} {data}")
```

添加新的图标簇

设计在`config.txt`中记录图标选项：

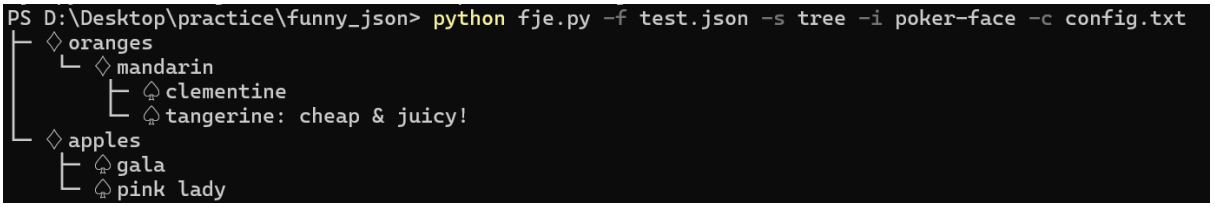
```
poker-face ♢ ♤
new-icon-family ▲ ▼
```

可以修改`new-icon-family`内容来添加新的图标簇。

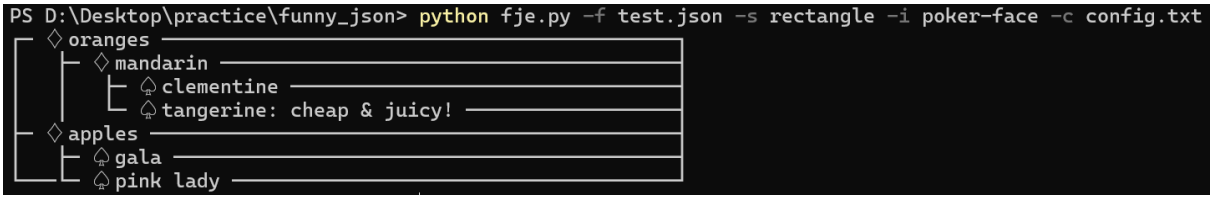
运行截图

使用`python fje.py [-h] -f FILE -s {tree,rectangle} -i ICON [-c CONFIG]`运行。

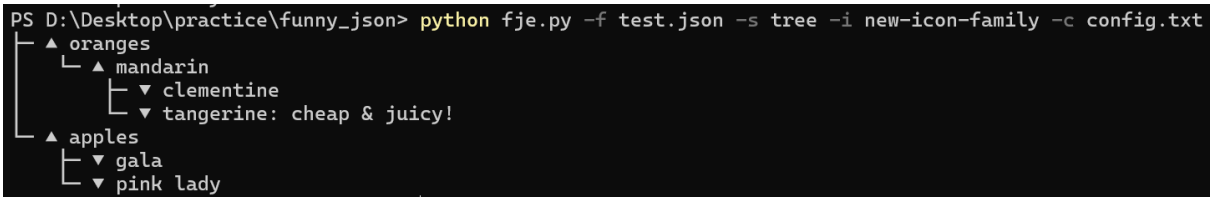
扑克图标+树形结构：



扑克图标+矩形结构：



自定义图标+树形结构：



自定义图标+矩形结构：