Параллельные потоки данных

Создание

Для любой коллекции метод parallelStream()

```
String contents = new String(Files.readAllBytes(Paths.get(“alise.txt”)),
                                        StandardCharsets.UTF_8);
List<String> words = Arrays.asList(contents.split(“\\PL+”));
Stream <String> parallelWords = words.parallelStream();
```

Для последовательного потока методом parallel()

```
Stream <String> parallelWords = Stream.of(wordsArray).parallel();
```

Требования

Получаемый при параллельном выполнении результат должен быть таким же, как и при последовательном выполнении

Источник данных должен легко разбиваться на части

Операция не должна иметь состояния

Операция должна быть ассоциативной

Скорость работы

```
long start = System.currentTimeMillis();  // System.nanoTime()

long finish = System.currentTimeMillis();

long elapsed = finish - start;


//Java 8

Instant start = Instant.now();

Instant finish = Instant.now();

long elapsed = Duration.between(start, finish).toMillis();
```

От чего зависит

От количества ядер

От объема данных

От затрат на организацию параллельного пула потоков

От требования упорядоченности

От длительности самой операции, выполняемой в потоке

Сравнение

```
long count = words.stream()
                    .filter(w ->w.length() >12)
                    .count();
```

```
long count = words.parallelStream()
                    .filter(w ->w.length() >12)
                    .count();
```

Чтобы замедлить

```
long count = words.stream()
                    .filter(w -> {
                                Thread.sleep(100);
                                return (w.length() >12);
                                })
                    .count();
```

Гонка потоков

```
int shortWords = new int[12];

words.parallelStream()
  .forEach( s-> {
        if (s.length() <12) shortWords[s.length()]++;
  });
  System.out.println(Arrays.toString(shortWords));
```

```
Map<Integer, Long> shortWords =

words.parallelStream
        .filter (s->s.length() <12)
         .collect( grouppingBy(
                String::length, counting()));
```