Streams

Creating Stream Source

Stream Creation Methods

Method	Finite?	Description
`Stream.empty()`	Yes	Creates `Stream` with zero elements
`Stream.of(varargs)`	Yes	Creates `Stream` with elements listed in `varargs`
`coll.stream()`	Yes	Creates `Stream` from `Collection`
`coll.parallelStream()`	Yes	Creates parallel `Stream` from `Collection`
`Stream.generate(supplier)`	No	Creates `Stream` by calling `Suppler` for each element upon request
`Stream.iterate(seed, unaryOperator)`	No	Creates `Stream` by using seed for first element and then calls `UnaryOperator` for each subsequent element
`Stream.iterate(seed, predicate, unaryOperator)`	Depends	Same as before, but stops if `Predicate` returns `false`

```
// creating finite streams
Stream<String> empty = Stream.empty();
Stream<Integer> singleElement = Stream.of(1);
Stream<Integer> fromArray = Stream.of(1, 2, 3);

// converting Collection to stream
var list = List.of("a", "b", "c");
Stream<String> fromList = list.stream();

// parallel stream (operations are done in parallel rather then in sequence)
Stream<String> fromListParallel = list.parallelStream();
```

```
// creating infinite streams

Stream<Double> randoms = Stream.generate(Math::random);

Stream<Integer> oddNumbers = Stream.iterate(1, n -> n + 2);

// these streams are infinite, they generate values ad infinitum

randoms.forEach(System.out::println);

  => program prints random numbers until you kill it


// operations like limit() can turn infinite stream to finite one


// create odd numbers less than 50

Stream<Integer> oddNumbersUnder50 = Stream.iterate(1, n -> n < 50, n -> n + 2);
```

supplier

seed

unary operator

predicate

(stream stops when returns false)