Рефакторинг с Roslyn

Марат Тураев, Антон Афанасьев

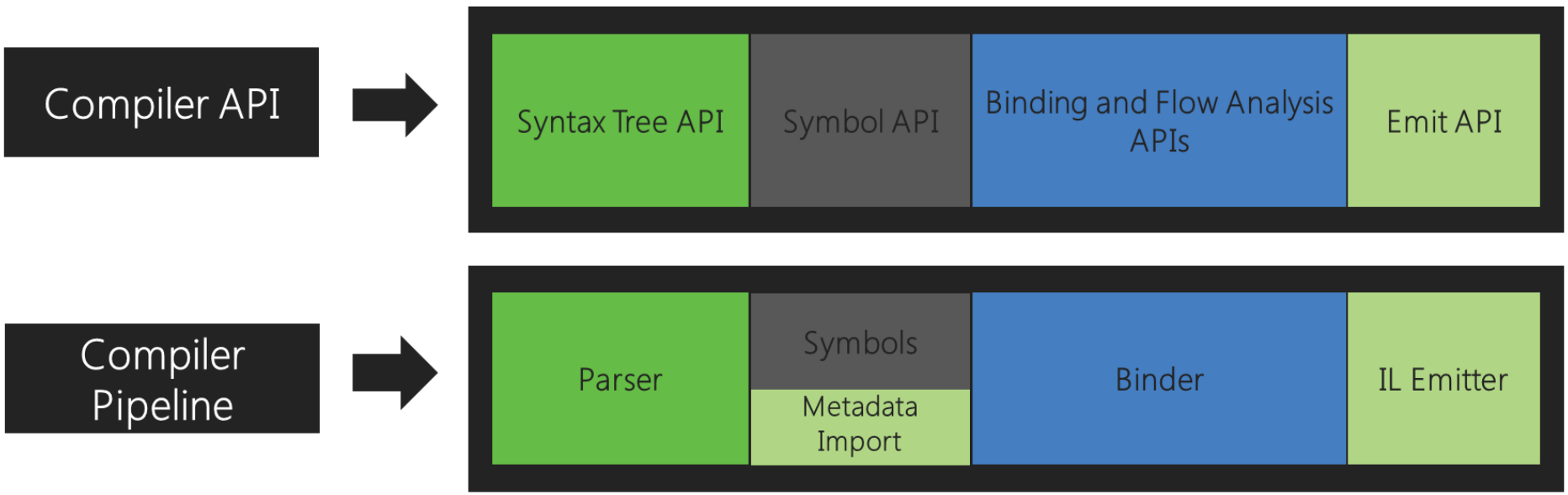
Руководитель: Александр Опейкин

26 мая 2014 г.

