**УДК 620.3.51**

**Э.Ю. Миннахметов**
Пермский национальный исследовательский
политехнический университет, г. Пермь

# ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПРИМЕРЕ КАЛЬКУЛЯТОРА ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ

Всякая программная система имеет две разные ценности: поведение и структуру. Разработчики отвечают за высокий уровень обеих. Но, к сожалению, они часто сосредотачиваются на чем-то одном, забывая про другое. Хуже того, они нередко сосредотачиваются на меньшей из двух ценностей, что в конечном итоге обесценивает систему. [1]

**Ключевые слова:** программа, программирование, архитектура, паттерн, шаблон, проектирование, чистый, исходный, код

**E.J. Minnakhmetov**
Perm National Research Politechnic University, Perm

# DESIGNING THE SOFTWARE ARCHITECTURE USING THE EXAMPLE OF A LOGICAL EXPRESSION CALCULATOR

Every software system has two different values: behavior and structure. Developers are responsible for keeping both of these values high. But, unfortunately, they often focus on one thing, forgetting about another. Worse, they often focus on the lesser of the two values, which ultimately devalues the system.

**Keywords:** program, programming, architecture, pattern, design, clean, source, code.

**Введение.** Накопленных знаний по проектированию программного обеспечения на сегодняшний день достаточно много, однако, они тяжелы в усвоении, поскольку качественных примеров и при этом простых не так много и их сложно найти.

Данная статья ориентирована на круг программистов, достигших уровня хорошего знания какого-либо объектно-

ориентированного языка программирования и принципов ООП, владеющих теоретическими заниями по проектированию монолитных приложений и по паттернам проектирования из книги «Банды четырёх». Целью же статьи является показать пример применения этих знаний, поэтому объяснения вышеперечисленных теорий в статье не будет.

Рассмотрим общепринятую модель разбиения архитектуры по слоям:


Рисунок 1 – Разбиение архитектуры по слоям

Наибольший интерес для статьи здесь несут слои «Данные» и «Бизнес-логика» в то время, как «Пользова-тельский интерфейс» лишь агрегирует ими [2], поэтому он рассмотрен не будет. Ценность для статьи несут модели и логика их взаимодействия, далее будут реализованным слои, ответственные за них.

**Технология реализации.** Возвращаясь к рис. 1, стоит описать выбранные слои:

1) В данных должны быть реализованны моледи и высокоуровневая логика - в булевом калькуляторе это будут переменные, операции над ними и функции, которые будут позволять строить выражения из нескольких переменных либо других функций и задавать тип операции;

2) Бизнес-логика, агрегируя моделями, должна определять поведение программы и предоставлять интерфейс к этому поведению – в проекте калькулятора эту будут формулировщики таблицы истинности, менеджеры смены типа операции и аргументов функций;

Были кратко описаны «Данные» и «Бизнес-логика», далее будет приведена реализация в виде диаграмм классов UML с описанием сущностей.

Ниже представленна разработанная диаграмма классов моделей «Данных»:

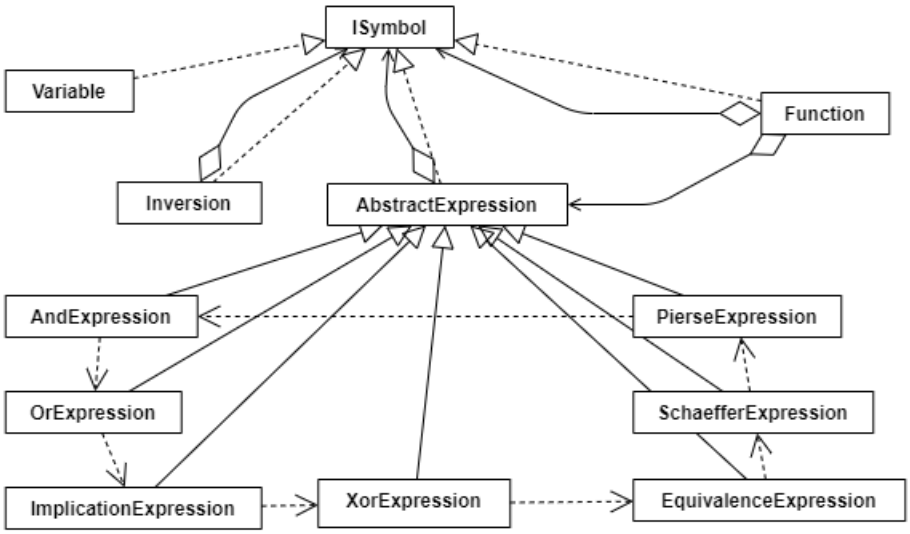


Рисунок 2 – Отношения внутри слоя «Данные»

На данной диаграмме показаны сущности:

1) ISymbol – интерфейс символа, который реализуют переменная, инверсия, абстрактная операция и функция, – предоставляет доступ к чтению значения;

