

1 设计文档

代码仓库：[Jareddddd/FunnyJsonExplorer: Funny JSON Explorer \(FJE\)](#)，是一个JSON文件可视化的命令行界面小工具([github.com](#))

提交截至日期前仓库为private

设计文档：类图与说明，说明使用的设计模式及作用

使用工厂方法（Factory）、抽象工厂（Abstract Factory）、建造者（Builder）模式、组合模式（Composition），完成功能的同时，使得程序易于扩展和维护。

深刻理解老师上课说过的一句话：过早的开始是万恶的根源

2 V1

这一个版本是在5.31号之前已经完成的，在这之前我以为是上述四个方法中任选几个方法来实现，所以我只选了工厂方法（Factory）、抽象工厂（Abstract Factory）、建造者（Builder）模式。但在这一天老师上课的时候讲了项目时提到需要四个设计模式都需要用到，所以在这基础上更新完成了V2版本，所以V1版本简陋带过（写都写了），主要分析在V2版本的说明中，在V2版本上发现可以优化为V3。

2.1 1. Builder模式

```
1 class JSONVisualizerBuilder:
2     def __init__(self, json_file):
3         with open(json_file, 'r') as f:
4             self.json_data = json.load(f)
5         self.style_factory = None
6         self.icon_factory = None
7
8     def with_style_and_icon_factory(self, style_factory, icon_factory):
9         self.style_factory = style_factory
10        self.icon_factory = icon_factory
11        return self
12
13    def build(self):
14        if not self.style_factory or not self.icon_factory:
15            raise ValueError("Both style factory and icon factory must be set.")
16        visualizer = self.style_factory.create_visualizer(self.json_data, self.icon_factory)
17        visualizer.display()
```

Builder模式用于创建复杂对象的构建过程。`JSONVisualizerBuilder`类用来逐步配置和创建一个JSON可视化器。它允许客户端通过链式调用`with_style_and_icon_factory`方法设置样式工厂和图标工厂，然后调用`build`方法完成对象的构建并显示。

2.2 2. 抽象工厂模式

```
1 class StyleFactory(ABC):
2     @abstractmethod
3     def create_visualizer(self, json_data, icon_factory):
4         pass
5
6 class TreeStyleFactory(StyleFactory):
7     def create_visualizer(self, json_data, icon_factory):
8         return TreeStyle(json_data, icon_factory)
9
10 class LinearStyleFactory(StyleFactory):
11     def create_visualizer(self, json_data, icon_factory):
12         return LinearStyle(json_data, icon_factory)
13
14 class TreeStyleFactory2(StyleFactory):
15     def create_visualizer(self, json_data, icon_factory):
16         return TreeStyle2(json_data, icon_factory)
17
18 class RectangleFactory(StyleFactory):
19     def create_visualizer(self, json_data, icon_factory):
20         return RectangleStyle(json_data, icon_factory)
```

抽象工厂模式用于创建一系列相关或依赖的对象，而无需指定它们的具体类。这里，`StyleFactory`是一个抽象工厂，定义了创建JSON可视化器的方法。具体工厂类如`TreeStyleFactory`、`LinearStyleFactory`等实现了这个接口，创建具体的JSON可视化器。

2.3 3. 工厂方法模式

```
1 class TreeStyleFactory(StyleFactory):
2     def create_visualizer(self, json_data, icon_factory):
3         return TreeStyle(json_data, icon_factory)
```

工厂方法模式定义了一个用于创建对象的接口，但由子类决定实例化哪一个类。每个具体工厂类都实现了`create_visualizer`方法，用于实例化具体的可视化器（如`TreeStyle`、`LinearStyle`等）。

3.1 1. 工厂方法模式

工厂方法模式通过定义一个创建对象的接口，让子类决定实例化哪一个类。其目的是让类的实例化延迟到子类中进行。

在代码中，`IconFactory` 是一个抽象类，定义了创建图标的方法，而具体的工厂类如 `PokerFaceIconFactory`、`FlowerIconFactory`、`NullIconFactory` 以及 `myIconFactory` 实现了这些方法，创建了具体的图标实例。

