Зачет по ППО II

(конспект книги Роберт Мартин Чистая Архитектура)

Часть II. Парадигмы программирования

- **Структурное** - ограничение на прямую передачу управления (goto - плохо, if/while - хорошо)
- **Объектно-ориентированное** - ограничение на косвенную передачу управления (отнимает указатели на функции)
- **Функциональное** - ограничение на присваивание (неизменяемость значений символов); лямбда исчисление

Парадигмы говорят скорее чего делать нельзя, а не что можно.

> ✂️ Каждая парадигма что-то отнимает.

Структурное программирование

Дейкстра обнаружил, что инструкция goto мешает рекурсивному разложению модулей на все меньшие и меньшие единицы; препятствует применению принципа "разделяй и властвуй".

Любую программу можно написать используя 3 структуры:

- **последовательность**
- **выбор**
- **итерации**

> 🧰 Структурное программирование дает возможность **функциональной декомпозиции**.

То есть решение большой задачи можно разложить на ряд функций.

Объектно-ориентированное программирование

- **Инкапсуляция** - возможность очертить круг связанных данных и функций. За пределами круга эти данные невидимы и доступны только некоторые функции.
- **Наследование** - повторное объявление группы переменных и функций в ограниченной области видимости.
- **Полиморфизм** - способность объекта использовать методы производного класса, который не существует на момент создания базового.

Факт поддержки языками ОО надежного и удобного механизма полиморфизма означает, что:

> любую зависимость исходного кода, можно **инвертировать**.

Инверсия зависимостей

Чтобы вызвать одну из функций верхнего уровня, функция main должна сослаться на модуль, содержащий эту функцию .

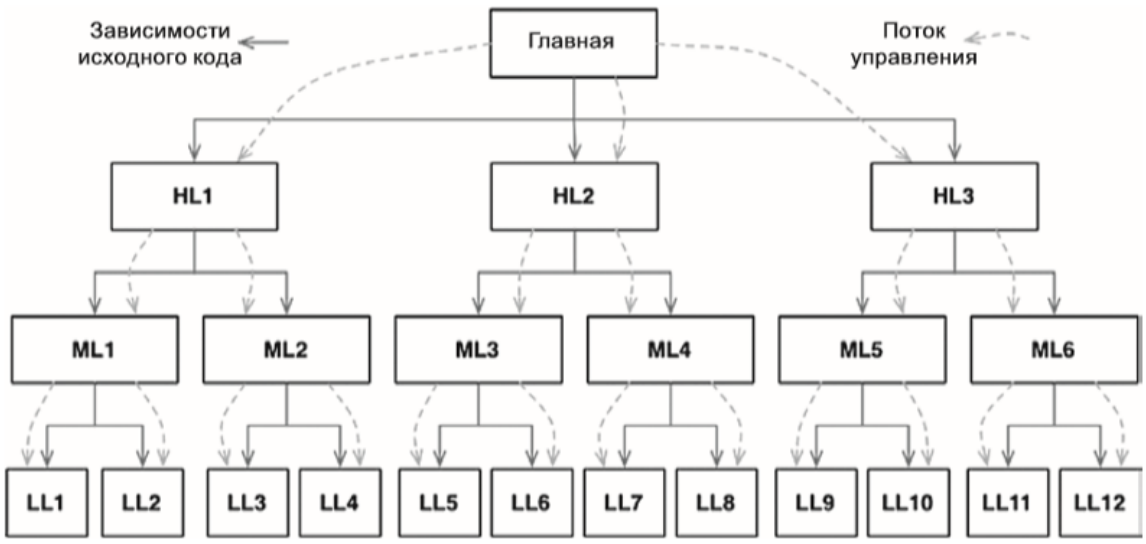

Рис. 5.1. Зависимости исходного кода следуют за потоком управления

После появления полиморфизма:

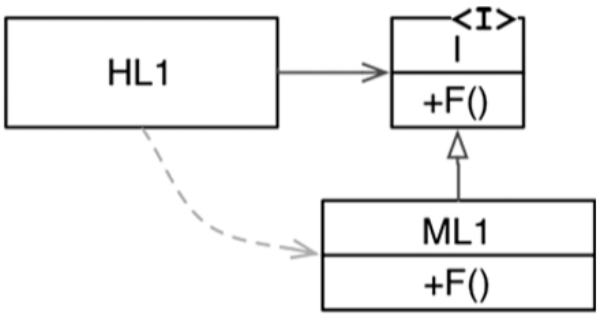

Рис. 5.2. Инверсия зависимости

При таком подходе архитекторы получают абсолютный **контроль над направлением зависимостей** в исходном коде .

Как следствие, бизнес-правила, ПИ и базу данных можно скомпилировать в три разных компонента. Компонент с бизнес-правилами не будет зависеть от компонентов, реализующих ПИ и базу данных (благодаря инверсии). Как результат, компоненты можно развертывать отдельно и независимо. Это **независимость развертывания.**

Если система состоит из модулей, которые можно развертывать независимо, их можно разрабатывать независимо, разными командами. Это **независимость разработки**.

Функциональное программирование

Эта парадигма в значительной мере основана на λ-исчислении.

В программе присутствуют инициализируемые переменные, но они никогда не изменяются.

Проблемы изменяемости переменных:

- состояние гонки
- взаимоблокировки
- проблемы параллельного обновления

Компромисс: ограничение изменяемости. Деление служб внутри приложения на изменяемые и неизменяемые.