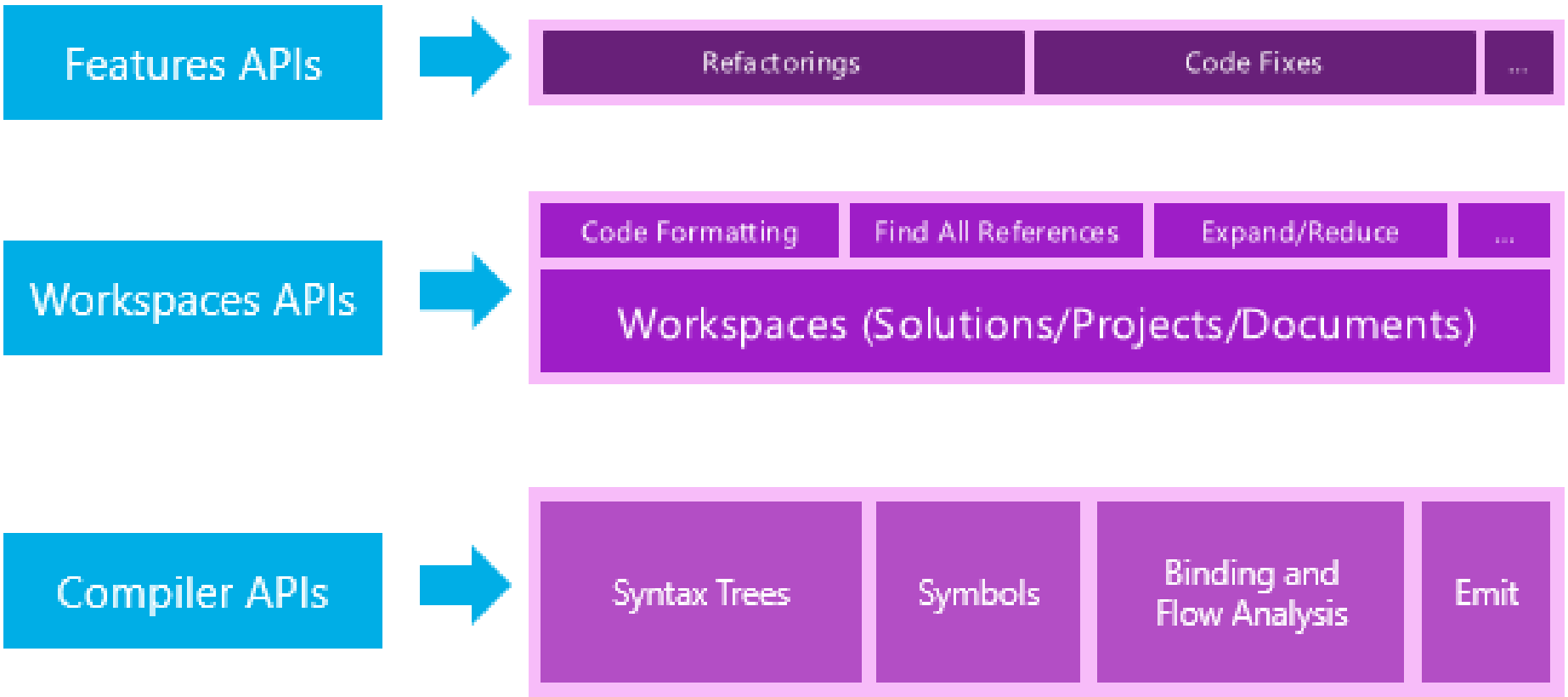
Roslyn

.NET Compiler Platform

- Предоставляет API ко всем этапам компиляции C# и Visual Basic.
- End User Preview (25 Apr 2014)
- Open Source



Roslyn API


Features APIs
Refactorings
Code Fixes
...
Workspaces APIs
Code Formatting
Find All References
Expand/Reduce
...
Workspaces (Solutions/Projects/Documents)
Compiler APIs
Syntax Trees
Symbols
Binding and Flow Analysis
Emit

Мотивация и цель

- Roslyn — будущий основной компилятор Visual Studio
- Обещает облегчить создание инструментов для работы с кодом, в том числе с интеграцией в Visual Studio (рефакторинги, quick fixes).

Цель: реализовать часть функциональности ReSharper на Roslyn.

Задачи

- Spell checker
- Refactorings
  - Use Base class when possible
  - Introduce argument
  - Convert for to foreach
  - Convert foreach to for

Refactoring

Use Base class

```
class A
{
    public int aMethod()
    {
        return 6;
    }
}

class B : A
{
    public int bMethod1()
    {
        return 34;
    }

    void bMethod2(B a1)
    {
        A a2 = a1;
        i = a2.aMethod() + 3;
    }

    static public int i = 4;

}
```

