

The background features a vertical gradient from light purple at the top to light blue at the bottom. Scattered throughout are several realistic water droplets of various sizes, some with highlights and shadows, giving them a three-dimensional appearance.

ПОСЕТИТЕЛЬ

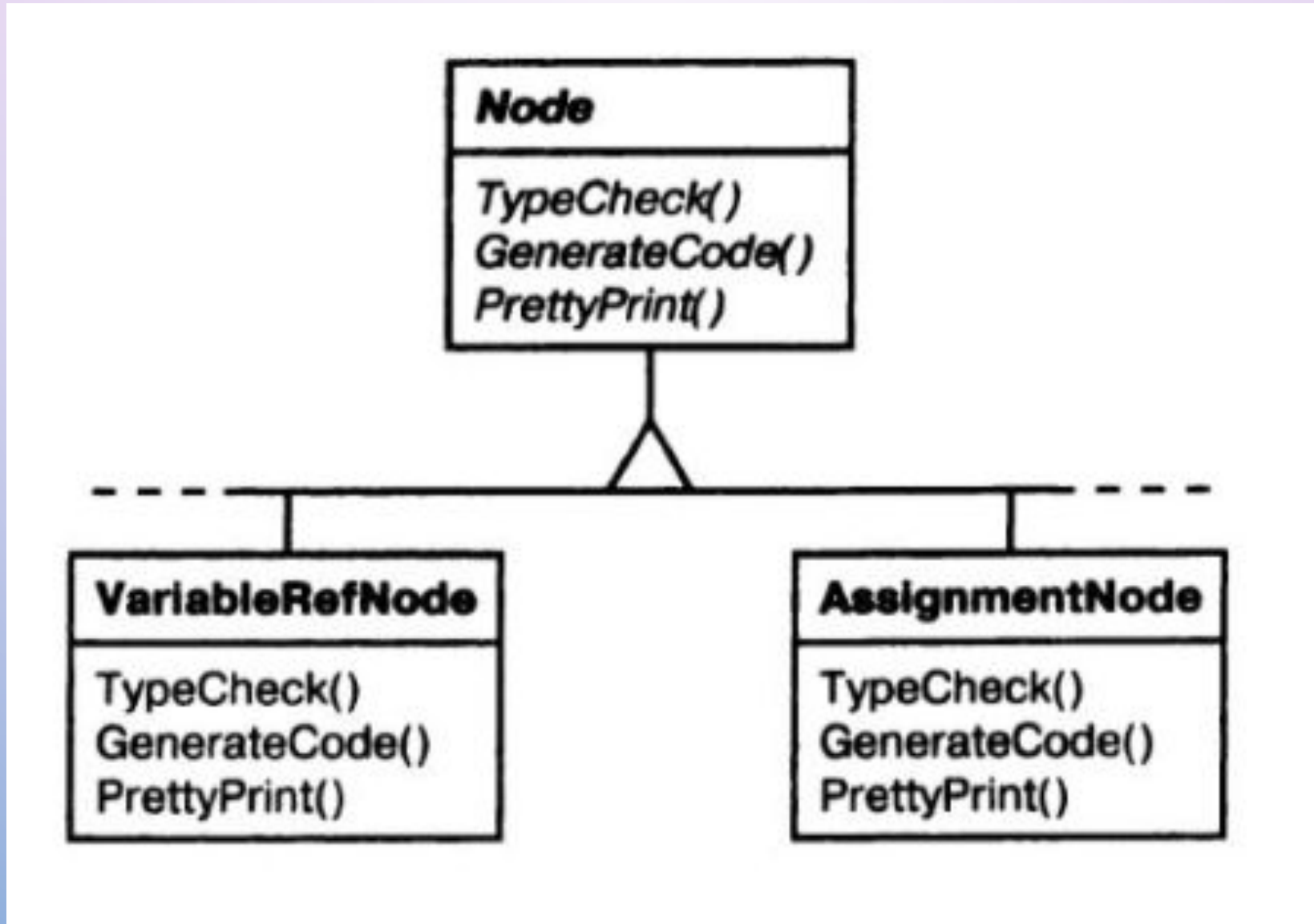
ПЛОТНИКОВ АЛЕКСАНДР, 271 ГРУППА, 2017

ПРИМЕРЧИК

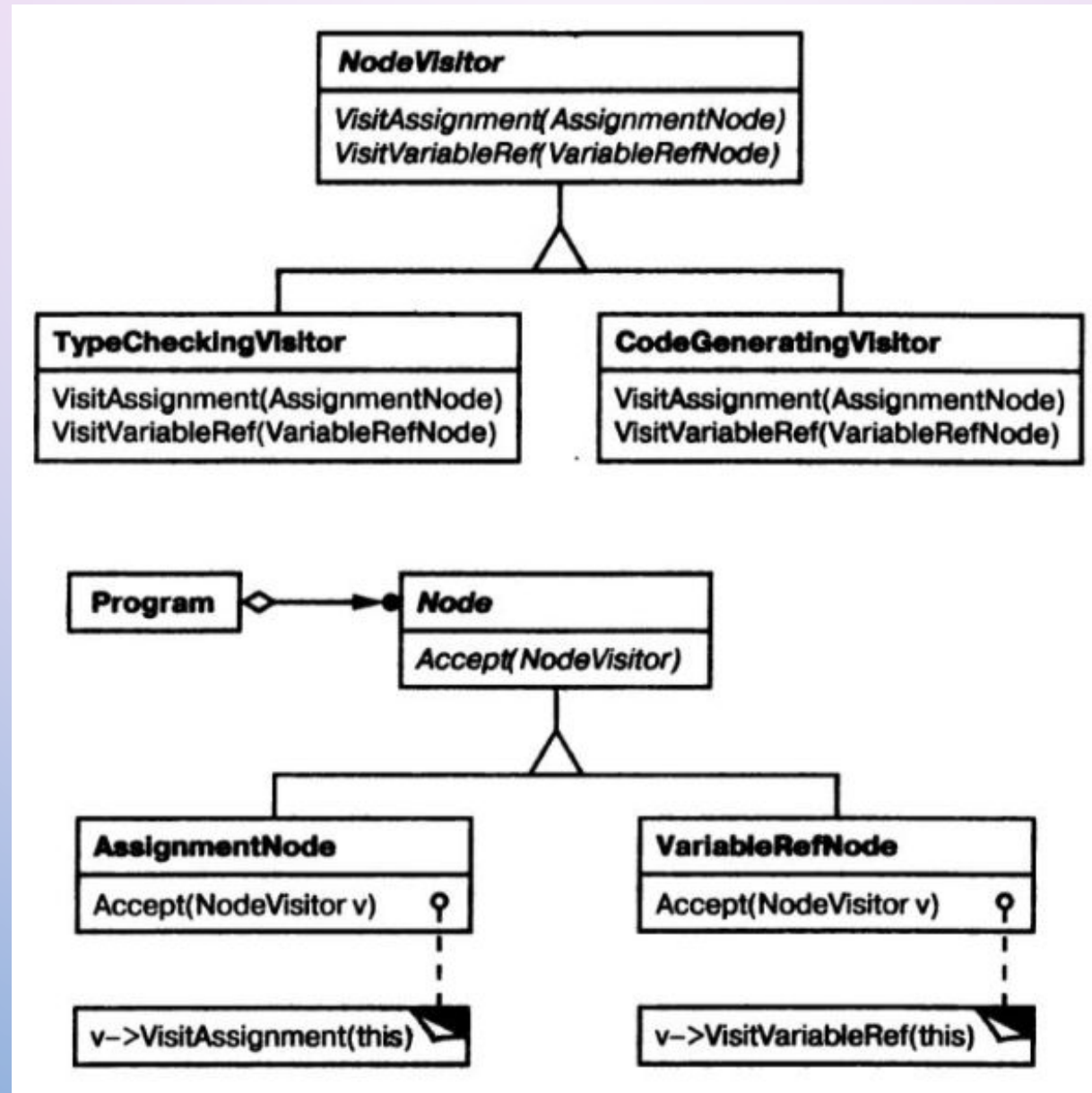
КОМПИЛЯТОР, ПРЕДСТАВЛЯЮЩИЙ ПРОГРАММУ
В ВИДЕ СИНТАКСИЧЕСКОГО ДЕРЕВА.

