Advanced Programming 2025
Task Oriented Programming, Part 1/2
Assignment 11

May 22 2025

1 Goal

The goal of this exercise is to make you familiar with the basic concepts of the iTask system. After making this exercise you should know how to use basic interactive tasks such as `enterInformation`, `updateInformation`, and `viewInformation` for new data types, how to customize them, work with sequential and parallel task combinators, and create your own task functions.

On Brightspace you find a Clean module, `TOP1.icl`, that imports `StudentAdmin`. Module `StudentAdmin` has a sample population of students. Please note that the sample population has the deliberate mistake that there are multiple occurrences of the same student number. Furthermore, note that a student number is correctly formatted if it is a string that starts with s and is followed by exactly four digits. In `TOP1.icl`, a function called `myTask` can be started as a single iTask with:

```
Start world = doTasks myTask world
```

To obtain separation of concerns, define all pure functions in `StudentAdmin` (don't forget to export their type signatures in the corresponding definition module `StudentAdmin.dcl`) and all task functions in `TOP1`.

2 Assignment

2.1 UoD

We advice to start with the Universe of Discourse first, i.e. implement module `StudentAdmin`. The intended functionality is documented in `StudentAdmin.dcl`.

2.2 Tasks

You are required to make the following tasks. Please use the suggested names for the task functions (see `TOP1.icl`).

1. Make a task `enter_student` to enter a new student.
2. Make a task `enter_students` to enter a list of students.
3. Make a task `update_student` to update the given student.
4. Make a task `select_student` to select one student from the sample population.
5. Make a task `select_student_by_name` to select one student from the sample population where you only display the name of the students.
6. Make a task `select_year_sorted` to select one of the registered years of the sample population, where only the available years are displayed without duplicates, and sorted in increasing order.

7. Make a task `select_same_study` that, when given a year y and a student s, allows you to select from the sample population arbitrarily many students from the same year y and programme as the given student, except the student s herself. Of the selected students, only their names should be displayed.

   If the student s did not study in year y, then the task should display a message to the user and return the empty list of students.

   If no other students study in year y in the programme of student s, then the task should display a message to the user and return the empy list of students.

8. Combine the previous three tasks into a task `select_same_study_from_population` that first lets you select a student (`select_student_by_name`) and a study year (`select_year_sorted`) in parallel. This must be followed by the previous task `select_same_study` that lets you select students from the same study year and programme.

9. Make a task `update_studentnumber` to change only the student number of a student. Only valid student numbers should be accepted. The other fields must be displayed in one way or another, but should not be changeable. Display the resulting student.

10. Make a task `fix` that, given an initial sample population, repeatedly finds students with duplicate student numbers and asks the user to update them correctly, using task `update_studentnumber`. As soon as no issues exist in the population, the user can let the task return the updated population.

Deadline

To receive feedback, hand in your solution before June 1 23:59h.