Object-Oriented Programming
August 2018

Notes or documents of any kind forbidden. Duration: 3 1/2h. Please answer the questions on separate sheets labeled with your name, section, and student ID.

1. The problem consists in programming in Java a class `Interval` suited for representing an interval over real numbers. An instance of this class is characterized by two *boundaries* a and b, with $a \in \mathbb{Z} \cup \{-\infty\}$ and $b \in \mathbb{Z} \cup \{+\infty\}$ such that $a \leq b$, and represents the set $[a, b]$ of all numbers $x \in \mathbb{R}$ such that $a \leq x \leq b$.

   The class `Interval` should satisfy the following requirements:

   - It must be possible to instantiate arbitrary intervals (including infinite ones), by specifying their boundaries.

     The boundaries of an interval cannot change after its instantiation.
   - It must be possible to check whether a given number $x \in \mathbb{R}$ belongs to a given interval.
   - It must be possible to check whether a given interval includes another one. An interval $[a, b]$ includes an interval $[c, d]$ if and only if $a \leq c$ and $b \geq d$.
   - It must be possible to check whether a given interval is finite or infinite.
   - It must be possible to compute (as a newly created interval) the smallest interval that contains two given intervals.
   - Instances of this class must be clonable, comparable to each other, and serializable. It must be possible to manipulate them simultaneously from separate threads.
   - In case of any error, a dedicated exception should be thrown.

   Note: You are free to choose the interface of constructors and methods, as well as to implement any additional class required by your solution.

2. (All answers should be thoroughly justified.)

   (a) In object-oriented programming, what is the purpose of defining abstract classes?

   (b) Explain the limited form of multiple inheritance allowed by the Java language, as well as how it can be used in programs.

   (c) If a Java instruction is able to raise a checked exception, what must be done by the programmer in order to avoid a compile-time error?

   (d) Why is it not allowed in Java to evaluate `new T()` if `T` is a type parameter?

   (e) Give a (small) example of a Java program that creates two concurrent threads, and then ends up in a deadlock.