1DV507, Programming and Data Structures, Spring 2018

The MathSet Competition (VG Exercise)

Deadline: March 18, Preliminary run: March 11. See submission instructions at the bottom of this page for more details.

Problems?
Do not hesitate to ask your teaching assistant at the practical meetings (or Jonas at the lectures) if you have any problems. You can also post a question in the assignment forum in Moodle.

Mathematical Sets

Mathematical sets can be considered as a data structure with the following properties:

- No duplicate elements!
- A binary operation named union (A union B) that produces a new set C containing all the elements in both A and B.
- A binary operation named intersection (A interseection B) that produces a new set C containing all the elements that can be found in both A and B.
- A binary operation named difference (A difference B) that produces a new set C containing all the elements in A that is not contained in B.

Example: Consider two sets:

```
A = {1,2,3,4}, B = {3,4,5,6}
```

Then

```
A union B = {1,2,3,4,5,6}
A intersection B = {3,4}
A difference B = {1,2}
```

The MathSet Competetion Kit

We have put together a MathSet Competition Kit with the following content:

- An interface MathSet defining what we mean by a mathematical set.
- A JUnit test program (TestMathSet) that checks the correctness of a MathSet implementation
- A benchmark program (MathSetBenchmark) that performs (and measures the time of) a large number of MathSet operations
- A MathSet generator class (MathSetGenerator) that can be used to generate different types of MathSets. For example, it comes with a method `MathSet getRandomSet(int size)` that generates a random set of size `size`. The MathSet generator class is used by both the JUnit test program and the benchmark program.
- A simple MathSet implementation named `array_based.MySet` that can be considered as an example.

Competetion Rules

- You are allowed to work in pairs (teams of 1-2 students)
- Each team should provide a MathSet implementation named `your_team_name.MySet`. The class must be named `MySet`, neither the test program nor the benchmark program will work otherwise.
- In addition to implementing the MathSet interface, each class `MySet` must have a public default constructor `MySet()` that creates an empty set, and a constructor `MySet(Collection col)` that creates a new MathSet containing the elements in `Collection col`, and a variable argument constructor `MySet(Object ... elements)` that can handle an arbitrary number of input data.
- To enter the competition you must pass the JUnit test program. Failing ==> you are not qualified to enter the competition.
- The benchmark program is the actual competition. The fastest MathSet implementaion (i.e. shortest time on the benchmark) wins. The parameter `WORKLOAD` inside the benchmark program decides (roughly) the sizes of the sets used in the benchmark. We suggest that you start working with `WORKLOAD = 30`. In the competition we will use `WORKLOAD = 100`.
- The competition deadline is Sunday March 18. However, just to make it a bit more exciting, we would like all teams to make a preliminary submission on Sunday March 11. The benchmark results for each team's preliminary submissions will then be made public on Moodle, giving each team a chance to see their current status, and a chance (one week) to come up with an improved version. Feel free to make even more submissions! We will run and present the results of your contribution as often as we can.
- We will use the JVM arguments `-Xmx2048m -Xms2048m` in the MathSet competition. This puts (a rather high) upper limit on the amount of RAM memory that you are allowed to use.

Details

- The test and benchmark programs are both using the set generator class to generate different types of sets. The import statement (currently `import set_competiton.array_based.MySet`) in the set generator class decides which types of sets that are generated. You must replace this import statement with a new import statement (`your_team_name.MySet`) pointing to your MathSet implementation to run both these programs.
- The print out when running the benchmark program (using our example `array_based.MySet`) looks like this:

```
Sets used in MathSetGenerator: set_competiton.array_based.MySet

Used Workload in Benchmark: 30

        Benchmarking Standard Operations
                Set Creation: 1006
                Equals: 653
                HashCode: 300
                Iterator+Contains: 1434
                ToString: 106
        Standard Operations total: 3499ms, Memory: 22 Mbytes

        Benchmarking Union
                Many small sets: 1367
                A few larger sets: 9281
        Union total: 10648ms, Memory: 7 Mbytes

        Benchmarking Intersection
                Many small sets: 806
                A few larger sets: 6255
        Intersection total: 7061ms, Memory: 13 Mbytes

        Benchmarking Difference
                Many small sets: 827
                A few larger sets: 6367
        Difference total: 7194ms, Memory: 5 Mbytes
```

```
Total time: 28.402 seconds
```

Thus, we have divided the benchmark into four different sections. In the first section we test standard data structure operations like equals(), hashCode(), toString(), contains(), etc. The final three are focused on the MathSet specific operations union(), intersection(), and difference().
Finally, 28 seconds for `WORKLOAD = 30` is actually a very poor result. We expect your implementation to run in less than 15 seconds for `WORKLOAD = 100`.

Submission

The competition deadline is Sunday March 18. However, we would like all teams to make a preliminary submission on Sunday March 11. Your submission should include (the .java files) for the class `your_team_name.MySet` and (possibly) other classes that we need to run your class `MySet`.