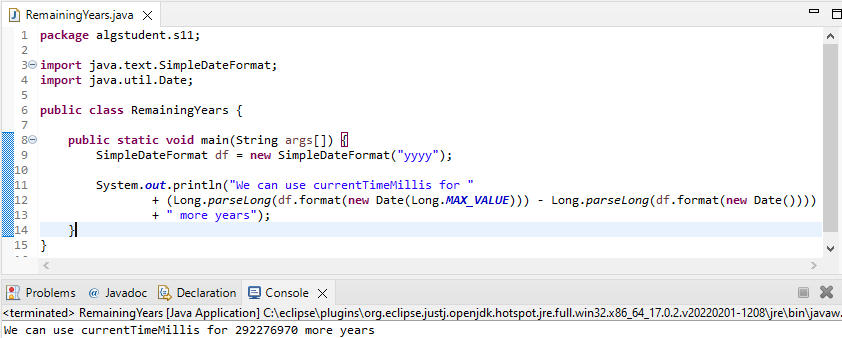
Activity 1. Remaining years for currentTimeMillis()



Activity 2. Input value at which we start to get reliable times

When the time measured is 0 it means that the time elapsed is so low that it is rounded to zero.



(note that many executions were made and this was always the time)

Activity 3. Vector4.java questions

If the complexity is O(n), the time should be doubled.

If the complexity is O(n), like in problem 2, we get 3 and 4 times greater execution times.

This is an example using Vector2.java







The times obtained are the expected ones, as it can clearly be seen.

TABLE1 (times in milliseconds WITHOUT OPTIMIZATION)

|  |  |  |
| --- | --- | --- |
| N | Tsum | Tmaximum |
| 10000 | 63 | 80 |
| 20000 | 129 | 201 |
| 40000 | 256 | 435 |
| 80000 | 473 | 648 |
| 160000 | 943 | 1204 |
| 320000 | 1847 | 23390 |
| 640000 | 3740 | 4900 |
| 1280000 | 7683 | 9046 |
| 2560000 | 14893 | 18142 |
| 5120000 | 30272 | 36279 |
| 10240000 | 59818 | OoT |
| 20480000 | OoT | OoT |
| 40960000 | OoT | OoT |
| 81920000 | OoT | OoT |

TABLE2 (times in milliseconds WITHOUT OPTIMIZATION)

|  |  |  |
| --- | --- | --- |
| N | Tmatches1 | Tmatches2 |
| 10000 | 689 | 0 |
| 20000 | 2797 | 0 |
| 40000 | 11117 | 0 |
| 80000 | 45410 | 1 |
| 160000 | OoT | 2 |
| 320000 | OoT | 7 |
| 640000 | OoT | 7 |
| 1280000 | OoT | 14 |
| 2560000 | OoT | 41 |
| 5120000 | OoT | 68 |
| 10240000 | OoT | 117 |
| 20480000 | OoT | 253 |
| 40960000 | OoT | 500 |
| 81920000 | OoT | 989 |

Note that the values obtained for matches2 are not in a ‘reliable’ range, but since this is a comparison, the same problem size was used.

For the sum function, it can be seen that, since the time complexity of the algorithm is O(n), each time the value of n is doubled, the execution time is approximately double as well.

For the maximum function, it can be seen that, since the time complexity of the algorithm is O(n) the time spent in the execution is approximately doubled each time n is doubled.

Regarding matches1, its complexity is O(n2), if we make the calculation for the first two times:

T1 / n1^2 = k ; T2 / n1^2 = k

T2 = T1 \* n2^2 / n1^2 = 689 \* 20000^2/10000^2 = 2756.

As it can be seen, the values are close to the expected ones.

For matches2, its complexity is O(n), so it is expected that the times are ridiculously low in comparison to the ones for matches1

All the measurements were done in a computer with a Ryzen5 3400g CPU and 16GB of RAM