

ORACLE®

Lesson 3-2: Finite and Infinite Streams

Dealing With The Indeterminate

Imperative Java

- How to continue processing when we can't predict for how long?

```
while (true) {  
    doSomeProcessing();  
  
    if (someCriteriaIsTrue())  
        break;  
  
    // Loop repeats indefinitely  
}
```


Using Infinite Streams

Making The Stream Finite


- Terminate the stream when an element is read from the input stream
 - `findFirst()`
 - `findAny()`

```
OptionalInt r = Random.ints()  
    .filter(i -> i > 256)  
    .findFirst();
```

Infinite stream of
random integers



stream terminates when a number
greater than 256 is encountered



Using Infinite Streams

Keeping It Infinite

- Sometimes we need to continue to use a stream indefinitely
- What terminal operation should we use for this?
 - Use `forEach()`
 - This consumes the element from the stream
 - But does not terminate it

Using Infinite Streams

Infinite Example

- Reading temperature from a serial sensor
 - Converting from fahrenheit to celcius, removing F
 - Notifying a listener of changes if registered

```
thermalReader.lines()  
  .mapToDouble(s ->  
    Double.parseDouble(s.substring(0, s.length() - 1)))  
  .map(t -> ((t - 32) * 5 / 9))  
  .filter(t -> !currentTemperature.equals(t))  
  .peek(t -> listener.ifPresent(l -> l.temperatureChanged(t)))  
  .forEach(t -> currentTemperature.set(t));
```

Section 2

Summary

- Streams can be infinite as well as finite
- There is no concept of 'breaking' out of a stream
- Use the appropriate terminal operation to stop processing
- Or use the infinite stream infinitely

ORACLE®