Concurrent Object Oriented Languages

Measure Performance on MTL

https://wiki.cse.yorku.ca/course/6490A

Test

Once your harness to measure performance is complete, first run it on a machine such as navy.cse.yorku.ca to test it.

- Run the harness for the different implementations of a stack.
- Create a dummy stack with push and pop that do nothing and measure its performance (this measures the overhead of the harness).

Heap

The JVM accepts two arguments, `-Xms` and `-Xmx`, which specify minimum and maximum heap sizes, respectively.

```
java -Xms1G -Xmx1G MyProgram
```

If these parameters are not specified, the JVM can resize the heap.

- In practice, JVMs frequently resize the heap.
- Since this may occur in some trials, and not in others, it is best to control this variable.

Garbage collection

Check whether garbage collection takes place.

Use

```
java -Xloggc:gc.log MyProgram
```

Hot-spot compilation

Java compilation occurs throughout an execution (but mostly in the first few seconds).

This is important when comparing algorithms.

- Some algorithms take longer to compile, and stay in a slow, interpreted state for longer.
- This reduces their measured performance compared to faster compiling algorithms.

One solution is to discard the first few trials of each experiment.

Use 64 bit JVM

Use a 64-bit JVM on a 64-bit machine.

```
java -d64 MyProgram
```

Run JVM in server mode

The server mode allows for aggressive optimizations.

```
java -server MyProgram
```

Run each experiment in seperate JVM

- Multiple experiments run in the same JVM are not statistically independent.
- It is not enough to simply run garbage collection between each pair of experiments.
- The internal state of the memory allocator, garbage collector and Hot-spot compiler are largely inaccessible.