ISEL

Ambientes Virtuais de Execução

2021

Week 8 – Benchmarking

Performance Evaluation

Benchmarking - Performance Evaluation

Program/Application to measure performance (i.e. desempenho)

Benchmarking - Performance Evaluation

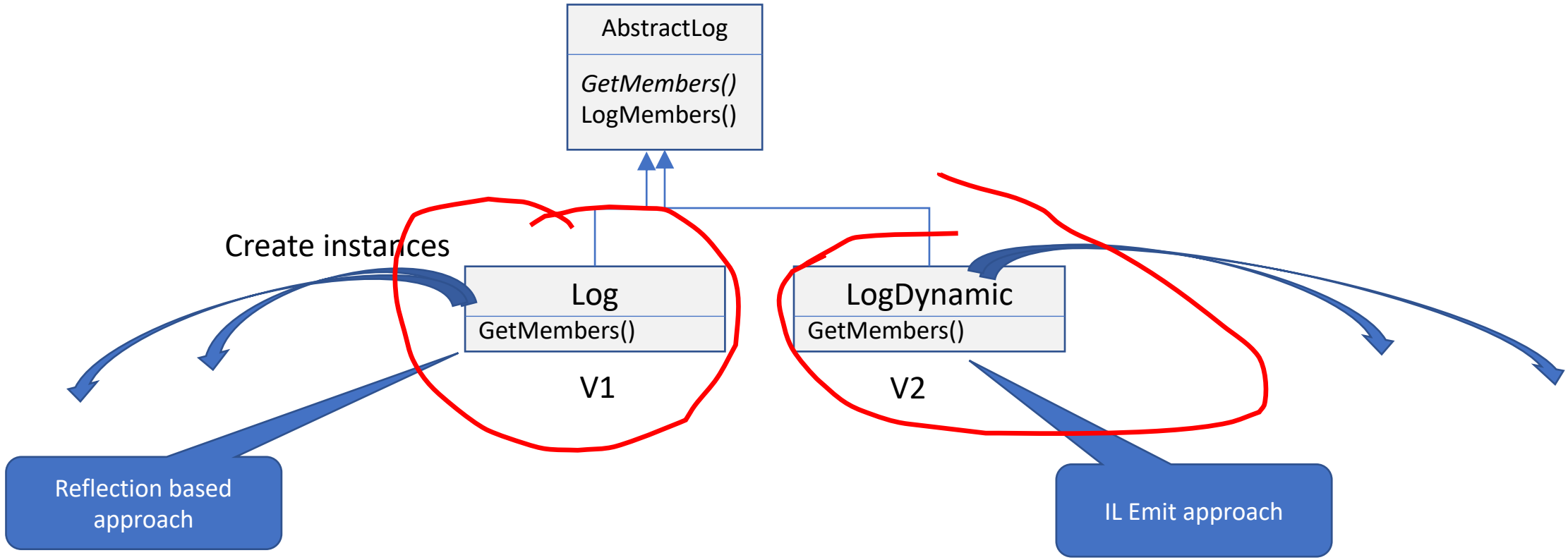
Benchmarking checks Performance – How fast it is?

!=

Unit tests checks correction

(the expected result is the actual result)

E.g. Benchmarking Log with LogDynamic



Benchmarking - Performance Evaluation

- **Are not unit tests**
- Focus on **CPU bound** operations:
  - **Eliminate** surrounding overheads, such as, *initializing* instances, application *setup*, *bootstrap*, etc.

```
Student s = new Student(763547, "Maria Papoila", 3547, "maria");
Log l = new Log();
l.Info(s);
```

Exclude it from the performance evaluation

- Avoid **IO operations,** because could be non-deterministic, e.g. the response of a network server.

Benchmarking - Performance Evaluation

```
Student s = new Student(763547, "Maria Papoila", 3547, "maria");
Log l = new Log();
l.Info(s);
```

Exclude it from the performance evaluation

- Split in Setup and Measure methods
- Avoid IO Operations e.g. **do not print to standard output (Console)**

```
Student s = new Student(763547, "Maria Papoila", 3547, "maria");
Log l = new Log(new BufferPrinter());
```

Setup

```
l.Info(s);
```

Measure

Benchmarking - Performance Evaluation

How can we build a Benchmark application?

Recommended approach:

- Java – **JMH** developed by Shipilev, today is part of JDK
- .Net –**BenchmarkDotNet** https://github.com/dotnet/BenchmarkDotNet
- Javascript - https://benchmarkjs.com/
- Others….

Benchmarking - Performance Evaluation

Reduce the side-effects from the VM:

- Runtime optimizations such as *code inlining*, dead-code elimination, etc
- GC Execution
- Exclude out of bounds results (e.g. the first call include the Jitter compilation)

➔Execute many iterations of the operation, and **NOT a single execution**!

Benchmarking Results?

Approaches:

- Average
- Average with standard deviation
- **The best result**

NBench follows this approach

Units:

- **Durations** in milliseconds, seconds, whatever…..
- **Throughput** - number of operations in a time slot, e.g. ops/ms, ops/sec

NBench

Entry point – receives the *operation* as argument

```
public static void Bench(Action handler)
```

- E.g. `NBench.Bench(Program.BenchLogReflectionStudent);`

```
logReflect.Info(maria);
```

Duration of each iteration

Number of iterations

```
Perform(Action handler, int time, int iters)
```

```
for (int i = 0; i < iters; i++) {
  ...
  ... CallWhile(handler, time);
  ...
}
```

During `time` it calls the `handler` and counts the *number of calls* to `handler`
=> `Throughput` = *number of calls* / `time`