**Introduction to Object-Oriented Programming**

**Java Exam (I0D41A & I0S75A)**

**Prof. Dr. Guy Baele**

**1pm, August 29th, 2022, 200C 01.01 / 01.03**

# Instructions

* This Java programming exam lasts for 1.5 hours at the most (for students of I0D41A: the R part starts after you hand in). There is only one question due to the time restriction of 1.5 hours (students with facilities are allowed to work for 2 hours on the Java exam).
* Assure yourself that there are 3 pages in this Word document (this instruction page + the exam question).
* To hand in, create a .zip file of all your Java files named “Surname\_Firstname.zip” (e.g., “Baele\_Guy.zip”). Do not submit .jar files! Only include .java files; there is no need to create separate packages in your code (i.e., keep all your .java files in the src folder). **There is no need to write a report.**
* You can use the provided file(s), which will be made available on Toledo during the length of the exam, for testing your code. There is no need to submit these files.
* The Toledo submission system will stay open for more than 2 hours, so be sure to remain calm and take your time to **ensure that you submit all required .java files**. Your submitted solution carries a time stamp, so try to submit on time. You can submit as many times as you want, but we will only correct your final submission. **Only submissions through Toledo are accepted! Late submissions will be penalized! Submissions via e-mail will not be graded!**
* Pay attention to encapsulation! As with all parts of this course, the key is to focus on principles of object-oriented programming, and hence not merely that “the program works”. Think about your code (structure) being reusable and easily extendible for future use; this is a key part of this exam! **Do not waste time on small details!**
* You are **not** provided with any .java files for this exam. There is **no example output** provided either. You need to write a small test program yourself to test the code you have written.
* You are allowed to use your own laptop as well as the computer in the computer room. Any form of communication is strictly forbidden. Please **turn off your smartphones**. Earphones / headphones are not allowed; soft earplugs to block out noise are allowed. **Put your student ID card on your desk for the duration of the exam.**
* **Don’t forget to sign your name on the sheet near the exit of the classroom before you leave! Good luck!**

# Question: Object-oriented programming

In this question / exam, the goal is to write your own Java classes to match the descriptions below. No .java code is provided as part of this question / exam. **Two example input files** are provided on Toledo.

Whenever someone becomes infected with a pathogen, such as the currently circulating coronavirus or monkeypox virus, bioinformatics tools can be used to study the genome sequences of that pathogen. To this end, the genome sequences (consisting of 4 possible nucleotides / characters: A, C, G and T) of a collection of samples are investigated by first creating an Alignment. The .fasta input file on Toledo contains such an alignment, in a commonly used file format known as FASTA. Basically, for each sequenced sample, the FASTA file contains 2 lines: first, the identifier (or name) of the sample (the lines that start with a ‘>’), followed by the corresponding genome. A small snippet of such an Alignment can be seen here, taken from the provided file on Toledo:

>2000.B.NL.00.671

TGTCCCGGGGACAGACCAACGCTAAGGCC…

>2001.B.UA.01.01UAKV252  
TGTCCCGGGGCCAGACCAACGCTAAGGCC…

>2005.B.JP.05.DR7060

TGTCCCGGGGACAGACCAACGCTAAGGCC…

You first need to implement an Alignment class that holds the identifiers and the corresponding genomes. Make sure this class has sufficient relevant features, e.g., adding and removing genomes according to an identifier, retrieving a genome according to an identifier, write its contents to screen, …

A bioinformatics team consists of a potentially large number of Users that can either simultaneously work on the same shared Alignment or work on their own individual copy (that is not shared with other users). A shared copy of the Alignment as well as all Alignments of the different Users are stored in a Repository. Make sure that you can perform the following operations on a Repository:

* adding and removing a specific User’s Alignment
* adding and removing the shared Alignment
* initialising the Repository for a collection of Users and their Alignments
* any other operations you deem necessary or useful

There are two types of team members:

* a Bioinformatician, who has access to a shared Alignment as well as her/his own Alignment (i.e., the content of these two Alignments can differ), but not to the Repository
* a TeamLead, who has direct access to the Repository to retrieve the shared Alignment but also the Alignment that each User is working on; the TeamLead does not have her/his own Alignment to work on and can select the Alignment of any User to be promoted as the shared Alignment

Each team member starts off with a copy of the same initial Alignment, which also serves as the shared Alignment. As different Users perform different operations on their Alignment, each User can hence have a different Alignment.

For your main program, you need to process the following information **through the command-line arguments** of the program:

* name of the file containing the bioinformatics team members; you can assume that each team member has a “simple” first name and last name without special characters
* name of the .fasta file

**Important, both input files can only be read once!** In your main program, show how to use of the functionality you have implemented in the different classes of your code and how the different classes are to be used together. You are free to implement any other classes you deem necessary to compose a fully working program.