SYSC2004 Lab 1

The objective of this lab is to start learning the BlueJ environment, and to review what is meant when we say that objects have identity, state, and behaviour.

In this lab, you will learn:

- how to open a project containing several classes,
- compile the classes,
- interactively create objects,
- invoke methods on those objects, and
- use an inspector to observe the state of those objects.

Some of the steps in this lab have questions. You do not need to submit your answers for grading; however, you should answer each question (e.g., write the answers on a piece of paper) before moving to the next step.

To receive credit for this lab, you must demonstrate your work. When you have finished the exercises, call a TA, who will ask you to demonstrate BlueJ and review the code you wrote for Question 7. For those who don't finish early, the TAs will ask you to demonstrate whatever code you've completed, starting at about 30 minutes before the end of the lab period.

The classes of interest in this lab will be "LabClass" and "Student", as provided in the "lab-classes" project available on the CD that comes with the book, or on the BlueJ web site:
http://www.bluej.org/objects-first/resources/projects.zip
(Download and extract the projects to your working directory. Most of the labs will be based on these projects.)

"Student" objects will represent specific people with their names and student numbers (think of hockey players and their names / jersey numbers), and "LabClass" objects will represent labs for specific courses, with a time, an instructor name, and a group of students. The interesting point in this lab will be to show how objects can reference one another, i.e. for example how a specific "Student" object can be related to one or several "LabClass" objects, representing the real-life fact that students can be enrolled in one or several labs.

1. Open the "lab-classes" project, and compile the classes LabClass and Student.

2. Following the indications of exercise 1.21 of the textbook, create three Student instances: one with your own full name and student number, then two others with other names that you can invent (or take from friends, etc.).

   Make sure the name consist of at least 4 letters (ex. acceptable names: “Mary Kate”, ”Dick Howard”, ”John Doe” etc).

   Make up some student numbers. The student numbers must contain at least 3 digits (ex. “100”, “100 000 000”, etc).

As these are entered as characters, do not forget to put quotes “ around your names and the student numbers

3. Use the methods offered by the student class to:
   - find the login associated with each student
   - change the name of one of the Students you created to "Donald Duck"
   - add a credit of 20 to yourself for completing this lab

4. Create a new LabClass instance, with a capacity of 30. This means that a maximum of 30 students can enroll into this lab. Following the indications of exercises 1.23 - 1.26, enroll your three students into the lab.

   What happened here is that, in the "enrollStudent" method, the argument passed was not a simple type such as a string or an integer, but an object of another class, in this case of the class Student.

   Use the inspector to look at the fields in the LabClass object.

5. Use the methods offered by the LabClass class to set the instructor name to "Albert Einstein", the room number to AA507, and the time to "Mondays 11:30-1:30".
   Notice that here, the time is a String.

   Question: Can you think of other (perhaps better) ways of associating this information (the time or timeslot for a class) to LabClass objects? Discuss briefly (why this solution might be better).

6. Create another LabClass with a capacity of 20, with instructor "Berthold Brecht", on Fridays from 3 pm until 5 pm, in room "Library 001". Enroll yourself and Donald Duck into this lab. Use the "printList" method of both your labClass objects to printout both lab lists.

7. We will now take a look at the java code making this all work.

   To view the code implementing a class, right-click on the class in the BlueJ window and select "open editor". Read the code, notice the code structure, with the class attributes, and the class methods.

   We will make a small change to the code. Let's suppose this is a very generous university system and every student who enrolls in a lab class automatically gets two credits, and even those who try to enroll but cannot because the lab is full still get one credit for trying.

   Where is the method that enrolls a student to a class? Is it in LabClass or in Student? Where is the method that adds credit to a student? Which student is being enrolled?

   Modify this method so that when a student is enrolled, he or she receives two credits, and in the case where the student cannot enrol because the lab is full, he or she gets one credit.