INF346: Hands-On Session
Lock-Based List-Based Set Implementations

The goal of this exercise session is to get familiar with fine-grained locking schemes on the example of a sorted linked list used to implement a set abstraction.

We are going to use the `synchrobench` benchmark (`https://github.com/gramoli/synchrobench`) to study the performance of three set implementations:

- Coarse-grained locking
- Hand-Over-Hand locking (already present as `linkedlists.lockbased.HandOverHandListIntSet.java`)
- Optimistic list with wait-free traversal and validation
- Lazy Linked List by Heller et al. (already present as `linkedlists.lockbased.LazyLinkedListSortedSet.java`)

For the coarse-grained and optimistic algorithms refer for the slides or Chapter 9 of the book by Herlihy and Shavit (`https://www.dawsonera.com/abstract/9780123977953`).

In the simplest setting, `synchrobench` allows you to define a workload varying the following parameters: number of threads, size of the list, range of the list values, duration of the experiment, and fraction of updates (adds and removes, equally shared). In the simulated runs, every update picks a random value in the range uniformly at random.

The collected statistics contains the throughput (the number of complete ops per second), as well as the fractions of successful removes, adds and contains.

Check INSTALL for instructions on adding new algorithms to the `synchrobench` library (you may also use `eclipse` with `build.xml` to do this), and README.md for the parameters of the simulations.

A sample command to run `synchrobench` with a lock-based linked list algorithm ALG on 10 threads, update ratio 10, list size 1024, range 0-2047, and duration 3000ms is:

```
> java -cp bin contention.benchmark.Test -b linkedlists.lockbased.ALG -d 3000 -t 10 -u
  10 -i 1024 -r 2048
```

(1) Implement coarse-grained and optimistic list-based sets. You may take Hand-Over-Hand locking (`src/linkedlists/lockbased/HandOverHandListIntSet.java`) as a basis. Make sure that your new classes implement `AbstractCompositionalIntSet` and export the right interface.

(2) Run the three algorithms: coarse-grained, optimistic and lazy, through `synchrobench` with the following parameters:

- Number of threads: 1, 2, 4, 6, 8
- Update ratios: 0, 10, 100
- List sizes: 100, 1k, 10k (choose the ranges of the lists twice as big, to maintain the expected list size constant)
- For the other parameters, specify `-W 0 -d 2000` (no "warmup", duration 2000ms).

(3) Prepare a short summary document. Prepare a graph depicting throughput depending of the number of threads for each of the three algorithms for some representative list size and update ratio. Also, prepare a graph depicting throughput depending on the number of threads for each of the algorithm, varying the update ratio, for list size 1k.

Each graph shows throughput exhibited by the three algorithms for each number of threads, i.e., it must contain three curves (for each algorithm) of 5 points each. You may use `gnuplot` or any other plotting program of your choice.

Explain the forms of the curves and their relations to each other.