1 Description

Perfidix is a generic benchmarking tool which works with similar annotations as JUnit 4.x. Methods which are void and parameter-free can be annotated. The annotated methods are then executed multiple times and the time of their execution is measured. Several statistical computations will then be applied to this results. A table with all results will then given back to the user.
Perfidix gives the possibility, similar to JUnit, of settingUp and cleaning methods which can be used for settings which should not be measured in the bench itself. Below, all possible annotations are listed.

2 Annotations

All methods annotated with the following metadata should be parameter and returnvalue free. Each annotation mustn't occur more than one time in a class, except the *Bench* annotation:

2.1 "@BenchClass"

- Has to be placed before the class declaration
- Each void-method, which has none of the below annotations is benched.

2.1.1 "@BenchClass(runs=)"

- Sets the number of runs for all benchs
- Can be overridden by the *Bench* annotation with own run-parameter

2.2 "@BeforeBenchClass"

- Executed before the first bench-method
- Executed once per class

2.3 "@BeforeFirstBenchRun"

- Executed before every bench-method and after the *BeforeBenchClass* - annotated method
- Executed for all bench-methods but just once for all runs

2.4 "@BeforeEachBenchRun"

- Executed before every bench-method and after the *BeforeFirstBenchRun* - annotated method
- Executed for all bench-methods before every run

2.5 "@Bench"

- Annotates method to bench

2.5.1 "@Bench(beforeFirstBenchRun=)"

- Specific setUp-method for this bench for settings before the bench
- is executed once for this bench

2.5.2 "@Bench(beforeEachBenchRun=)"

- Specific setUp-method for this bench for settings before the bench
- is executed for every run for this bench

2.5.3 "@Bench(afterEachBenchRun=)"

- Specific tearDown-method for this bench after the bench
- is executed after every run for this bench

2.5.4 "@Bench(afterLastBenchRun=)"

- Specific tearDown-method for this bench after the bench
- is executed after the last run of this bench

2.5.5 "@Bench(runs=)"

- Sets the number of runs for this method.
- Overrides the default value and a possible setting from a *BenchClass* annotation of the corresponding class.

2.6 "@AfterEachBenchRun"

- Executed after every bench-method
- Executed for all bench-methods after every run

2.7 "@AfterLastBenchMethod"

- Executed after every bench-method and after the *AfterEachBenchRun* - annotated method
- Executed for all bench-methods after the last run

2.8 "@AfterBenchClass"

- Executed after the last bench-method and after the *AfterLastBenchRun* - annotated method
- Executed once per class

2.9 "@SkipBench"

- Will be ignored by perfidix except the method is invoked as a specfic setUp- / tearDown method
- Useful in combination with the *BenchClass* annotation and a specific setUp- / tearDown method of one bench.

3 Example Use Cases

Perfidix 2.1 offers a very flexible usage based on annotation. The examples contains all the same usecase but are different implemented. A compressed file access is compared to a normal file access.

3.1 Example 1

The code in Listing 1,2 and 3 are doing exact the same. The *setUp()* and *tearDown()* methods are invoked before each run of each method. But with Perfidix 2.0 we can do much more.

Listing 1: Perfidix 2.0

```
1  public class SomeAnnoBenchmark {
2          CompressedHandler c;
3          SimpleFileHandler s;
4
5          //setUp, invoked before each run
6          @BeforeEachBenchRun
7          public void setUp() {
8                  c = new CompressedHandler();
9                  s = new SimpleFileHandler();
10         }
11
12         //tearDown, invoked after each run
13         @AfterEachBenchRun
14         public void tearDown() {
15                 c = null;
16                 s = null;
17         }
18
19         //bench Method 1
20         @Bench
21         public void benchCWrite() {
22                 c.write("hello world");
23         }
24
25         //bench Method 2
26         @Bench
27         public void benchSWrite() {
28                 s.write("hello world");
```

```
29          }
30
31          //bench Method 3
32          @Bench
33          public void benchCRead() {
34                  c.read();
35          }
36
37          //bench Method 1
38          @Bench
39          public void benchSRead() {
40                  s.read();
41          }
42  }
```

3.2 Example 2

In Listing 4 you see the usage of specific setUp and tearDown methods. These methods have the same behaviour than methods with the *BeforeBenchRun* annotation.

