ПОСЕТИТЕЛЬ

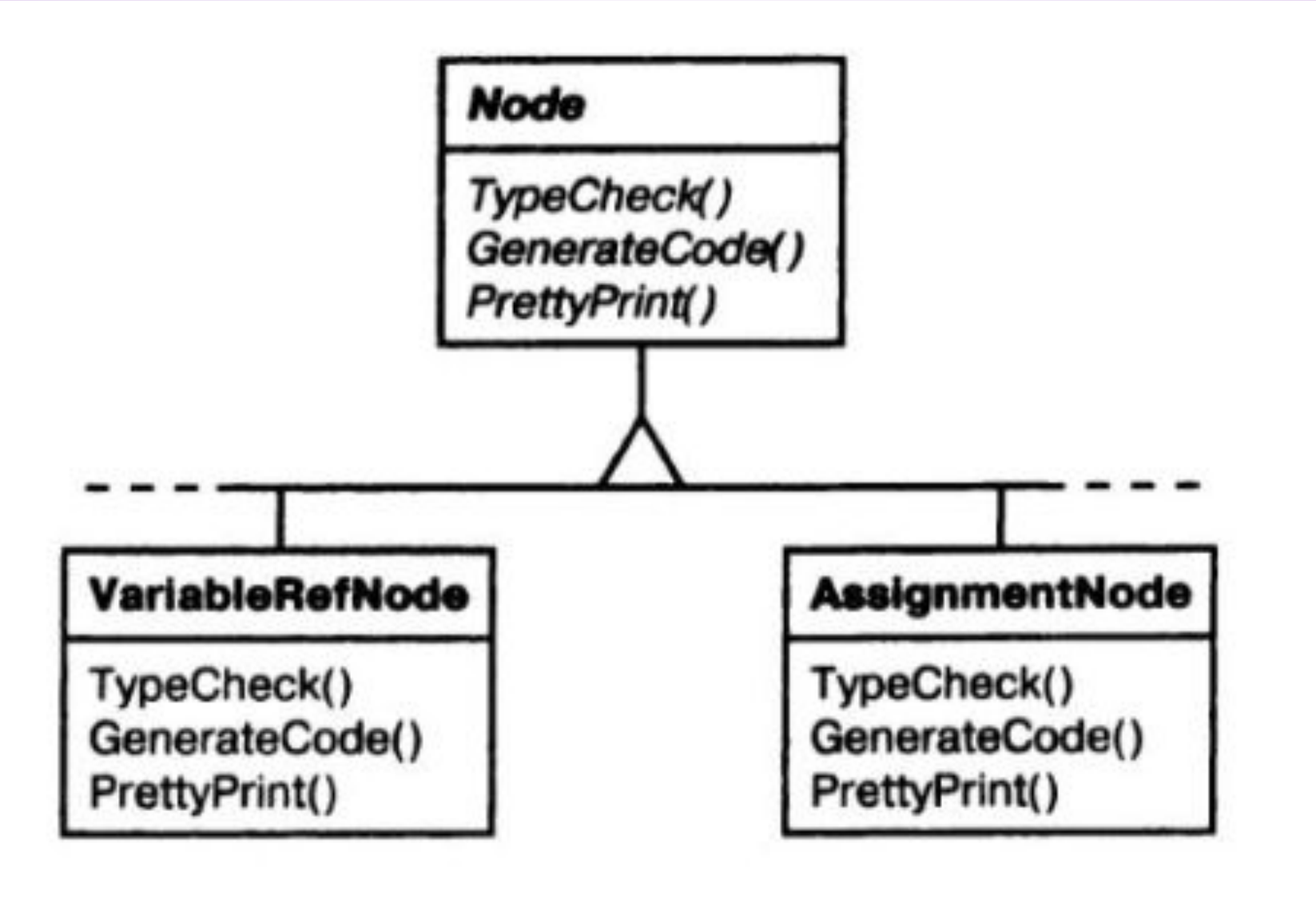
ПЛОТНИКОВ АЛЕКСАНДР, 271 ГРУППА, 2017

ПРИМЕРЧИК

