Week 10: Lab 7

CS2030S Lab 16B

Chrysline & Alissa

Overview

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

1: Recap

Infinite List

- With Lazy evaluation we can build infinite list
  - Any eager-evaluation-based solution will just run in an infinite loop if the list is infinitely long
- Now we can do:

```
 public static <T> InfiniteList<T> generate(Producer<? extends T> producer) {
   return new InfiniteList<T>(producer,
       () -> InfiniteList.generate(producer));
}
```

If you have a Lazy class, you can also do:

```
 public static <T> InfiniteList<T> generate(Producer<? extends T> producer) {
   return new InfiniteList<T>(Lazy.from(producer),
      Lazy.from( () -> InfiniteList.generate(producer)));
}
```

Stream

- Stream is Java's implementation of infinite list

- A typical way of writing code that operates on streams is to chain a series of intermediate operations together, ending with a terminal operation.
    - An intermediate operation on stream returns another Stream.
      eg: map, filter, flatMap

      Intermediate operations are lazy and do not cause the stream to be evaluated.

- When using terminal operation (eg: forEach) be careful of the possibility of infinite loop. Terminal operations on stream will evaluate the stream.
    - Need to convert infinite stream to finite with operations like:
      limit, takeWhile

Stream

- A stream can only be operated on once.

- Some examples of stream usage:
  (noneMatch returns true if either no elements of the stream match the provided predicate or the stream is empty, otherwise false)

```
boolean isPrime(int x) {
  for (int i = 2; i <= x-1; i++) {
    if (x % i == 0) {
      return false;
    }
  }
  return true;
}
```

  Can be rewritten into a 1-liner:

```
boolean isPrime(int x) {
  return IntStream.range(2, x)
      .noneMatch(i -> x % i == 0);
}
```

Stream

- Now what if we want to print the first 500 prime integers using Stream? (We can call the previous isPrime() function to determine if an integer is a prime or not)

  Hint: use filter, limit and forEach

Stream

Answer:

```
IntStream.iterate(2, x -> x+1)
   .filter(x -> isPrime(x))
   .limit(500)
   .forEach(System.out::println);
```

That’s all for today! Thanks for coming!

Feedback