Università della Svizzera italiana **Advanced Java Programming**
Assignment 4 **Fall 2022**
Dr. Andrea Rosà, Prof. Walter Binder, Mariano Marciello

Instructions

- This assignment consists of 60 points and counts 6% towards your overall grade.
- The deadline of the assignment will be published on iCorsi. The deadline is strict.
- You are supposed to submit a single zip file. The zip should contain both textual and code solutions. Textual solutions should be included in a single PDF. All pages of the PDF must have your group name in the upper right corner. Code solutions must consist of source code (no JARs or compiled classes). Use the following pattern for naming your submission files: $< group - name > .zip$.
- Unless otherwise noted, the assignment considers **Java OpenJDK 17 (HotSpot) VM**. For simplicity, code snippets may not always contain the full code (e.g. imports or called methods might be omitted). If not mentioned otherwise, you can assume that the snippets compile, the hidden code is implemented correctly, and methods do what their name suggests, without throwing any runtime exceptions.
- You will not receive full points for solutions which produce the correct result but don't follow fundamental programming best practices or documented contracts. Further, don't consider any solutions that involve the use of Java reflection.

Exercise A: Concurrency (3 Points)

Please answer the question below and provide a short and precise explanation.

Consider the following code snippet:

```
1  public class ExerciseA {
2
3      static public class ThreadExample {
4          public static void main(String[] args) {
5
6              Thread t1 = new Thread(() -> System.out.print("t1 "));
7              Thread t2 = new Thread(() -> System.out.print("t2 "));
8              Thread t3 = new Thread(() -> System.out.print("t3 "));
9
10             t1.start();
11             t2.start();
12             t3.start();
13         }
14     }
15 }
```

List all the possible textual outputs of the above code snippet.

Exercise B: Concurrency (10 Points)

Please answer the questions below and provide a short and precise explanation.

Consider the following code snippet:

```
1  public class ExerciseB {
2
3      public static class Counter {
4          private int count = 0;
5
6          synchronized void incrementCounter() {
7              count++;
8          }
9
10         public int getCounter() {
11             return count;
12         }
13     }
14
15     public static void main(String[] args) throws
           InterruptedException {
16
17         Counter counter = new Counter();
18         Thread a = new Thread(() -> {
19             for (int i=0; i < 10; i++) {
20                 counter.incrementCounter();
21                 System.out.println("Thread a " + counter.getCounter()
                       );
22             }});
23
24         Thread b = new Thread(() -> {
25             for (int i=0; i < 10; i++) {
26                 counter.incrementCounter();
27                 System.out.println("Thread b " + counter.getCounter()
                       );
28             }});
29
30         a.start();
31         b.start();
32         a.join();
33         b.join();
34
35         System.out.println(counter.getCounter());
36     }
37 }
```

1. May the two threads “a” and “b” execute the “incrementCounter” method simultaneously?
2. May the two threads “a” and “b” execute the “getCounter” method simultaneously?
3. May one threads execute the “incrementCounter” method while the other is executing the “getCounter” method?
4. Will line 35 of the above code snippet **always** print “20” as output?
5. On which object will the threads acquire the lock in this snippet?

Exercise C: Concurrency (2 Points)

Please answer the question below and provide a short and precise explanation.

Consider the following code snippet:

```
1  public class ExerciseC {
2
3      public static void main(String[] args) {
4          final List<Integer> integerList = Collections.
               synchronizedList(
5                  IntStream.rangeClosed(1, 1000)
6                           .boxed()
7                           .toList());
8          Runnable printIntegerList  = () -> {
9            for (Integer i : integerList) {
10               System.out.println(i);
11           }
12         };
13
14         Thread a = new Thread(printIntegerList);
15         Thread b = new Thread(printIntegerList);
16         Thread c = new Thread(printIntegerList);
17
18         a.start();
19         b.start();
20         c.start();
21     }
22 }
```

Since the "integerList" variable is a synchronized collection, can the foreach-loop at lines 9–11 (executed by three different threads) run simultaneously in all threads?

Exercise D: JVM Architecture (26 Points)

Please consider the class file distributed with this assignment ("ExerciseD/Secret.class"). You are expected to answer the following questions.

1. What's the name of the original Java file?
2. In the constant pool there are many sentences, spot the following two. The first one starts with the word "Hello". The second one starts with the word "Congratulations". Find these entries in the constant pool and report the line number *of the associated Utf8 tag.*
3. How many **public static** methods are defined in the provided class file?
4. This class file is used as a login command line tool. The goal of this exercise is to find the username and the password required to login.

   Your tasks:

   (a) Find the username, which is hard-coded in the classfile.

   (b) Find the "transform" method and provide, for every bytecode instruction in "transform", an explanation of their semantics.

   (c) According to the above explanation, provide an equivalent Java code for method "transform".

   (d) There is a string whose characters are (one-to-one) modified by the method "transform". Write the name of that string.

   (e) Write the password required to login.

Exercise E: Monitoring JIT (4 Points)

Consider the Java file distributed with this assignment ("ExerciseE/Inlining.java"). You are expected to answer the following questions. For this exercise you need to use some HotSpot JVM flags.

1. Which flags should be enabled for (if one flag requires other flags, specify all the flags in the order expected by the JVM.):
   (a) Enabling verbose diagnostic output with a message printed to the console every time a method is compiled.
   (b) Enabling printing of inlining decisions.
   (c) Enabling printing of collected native memory tracking data at JVM exit when native memory tracking is enabled in summary mode.
2. Which of the following methods is inlined during a standard execution of class Inlining? Motivate your answer providing the output of the proper java JIT flags. If some of the methods below are not inlined, provide an explanation looking at the source code.
   (a) Point::getX
   (b) Point::getY
   (c) Point::setX
   (d) Point::setY

Exercise F: Java Agent (15 Points)

For this exercise you need to install **maven** on your system.
Consider the Java files distributed with this assignment ("directory ExerciseF"). The directory contains the following files:

- compiled/ch/usi/inf/ajp22/Point.class: a class file that contains the specification of a class that represents a point;
- compiled/ch/usi/inf/ajp22/Main.class: the main class file that uses an instance of the Point class;
- src/main/java/ch/usi/inf/ajp22/agent/Agent.java: an agent class, that you are supposed to complete;
- src/main/java/ch/usi/inf/ajp22/agent/Modifier.java: an empty implementation of the ClassFileTransformer interface, that you are supposed to complete;
- run.sh: use this script to run the Main class without instrumentation;
- run-instrumented.sh: use this script to run the Main class with instrumentation;
- compile.sh: this script compiles the agent and creates the jar archive.

1. The provided compiled/ch/usi/inf/ajp22/Point.class file is malformed and cannot be executed on Java 17. Execute the application with the script "run.sh" and state what the problem is.

2. Write a Java agent that fixes the aforementioned problem. This **MUST** be implemented completing the Modifier.java and the Agent.java files and **MUST** affect only the bytecode of the Point class loaded.

    - To compile the Java agent use the provided "compile.sh" script that compiles the agent and creates the jar archive with a valid manifest.
    - You can verify your work, after compiling the Java agent, using the "run-instrumented.sh" script. This script loads the compiled agent and execute the Main class file.
    - **N.B.:** Use the provided script to compile and execute the project, any other compilation or execution command can modify the Point.class file and this can invalidate the exercise.