作用：

- 解耦了对象的创建和使用。
- 提高了代码的可扩展性，新增图标类型时只需添加新的具体工厂类即可。

3.2 2. 抽象工厂模式

抽象工厂模式提供一个创建一系列相关或依赖对象的接口，而无需指定它们具体的类。

在代码中，`IconFactory` 作为抽象工厂，而 `PokerFaceIconFactory`、`FlowerIconFactory`、`NullIconFactory` 和 `myIconFactory` 作为具体工厂，创建了不同类型的图标。

作用：

- 提供了一种封装一系列具有相互依赖关系对象的方式。
- 易于交换具体的工厂类。

3.3 3. 建造者模式

建造者模式将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

在代码中，`FunnyJsonExplorerBuilder` 是建造者类，通过一系列的设置方法配置需要构建的对象，并通过 `build` 方法构建最终的对象结构。

- `build` 方法递归地构建容器和叶子节点，这是建造者模式中的建造过程。
- `show` 方法调用多态的 `display` 方法来展示构建的结果。

作用：

- 允许逐步构建复杂对象。
- 提高了代码的可读性和可维护性。

3.4 4. 组合模式

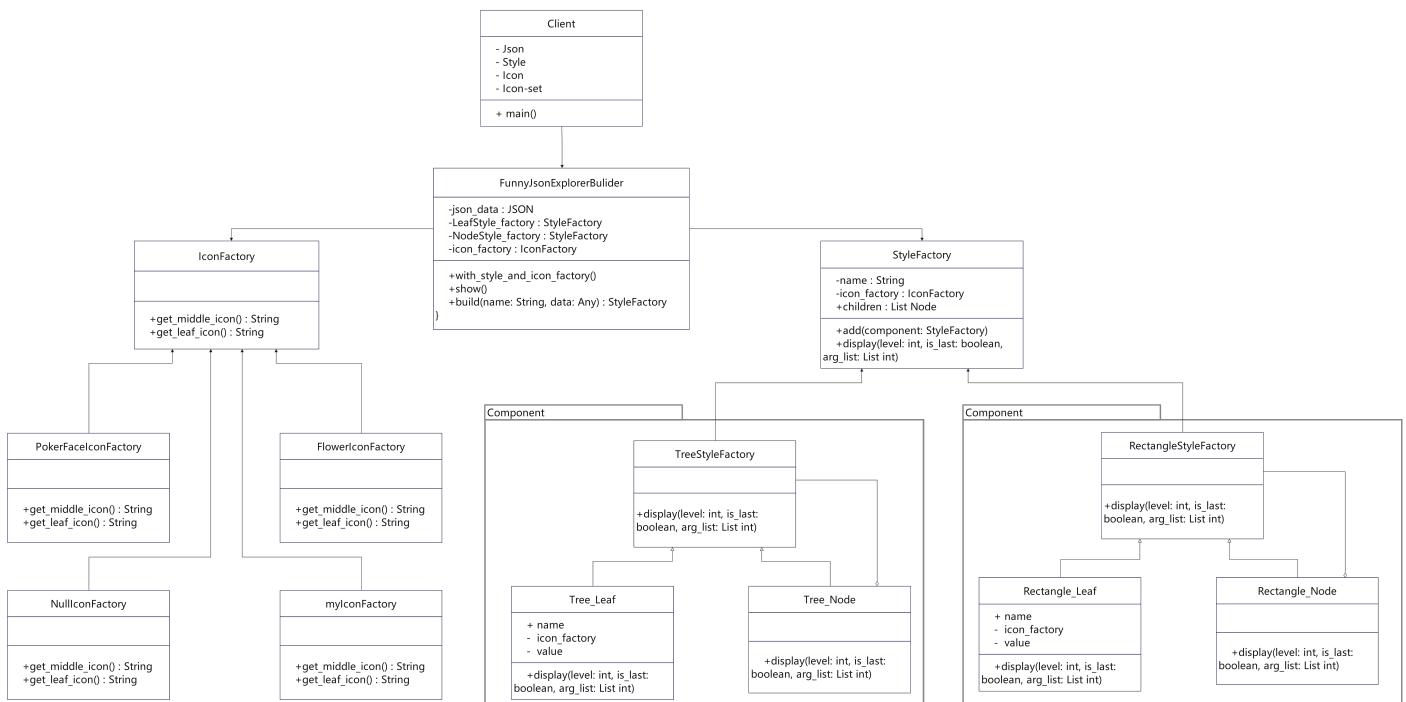
组合模式允许将对象组合成树形结构来表示“部分-整体”的层次结构。组合模式使得用户对单个对象和组合对象的使用具有一致性。

