**Note: all tasks should be developed with Java 8 (at least)**

**Task 1. (20 points)**

Design and implement code that will introduce:

1. java.lang.OutOfMemoryError: Java heap space. You can use different data structures. Do not tune heap size.
2. java.lang.OutOfMemoryError: Java heap space. Create big objects continuously and make them stay in memory. Do not use arrays or collections.
3. java.lang.OutOfMemoryError: Metaspace. Load classes continuously and make them stay in memory.
4. java.lang.StackOverflowError. Use recursive methods. Don’t tune stack size.
5. java.lang.StackOverflowError. Do not use recursive methods. Don’t tune stack size.

**Task 2. (15 points) Tune regions and monitor with jdk monitoring tools**

**Note: Try Java 8 jdk tools**

Take your pet or current project app for the next exercises with JVM settings via flags and console utilities.

1. Heap should be divided to Old and Young generations of equal size, where areas of Young generation are in following ratio: 3-Eden, 1-S0, 1-S1
2. Heap should be of 512MB and divided to Old and Young generations of equal size, where areas of Young generation are in following ratio: 2-Eden, 1-S0, 1-S1
3. Heap should be of 512MB with Old generation of 128MB and Young generation with areas in following ratio: 3-Eden, 1-S0, 1-S1
4. Heap should be divided to Old generation of 128MB and Young generations with areas: Eden of 384MB, S0 of 128MB, S1 of 128MB

After JVM start make screenshots with jdk monitoring tools with JVM flags for each point.

Use this [link](https://docs.oracle.com/javase/8/docs/technotes/tools/index.html) to find appropriate tool. Check yourself with JVisualVM.

**Task 3. (10 points) MAT for heap dump analysis**

1. Write simple application that consumes memory or take your mentoring pet application
2. Catch java.lang.OutOfMemoryError: Java heap space
3. Make heap dump (-XX:+HeapDumpOnOutOfMemoryError or via visualvm)
4. With Memory Analyzer Tool generate report that shows which classes are consuming the memory

Share results with mentor in essay format

**Task 4. (15 points) Classloading**

1. Create a new abstract class (or interface) Animal with methods: "Play", "Voice".
2. Inherit Cat and Dog from the Animal class.
3. Compile the classes.
4. Create your own ClassLoader by extension of the standard ClassLoader.
5. Load the classes Cat and Dog into an array or collection Animals by your own class loader and run the methods "Play", "Voice".
6. Output data to console by log4j library (logger).

Note: Methods should just print a text message.

**Task 5. (5 points) String Deduplication**

Run your pet/project application with enabled and disabled String Deduplication feature.

Analyze stat info to say final word to use or not to use this feature on your project. Discuss it with mentor.

**Possible penalties**

* also -1 for each task if mentee ignores Java Code Conventions
* - (total points / amount of subtask in subtask list) for each missed subtask in each task