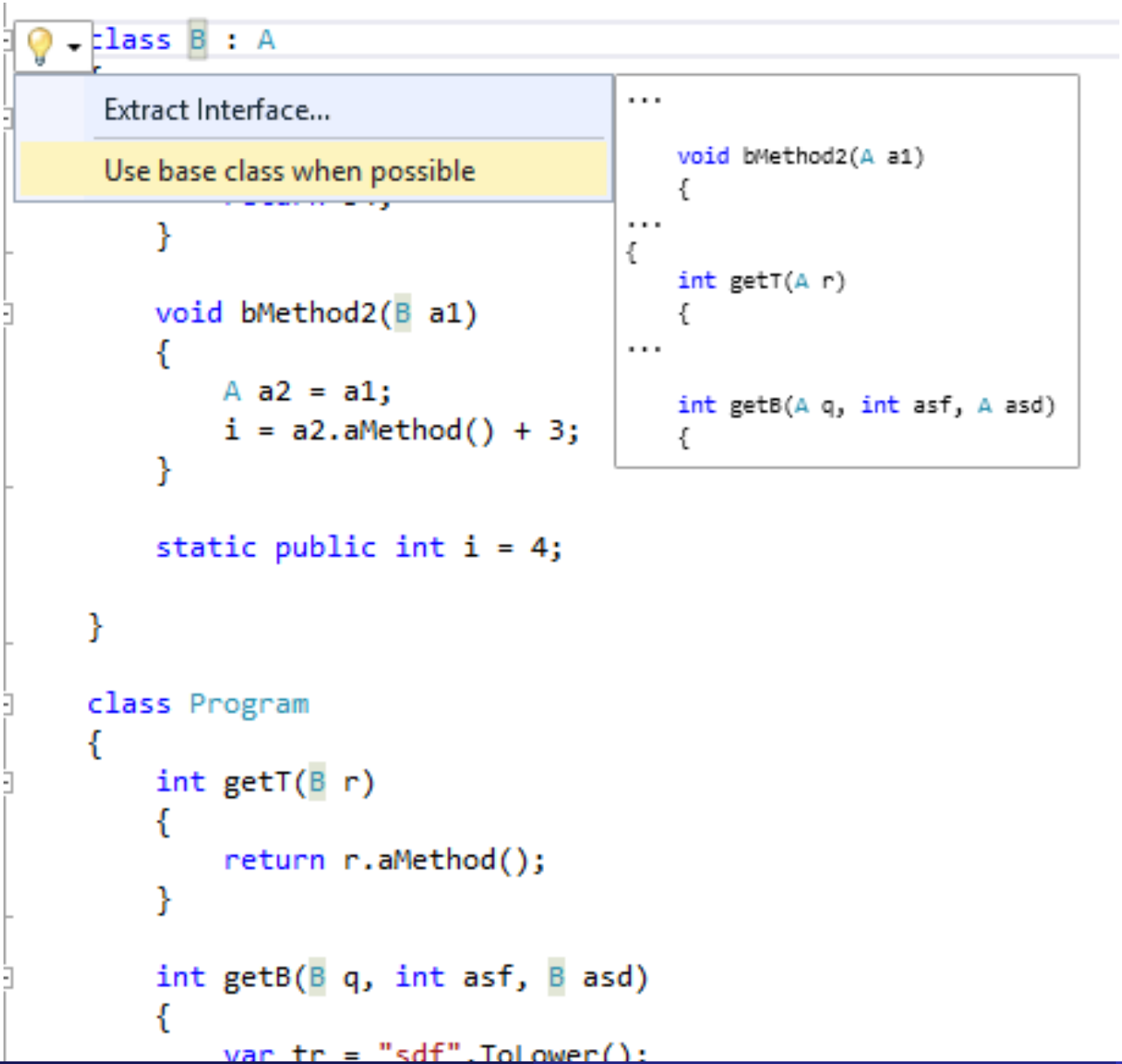
Refactoring

Use Base class

```
class B : A
{
    public int bMethod1()
    {
        return 34;
    }

    void bMethod2(B a1)
    {
        A a2 = a1;
        i = a2.aMethod() + 3;
    }

    static public int i = 4;

}

class Program
{
    int getT(B r)
    {
        return r.aMethod();
    }

    int getB(B q, int asf, B asd)
    {
        var tr = "sdf".ToLower();
        getT(asd);

        return q.aMethod() + asf + getT(q);
    }
```

