Week 14 - Stream API

⚠ This page is a draft, treat it accordingly,

Topics covered in this week

- stream creation
- stream transformations (filter, map, flatMap, stateful transformations)
- collectors
- parallel streams
- stream API with Lambda expressions
- reactive streams (java.util.concurrent.Flow)
- Optional

Reading material

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html (introduction to streams, streams vs collections)
- https://www.oracle.com/technical-resources/articles/java/architect-streams-pt2.html (collect, flatMap)
- https://www.baeldung.com/java-8-streams (stream creation, referencing a stream, pipelines, collectors)
- https://stackify.com/streams-guide-java-8/ (stream transformations/operations - forEach, map, collect, filter, findFirst, toArray, flatMap, etc.)
- https://docs.oracle.com/javase/tutorial/collections/streams/parallelism.html (parallel streams)
- https://javapapers.com/java/java-stream-api/ (stateless/stateful stream transformations)
- https://www.oracle.com/technical-resources/articles/java/java8-optional.html (optional)
- https://www.baeldung.com/java-optional (optional, examples)
- https://blog.softwaremill.com/how-not-to-use-reactive-streams-in-java-9-7a39ea9c2cb3 (reactive streams, examples)

Homework

Difficulty	Problem	Notes
EASY	Using IntStream.range() method find all the palindromic numbers within the range of 1000-10000 and count how many of them are odd and even numbers, writing the result on the standard output. Use only the stream pipeline to obtain this result, but you may use method references for complex lambda expressions (e.g. to check if a number is a palindrom). **Note**: A palindromic number reads the same both ways. E.g. 12321 is a palindrom.	Example solution: `import java.util.stream.*;`

```
private static int getReversed(int nr) {
    int reversed = 0;


int quotient = nr;

while (quotient > 0) {

int remainder = quotient % 10;

reversed = reversed * 10 + remainder;

quotient = (int) Math.floor(quotient / 10);

    }
    return reversed;
}
```

		IntStream.range(1000, 10000).filter(nr -> nr == getReversed(nr)).boxed()
		.collect(Collectors.groupingBy(nr -> nr % 2 == 0, Collectors.counting()))
		
		.forEach((even, count) -> {
		if (even) System.out.println("Even: " + count);
		else System.out.println("Odd: " + count);
		});

EASY		
EASY		