0.1 Introdução

Este documento apresenta um resumo dos benchmarks que podem ser realizados com a ferramenta *Java Microbenchmark Harness (JMH)*, descrevendo os tipos de testes e as variáveis utilizadas.

0.2 Resumo de Benchmarks

Classe	**Descrição**
JMHSample_01_HelloWorld	Exemplo "Hello World"do JMH, mede apenas o overhead do JMH.
JMHSample_02_BenchmarkModes	Mostra como medir throughput, averageTime, sampleTime, singleShotTime e all.
JMHSample_03_States	Mostra como usar estados no JMH para benchmarks concorrentes.
JMHSample_04_DefaultState	Mostra uma forma mais simples de usar estado no JMH.
JMHSample_05_StateFixtures	Mostra o uso de fixtures no JMH (@Setup e @TearDown).
JMHSample_06_FixtureLevel	Demonstra os diferentes níveis de fixtures: Trial, Iteration e Invocation.
JMHSample_07_FixtureLevelInvocation	Mostra o uso de fixtures no nível de invocação e seus custos de performance.
JMHSample_08_DeadCode	Demonstra como evitar eliminação de código morto usando retorno de valores e Blackhole.
JMHSample_09_Blackholes	Explica o uso de Blackhole para consumir valores e evitar otimizações indesejadas.
JMHSample_10_ConstantFold	Mostra como evitar constant folding (dobramento de constantes) pelo compilador.
JMHSample_11_Loops	Demonstra por que fazer loops dentro de métodos de benchmark é uma má prática.
JMHSample_12_Forking	Explica a importância do forking para isolar execuções de benchmark.
JMHSample_13_RunToRun	Demonstra variações entre execuções (run-to-run variance).
JMHSample_14_FalseSharing	Mostra o problema de false sharing em benchmarks concorrentes.
JMHSample_15_Asymmetric	Demonstra benchmarks assimétricos com diferentes números de threads para operações diferentes.
JMHSample_16_CompilerControl	Mostra como controlar o comportamento do compilador (inline, dont_inline, etc).
JMHSample_17_SyncIterations	Demonstra sincronização de iterações em benchmarks multi-threaded.
JMHSample_18_Control	Mostra como usar Control para interromper benchmarks dinamicamente.
JMHSample_19_Inline	Demonstra os efeitos de inlining em benchmarks.

Classe	**Descrição**
JMHSample_20_Annotations	Mostra como usar anotações JMH de forma efetiva.
JMHSample_21_ConsumeCPU	Demonstra como consumir ciclos de CPU de forma controlada.
JMHSample_22_FalseSharing	Outro exemplo sobre false sharing com diferentes técnicas de mitigação.
JMHSample_23_AuxCounters	Mostra como usar contadores auxiliares para coletar métricas adicionais.
JMHSample_24_Inheritance	Demonstra como herança funciona com benchmarks JMH.
JMHSample_25_API_GA	Mostra como usar a API do JMH programaticamente para algoritmos genéticos.
JMHSample_26_BatchSize	Demonstra o uso de batchSize para benchmarks com custo variável.
JMHSample_27_Params	Mostra como parametrizar benchmarks com diferentes valores de entrada.
JMHSample_28_BlackholeHelpers	Demonstra helpers adicionais do Blackhole para consumir valores.
JMHSample_29_StatesDAG	Mostra como criar grafos direcionados acíclicos (DAG) de estados.
JMHSample_30_Interrupts	Demonstra como lidar com interrupções durante benchmarks.
JMHSample_31_InfraParams	Mostra como injetar parâmetros de infraestrutura do JMH.
JMHSample_32_BulkWarmup	Demonstra warmup em lote para múltiplos benchmarks.
JMHSample_33_SecurityManager	Mostra como benchmarks interagem com SecurityManager.
JMHSample_34_SafeLooping	Demonstra técnicas seguras para fazer loops em benchmarks.
JMHSample_35_Profilers	Mostra como usar profilers integrados (stack, gc, perfasm, etc).
JMHSample_36_BranchPrediction	Demonstra efeitos de predição de branch em performance.
JMHSample_37_CacheAccess	Mostra padrões de acesso a cache e seus impactos.
JMHSample_38_PerInvokeSetup	Demonstra setup por invocação e suas implicações de performance.