1.Tracer

Необходимо реализовать измеритель времени выполнения методов.

Класс должен реализовывать следующий интерфейс:

public interface ITracer

{

// вызывается в начале замеряемого метода

void StartTrace();​

// вызывается в конце замеряемого метода

void StopTrace();​

// получить результаты измерений

TraceResult GetTraceResult();

}

Структура TraceResult на усмотрение автора.

Tracer должен собирать следующую информацию об измеряемом методе:

* имя метода;
* имя класса с измеряемым методом;
* время выполнения метода.

Пример использования:

public class Foo

{

private Bar \_bar;

private ITracer \_tracer;

​

internal Foo(ITracer tracer)

{

\_tracer = tracer;

\_bar = new Bar(\_tracer);

}

public void MyMethod()

{

\_tracer.StartTrace();

...

\_bar.InnerMethod();

...

\_tracer.StopTrace();

}

}

​

public class Bar

{

private ITracer \_tracer;

​

internal Bar(ITracer tracer)

{

\_tracer = tracer;

}

public void InnerMethod()

{

\_tracer.StartTrace();

...

\_tracer.StopTrace();

}

}

Также должно подсчитываться общее время выполнения анализируемых методов в одном потоке.

Результаты трассировки вложенных методов должны быть представлены в соответствующем месте в дереве результатов.

Результат измерений должен быть представлен в двух форматах: **JSON** и **XML** (для классов, реализующих сериализацию в данные форматы, необходимо **разработать общий интерфейс**).

Пример результата в JSON:

{

"threads": [

{

"id": "1",

"time": "42ms",

"methods": [

{

"name": "MyMethod",

"class": "Foo",

"time": "15ms",

"methods": [

{

"name": "InnerMethod",

"class": "Bar",

"time": "10ms",

"methods": ...

}

]

},

...

]

},

{

"id": "2",

"time": "24ms"

...

}

]

}

Пример результата в XML:

<root>

<thread id="1" time="42ms">

<method name="MyMethod" time="15ms" class="Foo">

<method name="InnerMethod" time="10ms" class="Bar"/>

</method>

...

</thread>

<thread id="2" time="24ms">

...

</thread>

</root>

Готовый результат (полученный JSON и XML) должен **выводиться в консоль** и **записываться в файл.** Для данных классов необходимо **разработать общий интерфейс**, допустимо создать один переиспользуемый класс, не зависящий от того, куда должен выводиться результат.

Код лабораторной работы должен состоять из трех проектов:

* Основная часть библиотеки, реализующая измерение и форматирование результатов.
* Модульные тесты для основной части библиотеки.
* Консольное приложение, содержащее классы для вывода результатов в консоль и файл, демонстрирующее общий случай работы библиотеки (в многопоточном режиме при трассировке вложенных методов).