Refactoring

Use Base class



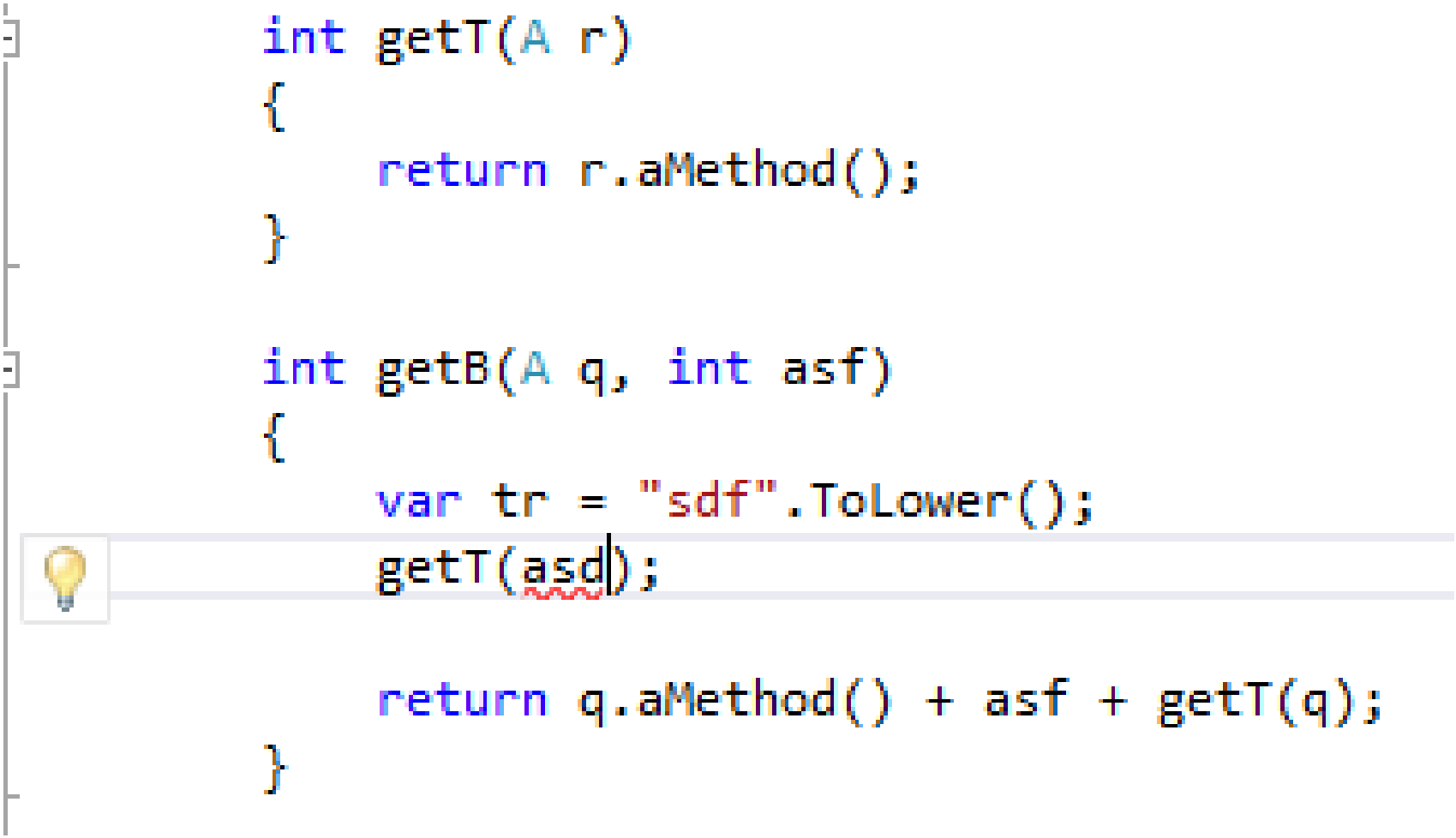
Refactoring

Use Base class

```
class B : A
{
    public int bMethod1()
    {
        return 34;
    }

    void bMethod2(A a1)
    {
        A a2 = a1;
        i = a2.aMethod() + 3;
    }

    static public int i = 4;

}

class Program
{
    int getT(A r)
    {
        return r.aMethod();
    }

    int getB(A q, int asf, A asd)
    {
        var tr = "sdf".ToLower();
        getT(asd);

        return q.aMethod() + asf + getT(q);
    }
```