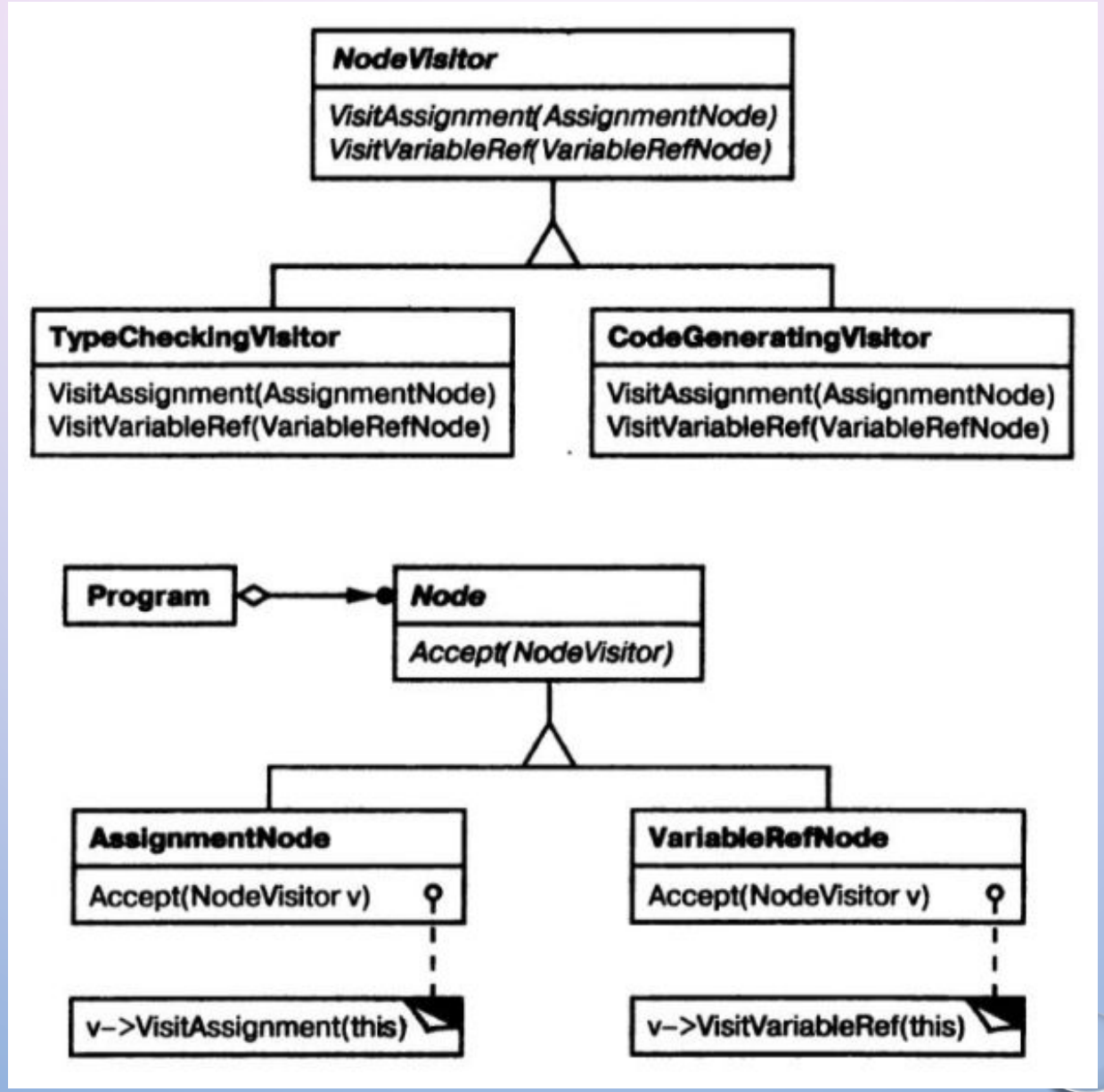
КОМПИЛЯТОР, ПРЕДСТАВЛЯЮЩИЙ ПРОГРАММУ В ВИДЕ СИНТАКСИЧЕСКОГО ДЕРЕВА.

