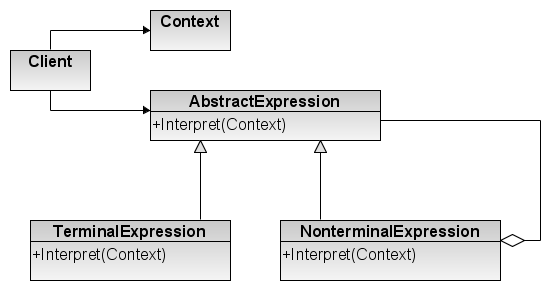
**ИНТЕРПРЕТАТОР** - для заданного языка определяет представление его грамматики, а также интерпретатор предложений этого языка.

ПРИМЕНЕНИЕ:

- когда некоторая задача возникает часто, то имеет смысл представить ее конкретные проявления в виде предложений на простом языке. Затем можно будет создать интерпретатор, который решает задачу, анализируя предложения этого языка

- когда есть язык для интерпретации, предложения которого можно представить в виде абстрактных синтаксических деревьев

****РЕАЛИЗАЦИЯ:

**1. АБСТРАКТНОЕ ВЫРАЖЕНИЕ**: объявляет абстрактную операцию Interpret, общую для всех узлов в абстрактном синтаксическом дереве

**2. ТЕРМИНАЛЬНОЕ ВЫРАЖЕНИЕ**: реализует операцию Interpret для терминальных символов грамматики. Необходим отдельный экземпляр для каждого терминального символа в предложении

**3. НЕТЕРМИНАЛЬНОЕ ВЫРАЖЕНИЕ**: по одному такому классу требуется для каждого грамматического правила. Хранит переменные экземпляра типа AbstractExpression для каждого символа. Реализует операцию Interpret для нетерминальных символов грамматики. Эта операция рекурсивно вызывает себя же для переменных

**4. КОНТЕКСТ**: содержит информацию, глобальную по отношению к интерпретатору

**5. КЛИЕНТ**: строит (или получает в готовом виде) абстрактное синтаксическое дерево, представляющее отдельное предложение на языке с данной грамматикой. Дерево составлено из экземпляров классов Nonterminal-Expression и Terminal-Expression. Вызывает операцию Interpret

ПЛЮСЫ:

- представление правил грамматики в виде классов упрощает реализацию языка

- можно легко изменять и расширять язык

- включение дополнительных методов в структуру классов позволяет добавлять новое поведение, не связанное с интерпретацией (например, форматированный вывод или проверка корректности интерпретируемого кода)

МИНУСЫ:

- при большом количестве правил реализация становится слишком громоздкой. Тогда лучше воспользоваться парсером/компилятором.