Advanced Programming (I00032) 2018
Basic iTasks

Assignment 4

Goal

The goal of this exercise is to get you started with the iTask system. After making this exercise you should know how to use basic editors like `enterInformation`, `updateInformation`, and `viewInformation` for new datatypes.

Version

The iTask system is under development. To prevent version problems we will use a single system throughout the course (unless big problems pop up). This version is available at `http://www.cs.ru.nl/~baslijns/clean-itasks-afp-2018/`. Do not use the daily builds for the work you hand in. They may contain experimental features as well as bugs.

In order to check whether this iTask component works correctly you can open the project `Examples/iTasks/BasicAPIExamples.prj` in the distribution and run it.

Compiling an running is a bit different than with normal Clean projects. The easiest way of getting things up and running is to use the provided project file from brightspace[1].

Running the basic example project should create a console window. Start your favorite web-browser (there might be issues with MS Edge, Chrome and Firefox are better choices), and direct it to `http://localhost/` on Windows or `http://localhost:8080` on Linux or MacOSX. This should open the login page. Username `root` with password `root` as a valid login.

After pressing the `New` button you will see a list of tasks. Select one task and start it with the button `Start task` on the bottom of the list. Under admin a task is available to edit new user accounts. You can browse this application to get used to the iTask look and feel. Start as many examples from the list as you want.

1 Basic Edit Tasks

On Blackboard you will find a skeleton that defines a record type `Student` and a list of students. A function called `myTask` can be started as a single iTask with.

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

You are required to make some task using the given type `Student`. Name these functions `task`n where n corresponds to the label of the enumeration.

1. Make a task to enter a new student.

[1] Linux and MacOSX users should use `cpm` and not `clm`. Compile the project by running `cpm skeleton4.prj`

2. Make a task to enter a list of students.
3. Make a task to update the given `student`.
4. Make a task to select your favorite student form the given list `students`.
5. Make a task to select your favorite student form `students` where you only display the name of the students.
6. Complete the given generic to string function `gToString`. Use it to construct a task to select your favorite student from `students` where you display each student with `gToString`.
7. Construct a task to select potential partners (as many as you like) from `students`. This task should only display the name of the students and whether they are a Bachelor or Master student.
8. Define a task to change only the name of `student`. The other fields must be displayed in one way or another, but should not be changeable.

Each of these tasks can be written as a oneliner. The `gToString` takes about 15 lines.

Deadline

The deadline for this exercise is October 8, 2018, 10:30h (just before the next lecture).

Post Scriptum

If you want to create a project file yourself you can do the following:

Windows

- Use the environment `iTasks`
- Increase the heapsize and stack size
- Make sure `Dynamics` is enabled under the Profiling tab

Linux or MacOSX

- Make sure that you have `cpm` available.
- Create a new project by running `cpm project skeleton4 create`. This will create a `skeleton4.prj` project file.
- Set the correct environment by running `cpm project skeleton4.prj target iTasks`.
- Set a bigger heap and stack size and enable `Dynamics` by runnig `cpm project skeleton4.prj set -h 200m -s 20m -dynamics`