Week 12 - Concurrency

This page is a draft, treat it accordingly,

Topics covered in this week

- thread vs. process
- thread synchronisation strategies
- thread pools and executors
- deadlocks, race conditions
- thread dump and memory dump
- concurrent collections

Reading material

- Thinking In Java - Concurrecy 797-931
- http://tutorials.jenkov.com/java-concurrency/index.html
- https://product.hubspot.com/blog/understanding-thread-dumps
- https://dzone.com/articles/how-to-take-thread-dumps-7-options
- https://www.baeldung.com/java-heap-dump-capture

Homework

Difficulty	Problem	Notes
MEDIUM	We know that if we have an application that is deployed in a cluster that has more than one node it is quite difficult to retrieve/search through the logs from these different nodes. A solution to this problem would be to have one single server that consumes logging messages (storing them in one repository). Each node from the cluster generates new logging messages. New message can be added by the server in its queue only if there are less than 10 messages in the queue. The client should wait until the server can consume it. Simulate this solution with threads.	We don’t really want a central physical server try to simulate the problem with threads instead. Implementation should be with wait/notify.
MEDIUM	Let's make a thread race program, where a *ThreadRace* class will create an instance of 10 *ThreadRaceCompetitor*'s. Then it'll run them all. There will be a results class called *ThreadRaceContext*, which will keep the scorecard of who finished in which place. When a *ThreadRaceCompetitor* finishes the race, it will have to inform the *ThreadRaceContext* his race *number* (an id), and that it's done. The *ThreadRaceContext* will list the final rankings to the console. Who'll arrive first? Make your bets!	
MEDIUM	Now let's try to make the *ThreadRace* even more interesting: let's make it a *ThreadRelayRace*, in which the competition will occur between 10 *ThreadRelayRaceTeam*'s, each of them composed of 4 *ThreadCompetitor*'s. Now the *ThreadCompetitors* will have a thread number and a team name, and the *ThreadRaceContext* will inform when a *ThreadRaceCompetitor* from a team finished, and then the whole team finishes, informing the results of the team's positions.	Commit the solution into a separate folder than the previous problem.