2) Variable – класс переменных, который предоставляет читать и изменять имя переменной, ее значение;

3) Inversion – класс инверсий, агрегирует одним символом, возвращает значение инвертированное по отношению к значению символа. Является реализацией паттерна Декоратор [3];

4) AbstractExpression – абстрактный класс булевых операций, агрегирует двумя символами, значением является значение конкретного выражения над двумя символами. Является реализацией паттернов Интерпретатор, Стратегия и Шаблонный метод [3];

5) Конкретные операции (Логическое «И» – And-Expression, Логическое «ИЛИ» – OrExpression и т.д.) – наследники абстрактной операции, вычисляют значение по агрегируемым символам, предоставляют константу приоритета и предоставляют ссылку на следующее по приоритету выражение;

6) Function – класс функций, агрегирует одной операцией и двумя символами. Способен менять агрегируемую операцию, является реализацией паттерна Состояние. [3]

**Итого**, был описан модуль Данных, хранящий в себе модели и высокоуровневую логику, далее будет описана бизнес-логика, работающая с этими моделями.

Ниже представленна разработанная диаграмма классов «Бизнес-логики»:

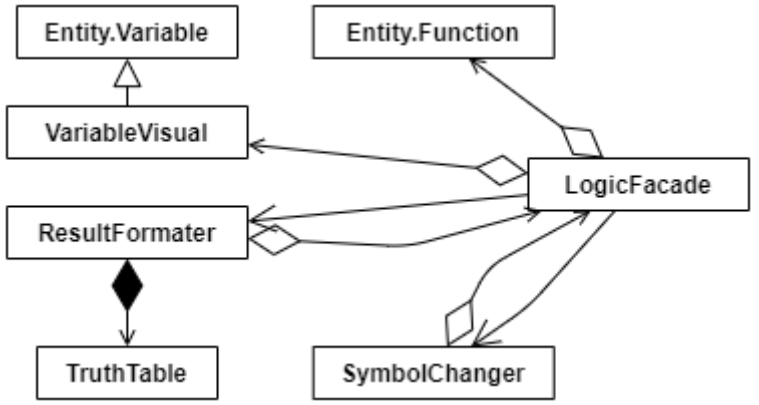


Рисунок 3 – Отношения внутри слоя «Бизнес-логика»

На данной диаграмме показаны отношения:

1) VariableVisual – класс, расширяющий класс переменных, посредством добавления возможности чтения значения получением символа «1» или «0», а не true и false;

2) TruthTable – класс, автоматизирующий создание таблицы истинности;

3) ResultFormater – класс, который создает и заполняет таблицу истинности, а затем формирует текстовый результат на ее основе и возвращает его; [4]

4) SymbolChanger – класс, занимающийся обновлением символов в активной функции; [4]

5) LogicFacade – скрывает логику ранее перечисленных классов, предоставляя удобный интерфейс для работы со всем модулем бизнес-логики. Агрегирует переменными и функциями отношением «один ко многим». Является реализацией паттерна Фасад. [3]

Подводя итоги, был описан модуль Бизнес-логики, агрегирующий моделями из модуля Данных и предоставляющий интерфейс для работы с ним. Показательный пример построения архитектуры калькулятора логических выражений на этом завершен.

**Заключение.** Обилие паттернов проектирования и методолгий построений архитектуры программного обеспечения затуманивает свою пользу отсутствием качественных примеров, поэтому целью данной работы было предоставить один из них.

Были реализованы паттерны Декоратор, Интерпретатор, Стратегия, Шаблонный метод, Состояние, Фасад. Кроме того, крупные по объему кода методы были вынесены в классы-операции, например, менеджер смены операций и формулировщих результата функции. Связность модулей была сведена к необходимуму минимумы. Все это позволило написать «чистый код, который работает [4]».

Изучения паттернов – процесс продолжительный и увлекательный, он позволять программистам почувствовать себя изобретателями – тем, кто стоял у истоков Computer Science. Поэтому азарт изучения паттернов еще долго будет входить в перечень радостей начинающих программистов.

## Список литературы

1. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2020. – 352 с.

2. Макконнелл С. Совершенный код. Мастер-класс. – Изд. «Русская редакция», 2015. – 896 с.

3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2020. – 368 с.

4. Фаулер М. Рефакторинг: Улучшение существующего кода. – Пер. с англ. – СПб: Символ-Плюс, 2018. - 432 с

## Сведения об авторах

**Миннахметов Эльдар Юлдашевич** – студент электротехнического факультета, кафедра «Информационные технологии и автоматизированные системы», Пермский национальный исследовательский политехнический университет, Пермь, e-mail: ciborg.brain@yandex.ru.

## About the authors

**Minnakhmetov Eldar Juldashevich** – student of electrotechnical faculty, Information Technologies and Automated Systems department, Perm National Research Polytechnic University, Perm, email: ciborg.brain@yandex.ru