- ГЕНЕРАЦИЯ КОДА
- КОНТРОЛЬ ТИПОВ
- КРАСИВЫЙ ВЫВОД
- И Т.Д.

ЕСЛИ ПРОСТО ДЕЛАТЬ В ЛОБ



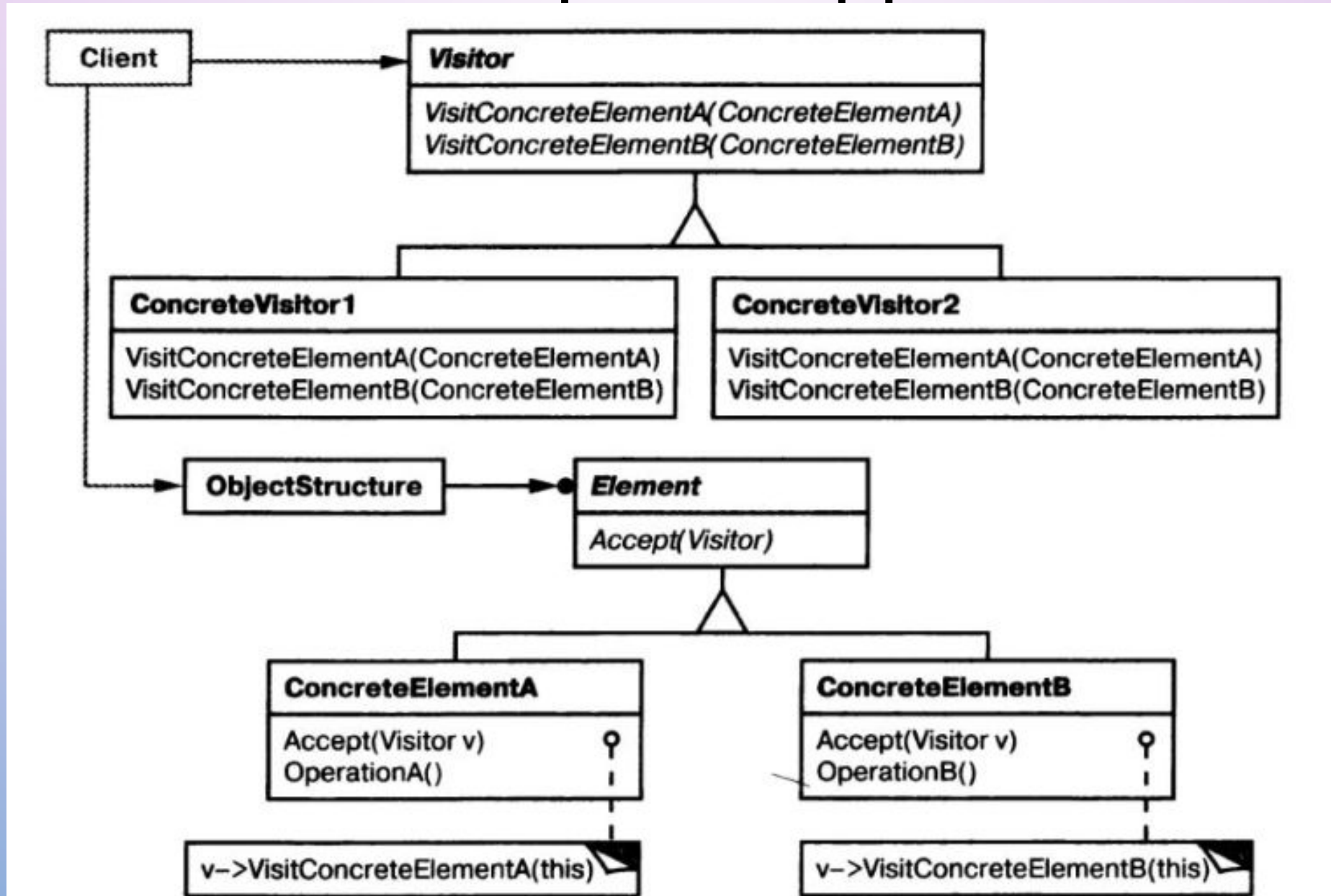
GOOD PRACTICE



ПРИМЕНИМОСТЬ

- СЛАБАЯ СВЯЗЬ МЕЖДУ ОПЕРАЦИЯМИ
- СТРУКТУРА ОБЪЕКТОВ ИЗМЕНЯЕТСЯ РЕЖЕ ЧЕМ ОПЕРАЦИИ НАД НИМИ

В ОБЩЕМ ВИДЕ



РЕЗУЛЬТАТЫ

- УПРОЩАЕТ ДОБАВЛЕНИЕ НОВЫХ ОПЕРАЦИЙ
- ОБЪЕДИНЯЕТ РОДСТВЕННЫЕ ОПЕРАЦИИ
- ДОБАВЛЕНИЕ НОВЫХ ЭЛЕМЕНТОВ СТРУКТУРЫ ЗАТРУДНЕНО
- НАРУШЕНИЕ ИНКАПСУЛЯЦИИ

```
1 #include <iostream>
2 #include <vector>
3
4 // Предварительное объявление AbstractDispatcher
5 class AbstractDispatcher;
6 |
7 // Родительский класс для элементов (ArchivedFile, SplitFile and ExtractedFile)
8 class File {
9 public:
10     /*
11     * Метод принимающий объект класса отнаследованного от AbstractDispatcher и
12     * реализуемый во всех наследниках класса
13     */
14     virtual void accept(AbstractDispatcher &dispatcher) = 0;
15 };
16
17 // Предварительное объявление конкретных элементов (файлов)
18 class ArchivedFile;
19 class SplitFile;
20 class ExtractedFile;
21
22 class AbstractDispatcher {
23 public:
24     // Объявление перегрузки для каждого вида файла
25     virtual void dispatch(ArchivedFile &file) = 0;
26     virtual void dispatch(SplitFile &file) = 0;
27     virtual void dispatch(ExtractedFile &file) = 0;
28 };
29
30 class ArchivedFile: public File {
31 public:
32     // Перегрузка метода, соответствующего ArchivedFile
33     void accept(AbstractDispatcher &dispatcher) override {
