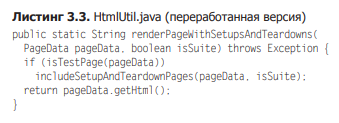
**Количество операций, выполняемых функцией.**



ФУНКЦИЯ ДОЛЖНА ВЫПОЛНЯТЬ ТОЛЬКО ОДНУ ОПЕРАЦИЮ. ОНА ДОЛЖНА ВЫПОЛНЯТЬ ЕЕ ХОРОШО. И НИЧЕГО ДРУГОГО ОНА ДЕЛАТЬ НЕ ДОЛЖНА.

Проблема в том, что иногда бывает трудно определить, что же считать «одной операцией». В листинге 3.3 выполняется одна операция? Легко возразить, что в нем выполняются минимум три операции:

1. Функция проверяет, является ли страница тестовой страницей.

2. Если является, то в нее включаются начальные и конечные блоки.

3. Для страницы генерируется код HTML.

Так как же? Сколько операций выполняет функция — одну или три? Обратите внимание: три этапа работы функции находятся на одном уровне абстракции под объявленным именем функции. Ее можно было бы описать в виде короткого TO1 -абзаца:

 TO RenderPageWithSetupsAndTeardowns, мы проверяем, является ли страница тестовой, и если является — включаем начальные и конечные блоки. В любом случае для страницы генерируется код HTML. Если функция выполняет только те действия, которые находятся на одном уровне под объявленным именем функции, то эта функция выполняет одну операцию. В конце концов, функции пишутся прежде всего для разложения более крупной концепции (иначе говоря, имени функции) на последовательность действий на следующем уровне абстракции.

С другой стороны, осмысленно сократить листинг 3.3 очень трудно. Команду if можно вынести в функцию с именем includeSetupsAndTeardownsIfTestPage, но это простая переформулировка кода без изменения уровня абстракции. Итак, чтобы определить, что функция выполняет более одной операции, попробуйте извлечь из нее другую функцию, которая бы не являлась простой переформулировкой реализации

**Секции в функциях**

Взгляните на листинг 4.7 на с. 98. Обратите внимание: функция generatePrimes разделена на секции (объявления, инициализация, отбор). Это очевидный признак того, что функция выполняет более одной операции. Функцию, выполняющую только одну операцию, невозможно осмысленно разделить на секции.

**Один уровень абстракции на функцию**

Чтобы убедиться в том, что функция выполняет «только одну операцию», необходимо проверить, что все команды функции находятся на одном уровне абстракции. Легко убедиться, что листинг 3.1 нарушает это правило. Некоторые из его концепций — например, getHtml() — находятся на очень высоком уровне абстракции; другие (скажем, String pagePathName = PathParser.render(pagePath)) — на среднем уровне. Наконец, третьи — такие, как .append("\n") — относятся к чрезвычайно низкому уровню абстракции. Смешение уровней абстракции внутри функции всегда создает путаницу. Читатель не всегда понимает, является ли некоторое выражение важной концепцией или второстепенной подробностью. Что еще хуже, при их смешении функция постепенно начинает обрастать все большим количеством второстепенных подробностей.

**Чтение кода сверху вниз: правило понижения**

Код должен читаться как рассказ — сверху вниз [KP78, p. 37]. За каждой функцией должны следовать функции следующего уровня абстракции. Это позволяет читать код, последовательно спускаясь по уровням абстракции в ходе чтения списка функций. Я называю такой подход «правилом понижения». Сказанное можно сформулировать и иначе: программа должна читаться так, словно она является набором TO-абзацев, каждый, из которых описывает текущий уровень абстракции и ссылается на последующие TO-абзацы следующего нижнего уровня.  Чтобы включить начальные и конечные блоки, мы сначала включаем начальные блоки, затем содержимое тестовой страницы, а затем включаем конечные блоки.

• Чтобы включить начальные блоки, мы сначала включаем пакетные начальные блоки, если имеем дело с пакетом тестов, а затем включаем обычные начальные блоки.

• Чтобы включить пакетные начальные блоки, мы ищем в родительской иерархии страницу SuiteSetUp и добавляем команду include с путем к этой странице.

Чтобы найти в родительской иерархии…

Опыт показывает, что программистов очень трудно научить следовать этому правилу и писать функции, остающиеся на одном уровне абстракции. Тем не менее освоить этот прием очень важно. Он играет ключевую роль для создания коротких функций, выполняющих только одну операцию. Построение кода по аналогии с набором последовательных TO-абзацев — эффективный метод поддержания единого уровня абстракции.