CSD-4464 Java EE

Class 6: Streams

Streams

- Stream is a new abstract layer introduced in Java 8.
- Stream represents a sequence of objects from a source, which supports aggregate operations
- Streams are lazy; computation on the source data is only performed when the terminal operation is initiated, and source elements are consumed only as needed.

Streams cont.

- Streams support many transformation methods, similar to the Optional class (e.x map() filter() flatMap())
- Stream() vs ParallelStream()
- You can create streams from the Stream Class, or by calling stream() on a collection

Example

myList.stream() //you just create a stream containing the elements of your list

ParallelStream()

- Multithreaded code tends to be error prone and very difficult to read, Java8 attempts to solve this with ParallelStreams, a simple way to apply functions to datasets in parallel over sequentially
- Except for operations identified as explicitly nondeterministic, such as findAny(), whether a stream executes sequentially or in parallel should not change the result of the computation.
- Default thread count may vary from JVM to JVM, generally threads are pulled from ForkJoinPool.commonPool()

Streams.findFirst() + findAny()

- Returns an Optional describing an element in the stream, or an empty optional if the stream is empty
- findAny() returns any value it finds in the stream
- findFirst() returns the first value it finds in the stream
- Both are short-circuiting terminal operations

.flatMap()

- Used for 1 to many relations
- Returns a stream containing the values from the mapped streams

e.x A school object, has a list of Class objects, which have a list of Student objects

```
School.getClasses().stream()
.flatMap(class -> class.getStudents.stream())//returns Stream<student>
```

.limit() .skip() .[any/all/none]Match()

- .limit(n) sets the max number of elements to flow through the stream
- .skip(n) skips a number of elements in the streams
- .anyMatch(Predicate) returns true if any values in the stream matches
- .allMatch(Predicate) returns true if all values in the stream matches
- .noneMatch(Predicate) returns true if no values in the stream matches

Lambton College

.iterate() .generate()

- Returns an infinite stream
- .iterate() generates values similar to a for-loop

Syntax - Stream.iterate(0, n -> n + 1).limit(10)
for (int I = 0; I < 10; I ++)

- .generate() generates values from a provided supplier

Syntax - Stream.generate(new Random()::nextInt)