Разбор домашнего задания #1

Trie

Михаил Веселов
mikhail.veselov@gmail.com

Юрий Литвинов
yurii.litvinov@gmail.com

05.10.2017г

Критерии оценки

- Возможное количество баллов: от 0 до 10
- Оценивался код самой реализации Trie
- Тестовый проект был необязательным
- За неработающую реализацию снимались баллы по формуле $n/5$ (n — количество некорректных методов)
- За оформление кода снималось от 1 до 2 баллов, в зависимости от количества замечаний

Частые ошибки в реализации

- Расчёт на то, что символы — только латинские
  - Ломается на добавлении русских букв
- Отсутствие проверки на конец слова в вершине: флаг (is)terminal
  - Ломается на проверке вхождения подстроки “a” после добавления полной строки “ab”
- Несогласованная работа с пустыми строками
  - Ломается, когда нельзя добавить пустую строку, но можно удалить
- Не создан pull-request и ветка для задания
  - В задании это было обязательным пунктом

Частые недочёты в Unit-тестах

- Именование тестов
  - Название теста должно содержать максимум информации об ошибке (удобно при просмотре списка из 200 тестов, упавших после merge)
  - Статья Naming standards for unit tests как **пример** именования (на MSDN тоже есть длинная статья)
- Один тест — не один случай проверки
  - Нехорошо, когда в одном тесте проверяется сразу несколько действий
  - Тест состоит из трёх частей, даже в простых случаях: Arrange, Act, Assert
- Неполное покрытие логики тестами (вопрос на SO про утилиты)
- Повторная инициализация переменных в каждом тесте вместо использования SetUp/TestInitialize/IDisposable

Стандарты кодирования (holywar!)

- Обратимся к первоисточнику (вспоминаем лекцию введения)
  - Наберите в Google *C# Coding Conventions*, нажмите “Мне повезёт!”, [и если вам повезёт,] вы попадёте на официальную точку зрения
  - В разделе Framework Design Guidelines раскрыты вопросы дизайна приложения, в том числе именования (об этом дальше)
- StyleCop
  - Не все правила соответствуют практике
  - Особенно много конфликтов с анализатором ReSharper
  - Не надо отключать правила пачками
    (нет никакого смысла анализировать код, если отключено >= 7 правил)

Holywar #2: Фигурные скобки

```
if (ch == 'h')
   Console.Write("H");
if (ch == 'h') Console.Write("H");
if (ch == 'h') while (true) for (;;) Console.Write("H");
```

- Выделяйте в блок даже одну короткую строку кода
- Отсутствие скобок ЯВЛЯЕТСЯ ошибкой, НУЖНО исправлять

Holywar #2: Фигурные скобки

- Каков результат этого метода?
- Сколько времени ушло на ответ?

```
// корректный C#
private int GetInt()
{
    if (Check());
        return 4;

    if (Check())
        return 5;
        return 6;

    return 7;
}
```

Holywar #3: поля классов и this

- (мнение MS):
  - **public** и **protected** не-**static** поля запрещены
  - Именование **private** и **internal** полей не регламентировано
  - Не используйте префиксов типа m_имяПоля
- (мнение StyleCop):
  - **this**.имяПоля
  - Символ “_” запрещён в начале имени члена класса
- (мнение Resharper):
  - **this** избыточен
  - **private** поле должно начинаться с “_”

Holywar #3: поля классов и this

- (практика):
  - Чем дольше вы пишите код, тем очевиднее вам избыточность **this**:
    this.Count += **this**.newItems * **this**.GetPrice();
  - “_” - простой способ различать поле класса и параметр/переменную
  - Если в вашем проекте подключён StyleCop и не выключены правила про именование полей и **this**, подобный код НЕ является ошибкой, МОЖНО не исправлять
  - В противном случае следуйте мнению Resharper
- Holywar на SO #1
- Holywar на SO #2

Holywar #4: Керниган & Ричи

- C, C++, Java, Javascript (египетские скобки):

```
private void method() {
}
```

- C# (пустая строка с одной фигурной скобкой):

```
private void method()
{
}
```

- Поведение C# в Visual Studio можно отключить
- Оба способа приемлемы, если соблюдаются по ВСЕМУ проекту

Holywar #5: **for** или **foreach**

- Разные замеры производительности — разные результаты (вспоминаем лекцию про benchmarking)
- Обращение по индексу в **for** может быть быстрее
- **foreach** отслеживает изменение коллекции и в целом понятнее
- Оба способа приемлемы, если соблюдаются по ВСЕМУ проекту

Holywar #6: **using** внутри или снаружи

- (мнение StyleCop):
  - Внутри, и вот почему
- (мнение MS):
  - Снаружи, и директивы “System.*” в начале списка
- (практика):
  - Если у вас код зависит от локальности директив **using**, это необычно
  - Обычно — один файл, одно пространство имён, директивы снаружи
  - Если в вашем проекте подключён StyleCop и не выключено правило про директивы **using**, подобный код НЕ является ошибкой, МОЖНО не исправлять

Мысли вслух

- Некоторые решения содержат покрытие тестами на 100%, а некоторые не работают с волшебной строкой
- Обращение по индексу массива с помощью переменной **char** это:
  - нехорошо (можно выпасть за пределы массива)
  - непонятно (происходит неявное преобразование char в int)
- NUnit и MSTest можно настроить на получение нескольких аргументов на вход: вопрос на SO
- Чем меньше target версия .Net Framework (4.5.2) или .Net Standard (1.1), тем на большем количестве устройств будет выполняться ваш код

Итоги

- Стандарты кодирования — это соглашения
- В случае несогласия — спорьте
- В случае неясного комментария — спрашивайте