Seminar 8
week 8 (21-27 November 2017)

NOTE: Starting from week 10 (4-8 December 2017) all groups will have the same topics for seminars and laboratories. The week will start on Monday and end on Friday. Only the groups from Tuesday and Wednesday will have the Seminar 9 and Laboratory 9 due to National Day Holiday.

1. Questions from **Lab-Assignment 5 from Laboratory 7**.
2. Discussion of the **Lab-Assignment 6 from Laboratory 8**.
3. Discussion of the asynchronous execution of tasks by using ExecutorService.
Please use the following examples taken from the tutorial :
http://winterbe.com/posts/2015/04/07/java8-concurrency-tutorial-thread-executor-examples/

Example 1:

```
ExecutorService executor = Executors.newSingleThreadExecutor();
executor.submit(() -> {
    String threadName = Thread.currentThread().getName();
    System.out.println("Hello " + threadName);
});

// => Hello pool-1-thread-1
```

Example 2:

```
try {
    System.out.println("attempt to shutdown executor");
    executor.shutdown();
    executor.awaitTermination(5, TimeUnit.SECONDS);
}
catch (InterruptedException e) {
    System.err.println("tasks interrupted");
}
finally {
    if (!executor.isTerminated()) {
        System.err.println("cancel non-finished tasks");
    }
    executor.shutdownNow();
    System.out.println("shutdown finished");
}
```

Example 3:

```
Callable<Integer> task = () -> {
    try {
        TimeUnit.SECONDS.sleep(1);
        return 123;
    }
    catch (InterruptedException e) {
        throw new IllegalStateException("task interrupted", e);
    }
};
ExecutorService executor = Executors.newFixedThreadPool(1);
Future<Integer> future = executor.submit(task);
```

```
System.out.println("future done? " + future.isDone());
Integer result = future.get();
System.out.println("future done? " + future.isDone());
System.out.print("result: " + result);
```

Example 4:

```
ExecutorService executor = Executors.newWorkStealingPool();

List<Callable<String>> callables = Arrays.asList(
        () -> "task1",
        () -> "task2",
        () -> "task3");

executor.invokeAll(callables)
    .stream()
    .map(future -> {
        try {
            return future.get();
        }
        catch (Exception e) {
            throw new IllegalStateException(e);
        }
    })
    .forEach(System.out::println);
```