在代码中，`StyleFactory` 及其子类 `TreeFactory`、`Tree_Leaf`、`Tree_Node`、`RectangleFactory`、`Rectangle_Leaf` 和 `Rectangle_Node` 实现了组合模式。树节点和叶节点都继承了 `StyleFactory`，在 `build` 中可以递归地包含子节点并进行绘制。

作用：

- 使得用户可以统一地使用单个对象和组合对象。
- 提供了树形结构的灵活性和可扩展性。

3.5 类图



3.5.1 解释

1. IconFactory

- 抽象类，定义了获取中间图标和叶子图标的方法。

2. 具体工厂类（`PokerFaceIconFactory`, `FlowerIconFactory`, `NullIconFactory`, `myIconFactory`）

- 实现 `IconFactory` 的接口，提供具体的图标实现。

3. StyleFactory

- 抽象类，定义了添加子组件和绘制的方法。包含子组件的列表，表示组合模式。

4. 具体风格类（`Tree_Leaf`, `Tree_Node`, `Rectangle_Leaf`, `Rectangle_Node`）

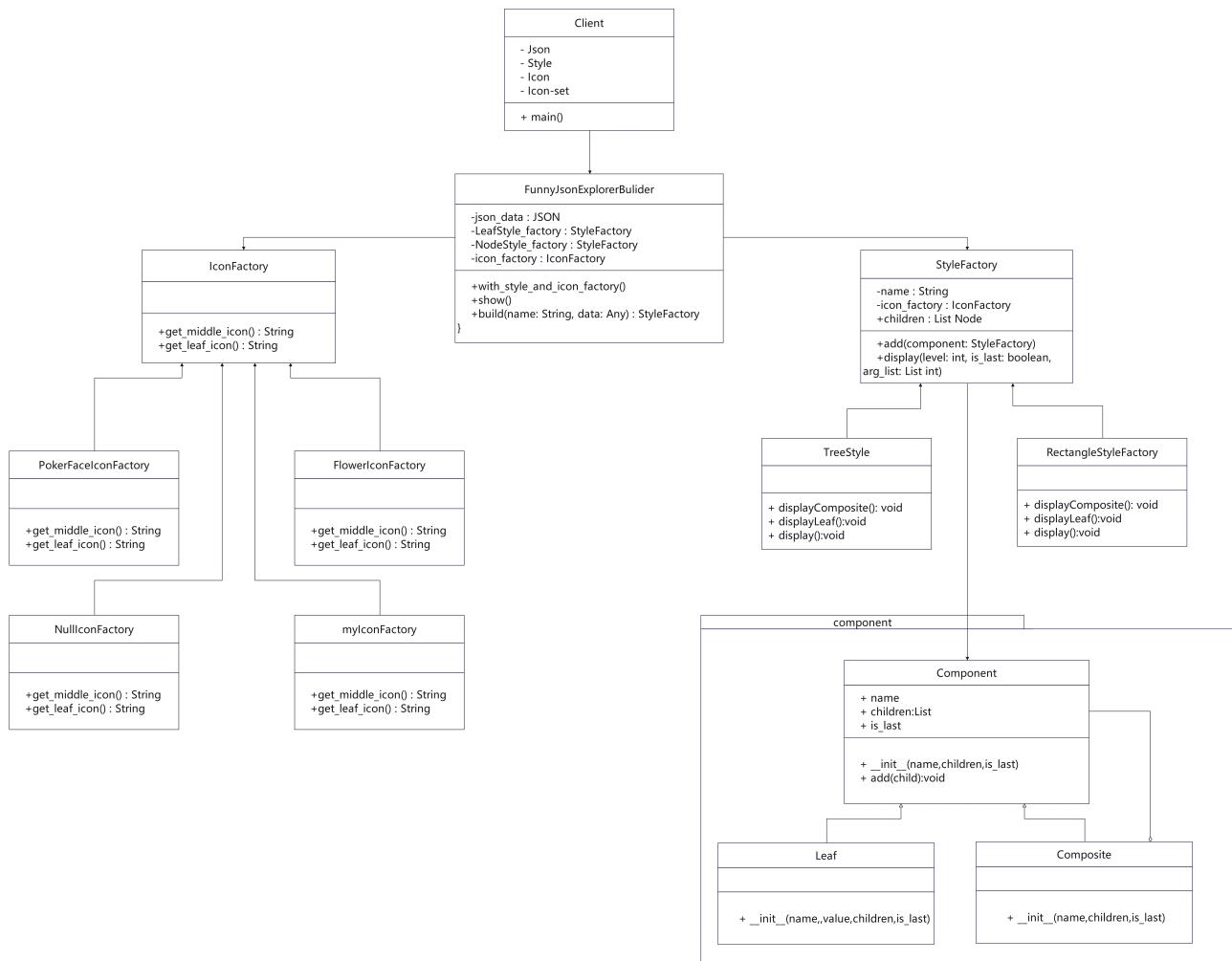
- 继承自 `StyleFactory`，实现具体的绘制逻辑。`Tree_Leaf` 和 `Tree_Node` 处理树形结构的绘制，`Rectangle_Leaf` 和 `Rectangle_Node` 处理矩形结构的绘制。