Refactoring

Introduce argument



Refactoring

Introduce argument

```
int getT(A r)
{
    return r.aMethod();
}

int getB(A q, int asf, A asd)
{
    var tr = "sdf".ToLower();
    getT(asd);

    return q.aMethod() + asf + getT(q);
}
```

Refactoring

Foreach to for

```
static void Main(string[] args)
{
    List<int> a = new List<int> { 3, 4, 6, 4 };
    int[] b = new int[435];

    foreach(var it in b)
    {
        Console.Write(it);
        for(int i = 0; i < a.Count; i++)
        {
            Console.WriteLine(a[i]);
            int asd = a[i] + 3 + it;
            Console.WriteLine(asd * a[i]);
            B.i += 5;
        }
    }

}
```

Refactoring

Foreach to for

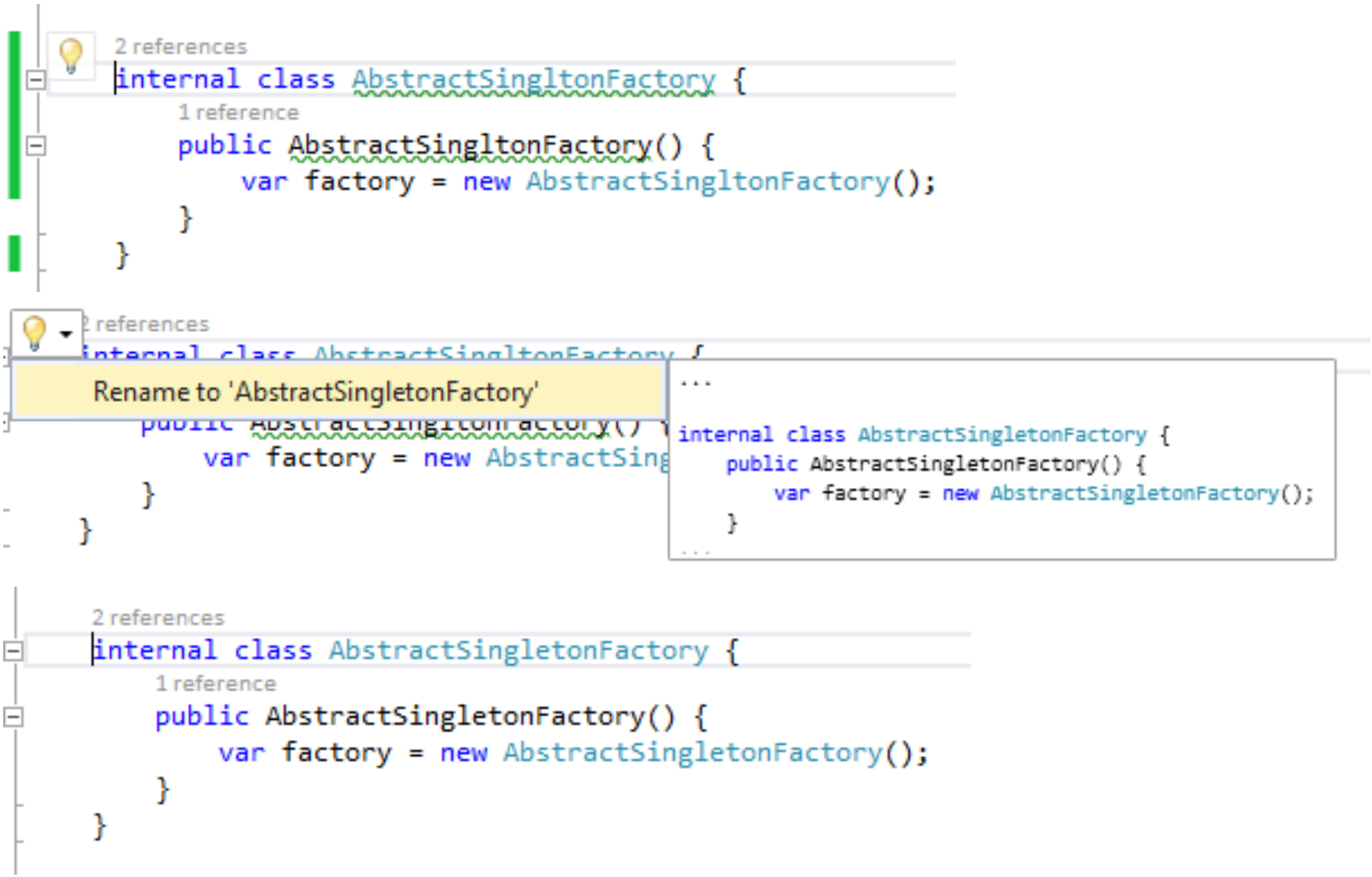
```
static void Main(string[] args)
{
    List<int> a = new List<int> { 3, 4, 6, 4 };
    int[] b = new int[435];

    for(int i1 = 0; i1 < b.Length; i1++)
    {
        Console.Write(b[i1]);
        for(int i = 0; i < a.Count; i++)
        {
            Console.WriteLine(a[i]);
            int asd = a[i] + 3 + b[i1];
            Console.WriteLine(asd * a[i]);
            B.i += 5;
        }
    }

}
```