34         dispatcher.dispatch(*this);
35     }
36 };
37
38 class SplitFile: public File {
39 public:
40     // Перегрузка метода, соответствующего SplitFile
41     void accept(AbstractDispatcher &dispatcher) override {
42         dispatcher.dispatch(*this);
43     }
44 };
45
46 class ExtractedFile: public File {
47 public:
48     // Перегрузка метода, соответствующего ExtractedFile
49     void accept(AbstractDispatcher &dispatcher) override {
50         dispatcher.dispatch(*this);
51     }
52 };
```



```
53
54 // Реализация dispatch для каждого вида элемента
55 class Dispatcher: public AbstractDispatcher {
56 public:
57     void dispatch(ArchivedFile &file) override {
58         std::cout << "dispatching ArchivedFile" << std::endl;
59     }
60
61     void dispatch(SplitFile &file) override {
62         std::cout << "dispatching SplitFile" << std::endl;
63     }
64
65     void dispatch(ExtractedFile &file) override {
66         std::cout << "dispatching ExtractedFile" << std::endl;
67     }
68 };
69
70 int main() {
71     ArchivedFile archivedFile;
72     SplitFile splitFile;
73     ExtractedFile extractedFile;
74
75     std::vector<File*> files;
76     files.push_back(&archivedFile);
77     files.push_back(&splitFile);
78     files.push_back(&extractedFile);
79
80     Dispatcher dispatcher;
81
82     for (File* file : files) {
83         file->accept(dispatcher);
84     }
85 }
```