CS106 – Introduction to Programming II

Spring 2025 – Software Development Project

1. Project Outline

Your final project for the module will be a software application in Java that **manages student information** in a higher education institution. It will have a Graphical User Interface (GUI) to allow the user to create and manage objects and it will store the information inside data files. You employ Object Oriented Development by designing and implementing classes of your own. You may find and use any library that serves you better, provided that it is freely downloadable or re-use classes that are handed out during the course. The final deliverable should be a Netbeans 23 project with Java JDK 23, must include any additional libraries, so that the source code could be able to be built/run by this specific IDE after an SVN checkout.

2. Project Requirements

Create the **Secretariat** application that keeps a list of student objects. Basic fields for the class `CStudent` are `ID`, `Name`, `Surname`, `Country`, `DateOfBirth`, `IsStudyAbroad` and you can add additional fields of your own design. The GUI of the application will have at least one main window with some command buttons (or a menu), text fields (or other UI elements) along with their corresponding labels, which allow the user to enter data for each object. Be creative and design a good user experience (UX) for your application, that will have the following command actions:

- The **"Browse"** action displays the list of students in a GUI form.
- The **"New"** action creates a new student, adds it to the list, and sets it as the current object.
- The **"Select"** action selects one student from the list of students as the current object.
- The **"Load"** action loads values stored in the current object to the GUI editors.
- The **"Save"** action saves the values from the GUI editors to the current object, in both cases of a new and an existing student.
- The **"Find"** action searches the list of students by their surname (or the starting part of a surname) given in a text field. You may extend this with other search capabilities (e.g. search by name) or add a "Filter" action that restricts the list (e.g. by specific country).
- The **"Export"** action will save the list of books to a data file with a format that you will decide (text file, CSV, XML, JSON, …)

You may create additional useful functionality on your own if you aim for a high mark.

3. Project Conditions and Rules

- This project counts **40%** towards your final module grade, please allocate the needed time, that is minimally estimated to 32 hours in total.
- The **first version** for the initial requirements should be submitted before the deadline. Afterwards deductions are applied according to the lesson syllabus. A second version that will be implemented inside the classroom is what determines your final project grade.
- This is an **AI orange** assessment. You may use AI to help you with small code snippets and record its use. Write down the AI tool used, time, the prompt, the response in a document. **Nevertheless, it is of utmost importance** that: a) you do **not rely on untrustworthy** code generation by LLM to build the application (you are going to lose time or get stuck) b) **build software development skills** without any AI / human assistance.

- The **second version** of the project will be done within a restricted amount of time and AI / human assistance would be restricted with the use of proctoring software. The **grade confirmation requirements** would need the skills that should have been acquired during the development of the first version.
    - If you have submitted an excellent first version, that means that you have the skills to develop all new requirements for version 2. If these are not confirmed a deduction as high as 50% could be made on your grade.
    - If you have reached a passing grade, you should be able to develop one simple requirement of version 2, to confirm your skills. If you cannot develop this bare minimum, the project will be marked as fail (F).

4. Criteria of Software Quality

During the development of your project, you should meet the following criteria. They are ordered from the highest importance to the lowest:

1. **Reliability** = The application must not have any critical flaws (e.g. hogging system resources) or wrong programming practices (e.g. global variables).
2. **Comprehensibility** = The code should be easily understood by others.
3. **Cleanliness** = Parts of code that serve no purpose (garbage code) should be deleted.
4. **Size of methods** = Unless it implements a complex algorithm, the size of each method should be the minimal required for increased cohesion.
5. **Standardization =** The code should follow some convention in its different parts, like the common coding and naming conventions handed out during the course.
6. **Error checking** = The application should prevent the user from entering erroneous input or performing action that could crash the application.
7. **High usability =** The application should provide a user-friendly experience, through a straightforward/clean/modern/attractive user interface, messages that help the user, etc.

Marking Scheme

This is a software design and development task which requires the best of your knowledge and use your coding skills. The following rubric explains how the deliverable software will be marked:

Range	Performance Description
100 - 80	A superb design and implementation of the application, possibly exhibiting qualities of a professional developer and deeper understanding of software engineering concept. Excellent chosen names for source files and code elements and strict use of code convention, allowing the solution to be quickly adopted and extended by another developer. Design leverages object-oriented concepts, methods, and best practices. The overall outcome provides clear indications of the participant's potential for a future career in the field of Software Engineering.
79 - 65	A sound and complete design and implementation of the application. Designed and implemented well, covers most of the issues in requirements correctly. Minor flaws or misunderstanding of concepts do not generate software defects, nor decrease the quality of the application. The overall outcome convinces that a very good level of understanding has been achieved on all subjects of the lesson.
64 - 50	A satisfactory solution for the design and implementation of the application, that has room for improvements. Omissions and errors may be present, but it provides convincing evidence of understanding how to develop an object-oriented solution. The overall outcome indicates a good level of understanding on the most important subjects of the lesson.

49 - 40	An accepted solution with an object-oriented design that runs correctly for the requested issues. Major design flaws and quality issues may be present, but the answer is in the right direction. The overall outcome shows that a baseline level of competence in the subjects of the lesson has been reached.
39 - 0	An application that crashes with an error or shows inadequate or no knowledge of OODD. Unable to provide a meaningful, related or even approach a correct solution for the required issues. There might be some objects present but they are merely replacing traditional procedural programming, not following a central design.