21307112 付芷怡

基于上一个实验进行了改进, 对已有的 FJE 实现进行了设计重构, 改用了迭代器+策略模式。

新增的类说明如下:

Strategy：

execute( ):抽象方法，运行策略

set_container( ):抽象方法，用于设置具体的容器类

set_leaf( ):抽象方法，用于设置具体的叶子节点类

TreeStrategy 和 RectangleStrategy 是继承于 Strategy 的具体类

Context:

set_strategy( ):用于设置具体策略

execute_strategy( ):用于运行具体策略

Iterator:

get_next( ):抽象方法，用于获取迭代器下一个目标

more( ):抽象方法，用于看是否为最后一个迭代目标

IteratorCollection:

create_iterator( ):抽象方法，用于创建迭代器

JSONIteratorCollection:

create_iterator( ):具体方法，用于创建相应迭代器

其余类复用上一个实验当中的实现，这里就不再赘述

策略模式：

策略模式是一种行为设计模式，它允许在运行时选择算法的行为。策略模式定义了一系列算法，将每个算法封装起来，并使它们可以相互替换，而不影响客户端的使用。

策略模式的组成角色如下：首先是一个策略接口，定义一个公共接口或抽象类，规定了具体策略类必须实现的方法。这个接口通常会包含一个或多个方法，用于定义算法的不同变体；其次就是具体策略类，实现了策略接口的具体类。每个具体策略类实现了一种算法或行为。在策略模式中，这些具体策略类通常是相互独立的，可以根据具体情况选择使用其中的一种；最后是环境类，维护一个对策略对象的引用，并在运行时切换不同的策略。这个环境类负责与客户端交互，并在运行时根据需求选择合适的策略。通常，环境类会在其内部持有一个策略接口的引用，或者允许客户端动态地传递不同的策略对象。在运行时，根据外部条件选择合适的策略，通过环境类的 `set_strategy` 方法设置策略。环境类的 `execute` 方法负责调用策略对象的相应方法，执行策略定义的算法，并返回结果。

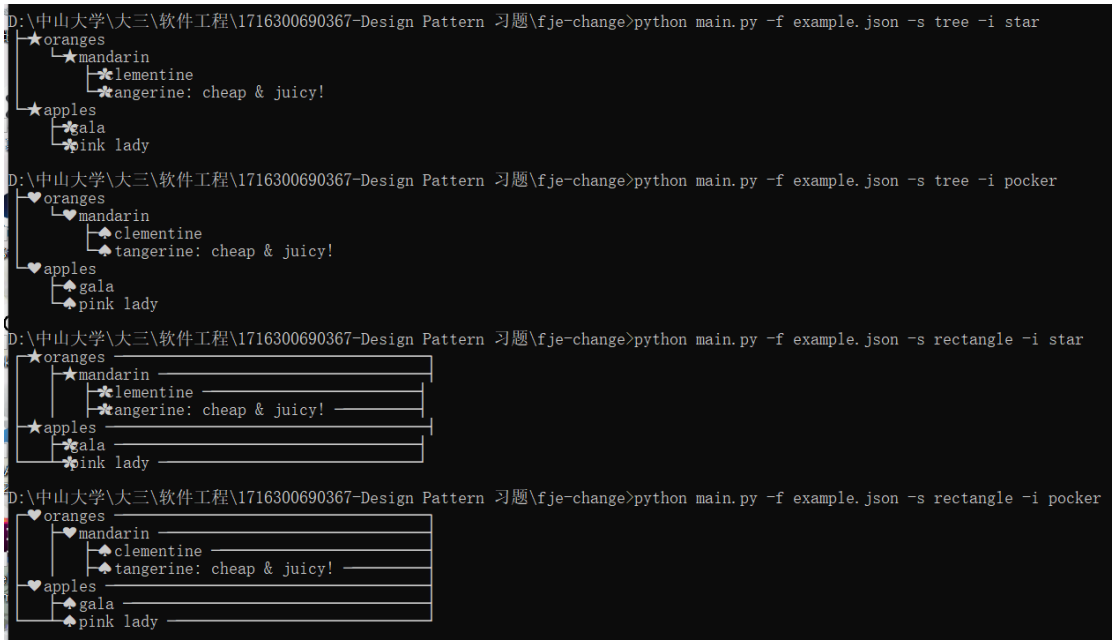
迭代器模式：

迭代器模式是一种行为设计模式，它允许客户端按顺序访问集合对象的元素，而无需了解底层数据结构。迭代器模式将遍历算法与集合分离，使得可以单独改变集合类或者遍历方式，而不会影响彼此。

迭代器模式的组成角色如下：首先是迭代器接口，定义访问和遍历元素的方法，如获取下一个元素、检查是否还有元素等。这个接口为具体迭代器类定义了一组通用的操作；其次是具体迭代器类，实现迭代器接口，并负责实现具体的遍历算法。每个具体迭代器类都维护了对特定集合的引用，并跟踪当前遍历的位置；然后是集合接口，定义一个或多个方法，用于创建相应的迭代器对象。这个接口可能包含用于获取迭代器的方法；最后是具体集合类，实现集合接口，负责创建具体的迭代器对象。这些具体集合类实现了在迭代器模式中定义的集合的方法，以及与具体数据结构相关的其他方法。

其他设计模式与上一次实验一致，此处不再赘述。

运行截图如下：



源代码库：https://github.com/chuanyunbaihe/Funny-JSON-Explorer-Change.git