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

ЕСЛИ ПРОСТО ДЕЛАТЬ В ЛОБ



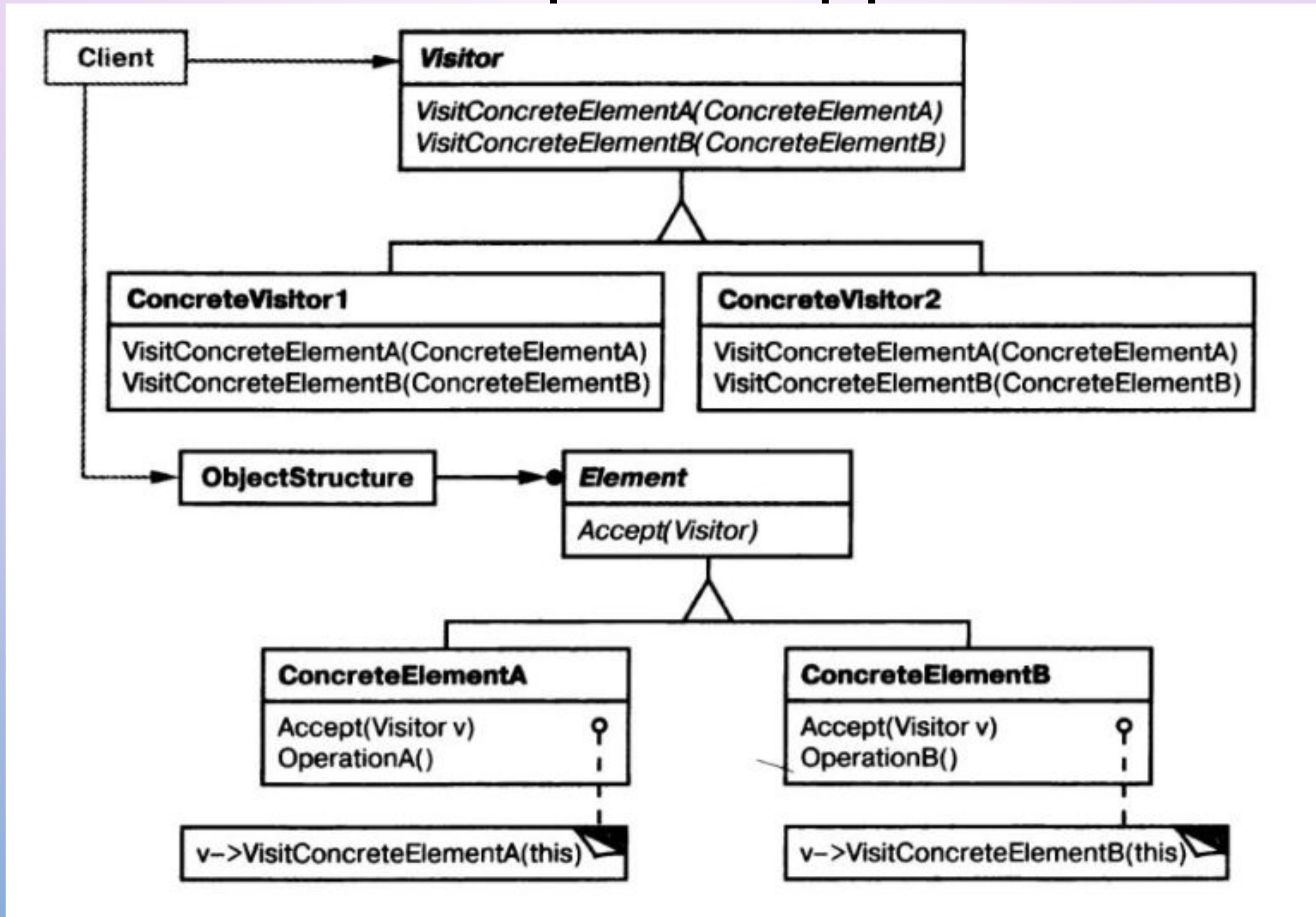
GOOD PRACTICE



ПРИМЕНИМОСТЬ

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

В ОБЩЕМ ВИДЕ


Client
Visitor
VisitConcreteElementA(ConcreteElementA)
VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor1
VisitConcreteElementA(ConcreteElementA)
VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor2
VisitConcreteElementA(ConcreteElementA)
VisitConcreteElementB(ConcreteElementB)
ObjectStructure
Element
Accept(Visitor)
ConcreteElementA
Accept(Visitor v)
OperationA()
ConcreteElementB
Accept(Visitor v)
OperationB()
v->VisitConcreteElementA(this)
v->VisitConcreteElementB(this)

РЕЗУЛЬТАТЫ

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

```cpp
#include <iostream>
#include <vector>

// Предварительное объявление AbstractDispatcher
class AbstractDispatcher;

// Родительский класс для элементов (ArchivedFile, SplitFile and ExtractedFile)
class File {
public:
    /*
     * Метод принимающий объект класса отнаследованного от AbstractDispatcher и
     * реализуемый во всех наследниках класса
     */
    virtual void accept(AbstractDispatcher &dispatcher) = 0;
};

// Предварительное объявление конкретных элементов (файлов)
class ArchivedFile;
class SplitFile;
class ExtractedFile;

class AbstractDispatcher {
public:
    // Объявление перегрузки для каждого вида файла
    virtual void dispatch(ArchivedFile &file) = 0;
    virtual void dispatch(SplitFile &file) = 0;
    virtual void dispatch(ExtractedFile &file) = 0;
};
```

```cpp

class ArchivedFile: public File {
public:
    // Перегрузка метода, соответствующего ArchivedFile
    void accept(AbstractDispatcher &dispatcher) override {
        dispatcher.dispatch(*this);
    }
};

class SplitFile: public File {
public:
    // Перегрузка метода, соответствующего SplitFile
    void accept(AbstractDispatcher &dispatcher) override {
        dispatcher.dispatch(*this);
    }
};

class ExtractedFile: public File {
public:
    // Перегрузка метода, соответствующего ExtractedFile
    void accept(AbstractDispatcher &dispatcher) override {
        dispatcher.dispatch(*this);
    }
};
```

```
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  }
```