5. FunnyJsonExplorerBulider

- 建造者模式的具体实现类，负责解析JSON文件并构建对应的风格结构（树形或矩形）。依赖于 `StyleFactory` 和 `IconFactory` 来实现具体的构建逻辑。

4 V3

类图：



4.1 组合模式

组合模式允许你将对象组合成树形结构来表示“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。

4.1.1 组合模式关键点

1. **Component**（抽象组件）：定义了叶子和容器的共同接口，通常包括对管理子组件的接口。
2. **Leaf**（叶子节点）：表示组合的基本对象，叶子节点没有子节点。
3. **Composite**（容器节点）：定义有子组件的那些组件的行为。容器节点存储子组件，并且实现与子组件相关的操作。

4.1.2 组合模式代码说明

```
1 # 组合模式: 定义抽象组件
2 class Component:
3     def __init__(self, name, children=None, is_last=0):
4         self.name = name
5         self.children = children if children is not None else []
6         self.is_last = is_last
7
8     def add(self, child):
9         self.children.append(child)
10
11
12 # 定义叶子节点组件, 继承自 Component
13 class Leaf(Component):
14     def __init__(self, name, value, is_last=0):
15         super().__init__(name, children=[], is_last=is_last)
16         self.value = value
17
18 # 定义非叶子节点组件, 继承自 Component
19 class Composite(Component):
20     def __init__(self, name, is_last=0):
21         super().__init__(name, children=[], is_last=is_last)
```

