Java Microbenchmark Harness (JMH)

1 O que é JMH

JMH (Java Microbenchmark Harness) é uma ferramenta oficial do OpenJDK que mede o desempenho de pequenos trechos de código Java, chamados *microbenchmarks*. Microbenchmarks medem o tempo de execução de um trecho isolado de código, normalmente dentro de um único método ou bloco lógico.

2 Anotações Principais

2.1 @Benchmark

Indica quais métodos serão medidos durante o benchmark.

```java
@Benchmark
public int addOne() {
    return x + 1;
}
```

2.2 @BenchmarkMode

Define como o desempenho será medido (*tipo de métrica*).

Modo	**Descrição**
Mode.Throughput	Operações por segundo
Mode.AverageTime	Tempo médio por operação
Mode.SampleTime	Estatísticas: min, max, p50, p90, p99
Mode.SingleShotTime	Tempo de uma única execução
Mode.All	Executa todos os modos simultaneamente

2.3 @OutputTimeUnit

Define a unidade de tempo usada na medição dos benchmarks.

Parâmetro	**Descrição**
TimeUnit.SECONDS	Mede em segundos
TimeUnit.MILLISECONDS	Mede em milissegundos
TimeUnit.MICROSECONDS	Mede em microssegundos
TimeUnit.NANOSECONDS	Mede em nanossegundos

2.4 @State

Mantém variáveis ou objetos durante a execução do benchmark.

Escopo	**Descrição**
@State()	Mantém variáveis durante o benchmark
@State(Scope.Benchmark)	Todas as threads compartilham a mesma instância
@State(Scope.Thread)	Cada thread tem sua própria instância
@State(Scope.Group)	Instância compartilhada somente entre threads do mesmo grupo

2.5 @Group

Agrupa threads que compartilham um mesmo estado.

Anotação	Descrição
@Group(”NomeDoGrupo”)	Define grupo de threads que compartilham o mesmo estado

2.6 @Setup / @TearDown

Executam métodos antes e depois da execução do benchmark.

Anotação	Descrição
@Setup	Método chamado antes da execução do benchmark
@TearDown	Método chamado após a execução do benchmark
@TearDown(Level.Iteration)	Método chamado após cada iteração

2.7 @Fork

Define quantos processos separados (forks) serão criados.

Anotação	Descrição
@Fork(N)	N forks; cada fork cria uma JVM limpa
@Fork(value=3, warmups=1)	value → quantidade de forks; warmups → rodadas de aquecimento

2.8 @Warmup

Configura a fase de aquecimento da JVM antes da medição do benchmark.

Parâmetro	Descrição
iterations=5	Executa 5 vezes durante a fase de aquecimento
time=100	Cada execução dura 100 unidades de tempo
timeUnit=TimeUnit.MILLISECONDS	Unidade de tempo utilizada (ms)

2.9 @Measurement

Configura a fase de medição do benchmark.

Parâmetro	Descrição
iterations=5	Executa 5 vezes durante a fase de medição
time=100	Cada execução dura 100 unidades de tempo
timeUnit=TimeUnit.MILLISECONDS	Unidade de tempo utilizada (ms)

2.10 @AuxCounters

Registra contadores auxiliares durante o benchmark.

Tipo	Descrição
AuxCounters.Type.OPERATIONS	Contadores representam quantidade de operações
AuxCounters.Type.EVENTS	Contadores monitoram ocorrências específicas, não apenas operações

2.11 @Param

Define valores possíveis que o benchmark vai testar.

```
@Param({"1", "31", "65", "101", "103"})
public int arg;

@Param({"0", "1", "2", "4", "8", "16", "32"})
public int certainty;

@Benchmark
public boolean bench() {
    return BigInteger.valueOf(arg).isProbablePrime(certainty);
}
```

2.12 @OperationsPerInvocation

Indica quantas operações são realizadas por invocação do método.

```
@OperationsPerInvocation(10)
public void doTenOps() {
    for (int i = 0; i < 10; i++) {
        computeSomething();
    }
}
```