Listing 2: Perfidix 2.0

```
1  public class SomeSpecificSetUpTearDownBenchmark {
2    CompressedHandler c;
3    SimpleFileHandler s;
4
5    //setUp for benchCRead/benchCWrite. Invoked via @Bench-params
6    public void setUpCompressed() {
7          c = new CompressedHandler();
8    }
9
10   //tearDown for benchCRead/benchCWrite. Invoked via @Bench-params
11   public void tearDownCompressed() {
12         c = null;
13   }
14
15   //setUp for benchSRead/benchSWrite. Invoked via @Bench-params
16   public void setUpSimple() {
17         s = new SimpleFileHandler();
18   }
19
```

```
20    //tearDown for benchSRead/benchSWrite. Invoked via @Bench-params
21    public void tearDownSimple() {
22          s = null;
23    }
24
25    //bench Method 1
26    @Bench(beforeEachBenchRun="setUpCompressed"
27          ,afterEachBenchRun="tearDownCompressed")
28    public void benchCWrite() {
29          c.write("hello world");
30    }
31
32    //bench Method 2
33    @Bench(beforeEachBenchRun ="setUpSimple"
34          , afterEachBenchRun ="tearDownSimple")
35     public void benchSWrite() {
36          s.write("hello world");
37    }
38
39    //bench Method 3
40    @Bench(beforeEachBenchRun ="setUpCompressed"
41          , afterEachBenchRun ="tearDownCompressed")
42    public void benchCRead() {
43          c.read();
44    }
45
46    //bench Method 4
47    @Bench(beforeEachBenchRun ="setUpSimple"
48          , afterEachBenchRun ="tearDownSimple")
49    public void benchSRead() {
50          s.read();
51    }
52
53 }
```

3.3 Example 3

In Listing 5 the same Bench is a little bit modified:
First of all, the class-annotation *BenchClass* with the param *runs* is used. That means that every method which is parameter-free and is not annotated with

a setUp / tearDown annotation, is benched 10 times, except the *benchSWrite* method, which has an extra *Bench* annotation with a run parameter. This method is benched 60 times.
Additional to that, every possible setUp and tearDown method is used in this example. A description is given in the code and in Section 3.

Listing 3: Perfidix 2.0

```
1  @BenchClass(runs=10)
2  public class ClassAnnoBenchmark {
3
4          CompressedHandler c;
5          SimpleFileHandler s;
6
7          String toTest;
8          long testLength;
9
10         //classwide setUp, invoked just one time, just setting the lengt
11         @BeforeBenchClass
12         public void beforeClass() {
13                 Math.abs(testLength = new Random().nextInt(100));
14         }
15
16         //methodWide setUp, invoked just one time per method, building a
17         @BeforeFirstBenchRun
18         public void beforeMethod() {
19                 for(int i = 0; i<testLength; i++) {
20                         toTest = toTest + (char)(new Random().nextInt(Ch
21                 }
22         }
23
24         //normal setUp, invoked one time per method per run, instantiati
25         @BeforeEachBenchRun
26         public void beforeRun() {
27                 c = new CompressedHandler();
28                 s = new SimpleFileHandler();
29         }
30
31         //normal tearDown, invoked one time per method per run, removing
32         @AfterEachBenchRun
33         public void afterRun() {
34                 c = null;
```

```
35                  s = null;
36          }
37
38          //methodWide tearDown, invoked just one time per Method, resetin
39          @AfterLastBenchRun
40          public void afterMethod(){
41                  toTest = null;
42          }
43
44          //classwide tearDown, invoked just one time, reseting the length
45          @AfterBenchClass
46          public void afterClass() {
47                  testLength = -1;
48          }
49
50          //bench 1, invoked because of class-annotation
51          public void benchCWrite() {
52                  c.write("hello world");
53          }
54
55          //bench 2, invoked because of method-annotation
56          @Bench(runs=60)
57          public void benchSWrite() {
58                  s.write("hello world");
59          }
60
61          //bench 3, invoked because of class-annotation
62          public void benchCRead() {
63                  c.read();
64          }
65
66          //bench 4, invoked because of class-annotation
67          public void benchSRead() {
68                  s.read();
69          }
70
71
72  }
```