其中 `Component` 是抽象基类，`Leaf` 是叶子节点，`Container` 是容器节点。

4.2 抽象工厂方法

工厂方法模式通过定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法使一个类的实例化延迟到其子类。

4.2.1 工厂方法模式代码说明

```
1 from abc import ABC, abstractmethod
2 # 工厂方法: 定义风格类
3 class StyleFactory(ABC):
4     def __init__(self, icon_factory):
5         self.icon_factory = icon_factory
6
7     @abstractmethod
8     def displayComposite(self, node, level, is_last, arg_list):
9         pass
10
11    @abstractmethod
12    def displayLeaf(self, node, level, is_last, arg_list):
```

```
13     pass
14
15     def display(self, node, level=0, is_last=False, arg_list=[]):
16         pass
```

```
1 from StyleFactory import StyleFactory
2
3 class TreeStyle(StyleFactory):
4     def displayComposite(self, node, level, is_last, arg_list):
5         icon = self.icon_factory.get_middle_icon()
6         if not is_last and level > 0 and level not in arg_list:
7             arg_list.append(level)
8         if is_last and level in arg_list:
9             arg_list.remove(level)
10        if level > 0:
11            prefix = ''
12            for i in range(1, level):
13                if i in arg_list:
14                    prefix += '| '
15                else:
16                    prefix += ' '
17            prefix += '└ ' if is_last else '├ '
18            line = prefix + icon + ' ' + node.name
19            print(line)
20
21     def displayLeaf(self, node, level, is_last, arg_list):
22         icon = self.icon_factory.get_leaf_icon()
23         prefix = ''
24         for i in range(1, level):
25             if i in arg_list:
26                 prefix += '| '
27             else:
28                 prefix += ' '
29         prefix += '└ ' if is_last else '├ '
30         if node.value is None:
31             line = prefix + icon + ' ' + node.name + ' '
32         else:
33             line = prefix + icon + ' ' + node.name + ': ' + str(node.value) + ' '
34         print(line)
35
36     def display(self, node, level=0, is_last=False, arg_list=[]):
37         if node.children:
38             self.displayComposite(node, level, is_last, arg_list)
39             for i, child in enumerate(node.children):
40                 child_is_last = i == len(node.children) - 1
41                 # is_top = i == 0
42                 # arg_list.insert(0, is_top)
43                 self.display(child, level + 1, child_is_last, arg_list)
44         else:
45             self.displayLeaf(node, level, is_last, arg_list)
```

4.3 建造者模式作业

建造者模式将一个复杂对象的构建与其表示分离，使得同样的构建过程可以创建不同的表示。

4.3.1 建造者模式代码说明

```
1 class FunnyJsonExplorerBuilder:
2     def __init__(self, json_file):
3         with open(json_file, 'r') as f:
4             self.json_data = json.load(f)
5             self.LeafStyle_factory = None
6             self.NodeStyle_factory = None
7             self.icon_factory = None
8
9     def with_style_and_icon_factory(self, LeafStyle_factory, NodeStyle_factory, icon_factory):
10        self.LeafStyle_factory = LeafStyle_factory
11        self.NodeStyle_factory = NodeStyle_factory
12        self.icon_factory = icon_factory
13        return self
14
15    def show(self):
16        root = self.build('root', self.json_data)
17        root.display()
18
19    def build(self, name, data):
20        if isinstance(data, dict):
21            container = self.NodeStyle_factory(name, self.icon_factory)
22            for key, value in data.items():
23                child = self.build(key, value)
24                container.add(child)
25            return container
26        elif isinstance(data, list):
27            container = self.NodeStyle_factory(name, self.icon_factory)
28            for index, item in enumerate(data):
29                child = self.build(str(index), item)
30                container.add(child)
31            return container
32        else:
33            return self.LeafStyle_factory(name, data, self.icon_factory)
34
35 # 使用建造者模式
36 builder = FunnyJsonExplorerBuilder('example.json')
37 builder.with_style_and_icon_factory(Tree_Leaf, Tree_Node, PokerFaceIconFactory()).show()
```

其中 `FunnyJsonExplorerBuilder` 是具体建造者类，`LeafStyle_factory` 和 `NodeStyle_factory` 分别是叶子节点和容器节点的工厂，`icon_factory` 是图标工厂。

5 使用方法

如果要使用config文件配置icon，请将文件命名为icon_config.json，然后放置在与fje.exe或main.py同级目录下，参考格式

```
1 {  
2   "middle": "👑",  
3   "leaf": "▶",  
4   "array": "✖"  
5 }
```

需要注意的是在版本二（修改为使用组合模式）中取消了array的icon样式，所以之后命令中有关array的都要去掉，已经更新到help中

5.1 方法一：python文件

一般使用方法：

```
1 | python .\main.py -f example.json -s tree -i poker
```

查询帮助:

```
1 | python .\main.py --help
```

自定义icon

一、命令行输入

```
1 | python .\main.py -f example.json -s tree -i myicon --icon-set $ * +
```

三、用配置文件，配置文件默认同名

```
1 | python .\main.py -f example.json -s tree -i configfile
```