Refactoring

For to foreach

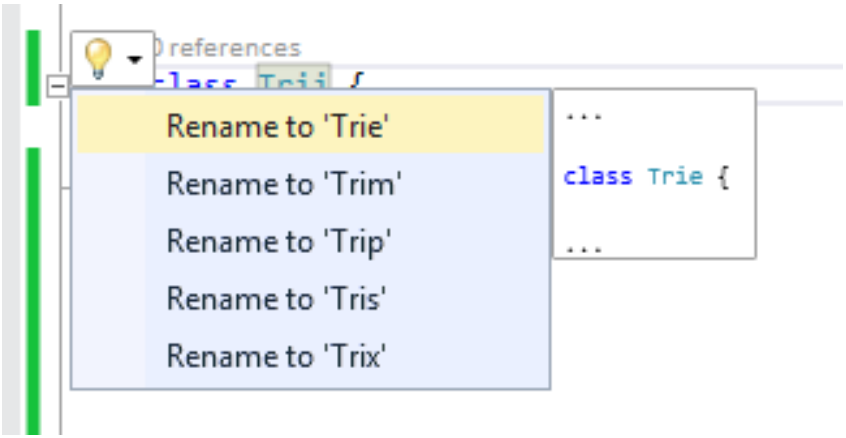
```
static void Main(string[] args)
{
    List<int> a = new List<int> { 3, 4, 6, 4 };
    int[] b = new int[435];

    for(int i1 = 0; i1 < b.Length; i1++)
    {
        Console.Write(b[i1]);
        foreach(var it in a)
        {
            Console.WriteLine(it);
            int asd = it + 3 + b[i1];
            Console.WriteLine(asd * it);
            B.i += 5;
        }
    }

}
```

Spell Checker



Spell Checker

- $O(T)$; $O(P^{\#error+1})$ – построение и поиск в словаре (trie)
- Классы, поля, переменные, методы . . .
- CamelCase токенизация
- Множественный выбор замены



Полученные навыки

- Roslyn API
- Visual Studio plugins
- Архитектура компиляторов
- Строковые алгоритмы

Спасибо за внимание!

GitHub

https://github.com/Icemore/RoslynExperiments