Week 11: Lab 8

CS2030S Lab 16B

Chrysline & Alissa

Overview

1. Recap
2. Lab 6 Feedback
3. Lab 7 Brief

1: Recap

Unit 34 - Streams

Important Methods (can be found in Oracle docs)

Making a new stream:

- Stream.of(T ...)
- Stream.iterate(T, UnaryOperator<T>)
- Stream.generate(Supplier<T>)
- collection.stream()

Interacting with the items in the stream

- map
- flatmap
- filter
- reduce

Unit 34 - Streams

Important Methods (can be found in Oracle docs)

Using the items in the stream

- noneMatch
- anyMatch
- allMatch
- count
- forEach
- toArray
- limit

Unit 36 - Monads

- Left identity law
  - Monad.of(x).flatMap(x -> f(x)) must be the same as f(x)
  - Meaning:
    1. Create a new item,
    2. Apply some function to the value contained inside using flatMap
    3. (Function f will transform x into the same type Monad)
    4. The result of this should be the same as applying the function F on x directly

- Right identity law
  - monad.flatMap(x -> Monad.of(x)) must be the same as monad
  - Meaning: factory method should not be doing anything “extra”

Unit 36 - Monads

- Associative Law
  - monad.flatMap(x -> f(x)).flatMap(x -> g(x)) must be the same as monad.flatMap(x -> f(x).flatMap(y -> g(y)))
  - Meaning:
    - Applying consecutively = applying all at once

Unit 36 - Monads (Functors)

- Preserving Identity:
  - functor.map(x -> x) is the same as functor
- Preserving composition:
  - functor.map(x -> f(x)).map(x -> g(x)) is the same as functor.map(x -> g(f(x)))
- Something can be both a monad and a functor at the same time.

Unit 37 - Parallel Streams (Concurrency)

- Concurrency: Illusion of running tasks at the same time, but actually simply switching between tasks
- Parallelism: Truly running tasks at the same time.
- Qn: Are all parallel programs concurrent?
- Qn: Are all concurrent programs parallel?

Unit 37 - Parallel Streams (Threads)

- How do we execute a single process in parallel?
  - Split it up into “sub-processes” called threads.
  - Execute each thread independently and at the same time.
  - Having threads does not guarantee parallel execution. It only allows parallel execution to occur.

Unit 37 - Parallel Streams

- To make a stream parallel, simply do .parallel() on an existing stream.
- What stream actions can be parallelised?
  - Any action that does not modify the stream.
- Initially, x = 0. There are 2 processes that do x = x + 1. What are the possible values of x in the end?

Unit 37 - Parallel Streams (Performance)

- Creating new threads involves more than simply splitting up tasks.
- Other work involved like assigning information to each thread. (More in CS2106)
- Hence, there is overhead in creating threads.
- Creating threads will not ALWAYS better performance. It depends on how complicated the process is.

Unit 37 - Parallel Streams (Ordering)

- Streams created from collection.stream() may have an inherent ordering
  - Unordered collections eg Set
  - Ordered collections eg ArrayList
  - Streams created from eg iterate will also be ordered.
- Some stream operations respect ordering by default. To ignore ordering, do .unordered()
- This helps to further improve the performance for parallel streams.

That’s all for today! Thanks for coming!

Feedback