Week 11 Lab Session

CS2030S AY21/22 Semester 2

Lab 14B

Yan Xiaozhi (David)
@david_eom
yan_xiaozhi@u.nus.edu

31 Mar 2022

Admin

- Contact tracing & QR code
- PE 2: 9 Apr (Sat), 9am - 12pm
- Do start on your Lab 7!
- Today’s schedule:
  - One hour to solve a mock PE question
  - Briefly go though the solution

Mock PE 2

- Accept via GitHub Classroom
- Run `~cs2030s/get-pe2-2021s2`
- Real question from AY20/21 Sem 2 PE 2
- Spend the next hour or so to attempt Q2!
  - 2 hours, 40 marks in total, 15 marks for Q2, technically 45 min

Mock PE 2 Q2 Review

pointStream

- `static <T> Stream<T> iterate(T seed, UnaryOperator<T> f)`

- ```
  public static Stream<Point> pointStream(
            Point point, Function<Point, Point> f) {
    return Stream.iterate(point, p -> f.apply(p));
  }
  ```

- ```
    return Stream.iterate(point, f::apply);
  ```

generateGrid

- `Stream<T> limit(long maxSize)`
- `<R> Stream<R> flatMap(Function<? super T, ? extends Stream<? extends R>> mapper)`
-

```
public static Stream<Point> generateGrid(
        Point point, int n) {
  return
    pointStream(point, p -> new Point(p.getX() + 1, p.getY()))
    .flatMap(
      p -> pointStream(p, x -> new Point(x.getX(), x.getY() + 1))
      .limit(n)
    )
    .limit(n);
}
```

concentricCircle

- `static <T> Stream<T> iterate(T seed, UnaryOperator<T> f)`

- ```
  public static Stream<Circle> concentricCircles(
      Circle circle, Function<Double, Double> f) {
    return Stream.iterate(
      circle,
      c -> new Circle(c.getCenter(), f.apply(c.getRadius())))
    );
  }
  ```
  ```

pointStreamFromCircle

- ```
  <R> Stream<R> map(
      Function<? super T, ? extends R> mapper
  )
  ```

- ```
  public static Stream<Point> pointStreamFromCircle(
      Stream<Circle> circles) {
    return circles.map(c -> c.getCenter());
  }
  ```

estimatePi

- `static <T> Stream<T> generate(Supplier<? extends T> s)`
- `Stream<T> limit(long maxSize)`
- `Stream<T> filter(Predicate<? super T> predicate)`
- `long count()`
-
```
public static double estimatePi(
    Circle c, Supplier<RandomPoint> supplier, int k) {
  return Stream
    .generate(supplier)
    .limit(k)
    .filter(x -> c.contains(x))
    .count() * 4.0 / k;
}
```

About AY21/22 Sem 2 PE 2

- Stats for last year's PE2:

Question	Number of Correct Answers
`pointStream`	382
`generateGrid`	135
`concentricCircle`	279
`pointStreamFromCircle`	290
`estimatePi`	100

Good luck for PE2! 🍀