

Java Stream API — PDF

```

#####, ## ##### ##### java.util.stream: #####, #####
#####, ##### ##### a ##### #####.

```

Stream API?

Stream API —  API  Java ( Java 8).

THE INFORMATION CONTAINED HEREIN IS UNCLASSIFIED EXCEPT WHERE SHOWN OTHERWISE:
DATE 08-26-2019 BY 60322 UCBAW/BJS

Streams  (Streams, Streams, I/O, Streams).

[illegible][illegible]

XXXXXXXXXX XXXXXXXXXX (XXXXXXXXXXXX)

```

##### — ##### ##### ##### #####. ##### ##### `List`
`List`.

```

□ □ □ □ □ □ □ □ □ □ □ □

■■■■■ :

■■■■■■■■ stream

```
List<String> names = List.of("Anna", "Bob", "Charlie"); Stream<String> s = names.stream();
Stream<String> pStream = names.parallelStream();
```

■■■■■■■■■■ ■ map

```
List<String> filtered = names.stream().filter(n -> n.length() > 3)
    .map(String::toUpperCase).collect(Collectors.toList());
```

FlatMap

```
List<List<Integer>> lists = List.of(List.of(1,2), List.of(3,4)); List<Integer> flat = lists.stream()
    .flatMap(List::stream)
    .collect(Collectors.toList());
```

```
Map<String, Long> counts = names.stream()                .collect(Collectors.groupingBy(Function.identity(),
Collectors.counting()));
```

```
int sum = List.of(1,2,3,4).stream().reduce(0, Integer::sum);
```

```
int total = largeList.parallelStream().mapToInt(Integer::intValue).sum(); // ■■■■:
```

■■■■■■■■■■ ■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■■■■■■ ■■■■ ■■■■-■■■■■■■■■■

```
Optional<Person> oldest = people.stream().max(Comparator.comparingInt(Person::getAge));
oldest.ifPresent(p -> System.out.println(p.getName()));
```

```
- **parallelStream***: overhead of GC.
- **GC***: GC.
- **stream-parallel***: stream-parallel (forEach parallel pipeline).
- **Boxing/unboxing***: boxing/unboxing of primitive types.
- **IntStream/LongStream/DoubleStream***: mapToInt, mapToLong, mapToDouble.
- **toList***: toList() method of Stream, toList() method of ParallelStream.
- **toArray***: toArray() method of Stream, toArray() method of ParallelStream.
```

- **Parallelism** (number of threads): `parallelism(10)`
- **Execution mode** (forkJoin, sequential, ordered): `forkJoin()`
- **Backpressure** (control flow): `onBackpressureBuffer()`
- **Timeouts** (cancel after): `timeout(10, TimeUnit.SECONDS)`
- **Logging** (debug, info, warn, error): `logger.debug()`
- **Configuration** (properties, environment variables): `System.getProperty()`
- **Testing** (unit tests, integration tests): `JUnit`, `TestNG`
- **Deployment** (JAR, WAR, Docker): `mvn package`, `docker build`
- **Monitoring** (metrics, logs): `Prometheus`, `ELK`
- **Security** (authentication, authorization): `OAuth2`, `JWT`
- **Compliance** (GDPR, HIPAA): `GDPR`, `HIPAA`
- **Documentation** (README, API docs): `README.md`, `Swagger`
- **Deployment** (CI/CD): `Jenkins`, `GitLab CI`
- **Testing** (unit tests, integration tests): `JUnit`, `TestNG`
- **Deployment** (JAR, WAR, Docker): `mvn package`, `docker build`
- **Monitoring** (metrics, logs): `Prometheus`, `ELK`
- **Security** (authentication, authorization): `OAuth2`, `JWT`
- **Compliance** (GDPR, HIPAA): `GDPR`, `HIPAA`
- **Documentation** (README, API docs): `README.md`, `Swagger`
- **Deployment** (CI/CD): `Jenkins`, `GitLab CI`

	
	 parallelStream  , 
	 : overhead parallelStream;  
ConcurrentModificationException	 .

■■■■■■■■■■ ■■■■■■■■■■ (■■■■■■)

```
// ■■■■■■: ■■■■■■■■ ■■■■ ■ ■■■■■■■ Map<String, Long> freq = words.stream() .flatMap(w
Arrays.stream(w.split("\\s+"))) .map(String::toLowerCase)
.collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));
```