5.2 方法二：使用exe文件

使用pyinstaller打包: Pyinstaller -F main.py -n fje

如果是终端输入可能需要将file改为./file

```
fje -f <json file> -s <style> -i <icon family>
```

一般使用方法：

```
1 | fje -f example.json -s tree -i poker
```

查询帮助:

```
1 | fje --help
```

一、命令行

一、命令行输入

```
1 | fje -f example.json -s tree -i myicon --icon-set $ * +
```

三、用配置文件，配置文件默认同名

```
1 | fje -f example.json -s tree -i configfile
```

6 V1结果展示

6.1 查询帮助：

6.2 自定义icon:

```

PS C:\onedrive-lvjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree -i myicon --icon-set $ * +
your icon set is,middle:$,lead:*,array:+

$ oranges
  $ mandarin
    * clementine
    * tangerine: cheap & juicy!
$ apples
  * gala
  * pink lady

PS C:\onedrive-lvjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s rectangle -i myicon --icon-set $ * +
your icon set is,middle:$,lead:*,array:+

$ oranges
  $ mandarin
    * clementine
    * tangerine: cheap & juicy!
$ apples
  * gala
  * pink lady

PS C:\onedrive-lvjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree2 -i myicon --icon-set $ * +
your icon set is,middle:$,lead:*,array:+

$ oranges
  $ mandarin
    * clementine
    * tangerine: cheap & juicy!
$ apples
  * gala
  * pink lady

PS C:\onedrive-lvjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s linear -i myicon --icon-set $ * +
your icon set is,middle:$,lead:*,array:+

```

自定义icon展示四种风格

6.3 配置文件设置icon

```
|   pink lady
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree -i configfile
use config file to set icon
```

用配置文件设置icon

```
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
+ apples
  └ gala
  └ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s rectangle -i configfile
use config file to set icon
```

```
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
+ apples
  └ gala
  └ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree2 -i configfile
use config file to set icon
```

```
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
+ apples
  └ gala
  └ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s linear -i configfile
use config file to set icon
```

```
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
+ apples
  └ gala
  └ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code>
```

6.3.1 内置icon样式

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree2 -i poker
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
```

三种内置icon

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree2 -i flower
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
```

```
+ apples
  └ gala
  └ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code> fje -f example.json -s tree2 -i null
+ oranges
  └ mandarin
    └ clementine
      └ tangerine: cheap & juicy!
```

```
+ apples
  └ gala
  └ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\code>
```

6.4 完整性测试

以下json文件基本包含所有语法:

```
1  {
2      "name": "Bob",
3      "age": 30,
4      "isStudent": false,
5      "contact": {
6          "email": "zhangsan@example.com",
7          "phone": "+1234567890"
8      },
9      "education": [
10         {
11             "degree": "Bachelor",
12             "major": "Computer Science",
13             "year": 2015,
14             "university": "Example University"
15         },
16         {
17             "degree": "Master",
18             "major": "Data Science",
19             "year": 2018,
20             "university": "Another Example University"
21         }
22     ],
23     "hobbies": ["reading", "cycling", "coding"],
24     "address": {
25         "street": "123 Elm Street",
26         "city": "Springfield",
27         "state": "Illinois",
28         "zip": "62704"
29     }
30 }
31 }
```

```
PS C:\onedrive\lvjw\OneDrive - mail2.sysu.edu.cn\SVSU\Hostsite_G1_DOMAIN\SoftwareEngineering\Json\code> fje -f strength.json -s tree -l poker
{
    "name": "Bob",
    "age": 30,
    "isStudent": false,
    "contact": {
        "email": "zhangsan@example.com",
        "phone": "+1234567890"
    },
    "education": [
        {
            "degree": "Bachelor",
            "major": "Computer Science",
            "year": 2015,
            "university": "Example University"
        },
        {
            "degree": "Master",
            "major": "Data Science",
            "year": 2018,
            "university": "Another Example University"
        }
    ],
    "hobbies": [
        "reading",
        "cycling",
        "coding"
    ],
    "address": {
        "street": "123 Elm Street",
        "city": "Springfield",
        "state": "Illinois",
        "zip": "62704"
    }
}
```

(a)

```
PS C:\onedrive\lvjw\OneDrive - mail2.sysu.edu.cn\SVSU\Hostsite_G1_DOMAIN\SoftwareEngineering\Json\code> fje -f strength.json -s linear -l poker
{
    "name": "Bob",
    "age": 30,
    "isStudent": false,
    "contact": {
        "email": "zhangsan@example.com",
        "phone": "+1234567890"
    },
    "education": [
        {
            "degree": "Bachelor",
            "major": "Computer Science",
            "year": 2015,
            "university": "Example University"
        },
        {
            "degree": "Master",
            "major": "Data Science",
            "year": 2018,
            "university": "Another Example University"
        }
    ],
    "hobbies": [
        "reading",
        "cycling",
        "coding"
    ],
    "address": {
        "street": "123 Elm Street",
        "city": "Springfield",
        "state": "Illinois",
        "zip": "62704"
    }
}
```

(b)

7 V2结果展示

V3同V2

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\reference> fje -f example.json -s tree -i myicon --icon-set $ *
```

```
$ oranges
  $ mandarin
    * clementine
      * tangerine: cheap & juicy!
  $ apples
    * gala
```

配置文件

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\reference> fje -f example.json -s tree -i configfile
use config file to set icon
```

```
$ oranges
  $ mandarin
    ▶ clementine
    ▶ tangerine: cheap & juicy!
  ▶ gala
  ▶ pink lady
```

风格

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\reference> fje -f example.json -s tree -i poker
```

```
@ oranges
  @ mandarin
    @ clementine
      @ tangerine: cheap & juicy!
  @ apples
    @ gala
    @ pink lady
```

icon

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\reference> fje -f example.json -s rectangle -i poker
```

```
◊ oranges
  ◊ mandarin
    ◊ clementine
    ◊ tangerine: cheap & juicy!
  ◊ apples
    ◊ gala
    ◊ pink lady
```

```
PS C:\onedrive-lwjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\reference> fje -f example.json -s rectangle -i flower
```

```
@ oranges
  @ mandarin
    @ clementine
    @ tangerine: cheap & juicy!
  @ apples
    @ gala
    @ pink lady
```

7.1 完整性测试

```
PS C:\onedrive-lvjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\V2> python .\main.py -f strength.json -s tree -i configfile
```

● use config file to set icon

```
▶ name: Bob
▶ age: 30
▶ isStudent: False
▶ contact
└▶ email: zhangsan@example.com
   └▶ phone: +1234567890
▶ education
└▶ 0
  └▶ degree: Bachelor
     └▶ major: Computer Science
     └▶ year: 2015
     └▶ university: Example University
   └▶ 1
     └▶ degree: Master
       └▶ major: Data Science
       └▶ year: 2018
       └▶ university: Another Example University
▶ hobbies
└▶ 0: reading
   └▶ 1: cycling
     └▶ 2: coding
▶ address
└▶ street: 123 Elm Street
   └▶ city: Springfield
     └▶ state: Illinois
   └▶ zip: 62704
```

```
PS C:\onedrive-lvjw7\OneDrive - mail2.sysu.edu.cn\SYSU\MostUse_G3_DOWN\SoftwareEngineering\Json\V2> python .\main.py -f strength.json -s rectangle -i configfile
```

● use config file to set icon

```
▶ name: Bob
▶ age: 30
▶ isStudent: False
▶ contact
└▶ email: zhangsan@example.com
   └▶ phone: +1234567890
▶ education
└▶ 0
  └▶ degree: Bachelor
     └▶ major: Computer Science
     └▶ year: 2015
     └▶ university: Example University
   └▶ 1
     └▶ degree: Master
       └▶ major: Data Science
       └▶ year: 2018
       └▶ university: Another Example University
▶ hobbies
└▶ 0: reading
   └▶ 1: cycling
     └▶ 2: coding
▶ address
└▶ street: 123 Elm Street
   └▶ city: Springfield
     └▶ state: Illinois
   └▶ zip: 62704
```