Java Programming
Course Description

Onur Derin
Advanced Learning and Research Institute
Faculty of Informatics
Università della Svizzera Italiana
derino@alari.ch

1 Attendance and Grading Policy

Due to the variety of student backgrounds at ALaRI, the first month of the academic year is composed of pre-courses that aim at creating a more uniform class. Java programming course is one of these short courses and spans a two-week period, totalling twelve hours of class. It is a non-credit course and can be omitted by those who have had it in their previous studies. The course assumes some familarity with C/C++. Figure 1 summarizes who should attend this course and how the grades are assigned. In the case of a conflict with ALaRI grading policies, ALaRI policies take precedence.

2 Lab

This section describes how to work with the exercises that will be done during the class. In the course materials, you will find the *lab-skeletons.tar.gz* file that contains the code skeletons for the class exercises and assignments. The description of these exercises are given in the lecture notes.

- lab-skeletons.tar.gz and lecture notes are available online:
  http://www.alari.ch/people/derino/Teaching/Java/index.php

2.1 How to work with the exercises

It is assumed that you are able to work under Linux, use the shell and a text editor (vi, Emacs, gedit etc.). Extract the contents of lab skeletons file by issuing **tar xvf lab-skeletons.tar.gz**
For a given exercise

- **cd** into the exercise folder,
- create the required classes,
- edit Makefile if necessary,
- **make** to compile,
- **make run** to run,
- **make check** to see if your program is correct,
- **make clean** to remove *.class files.

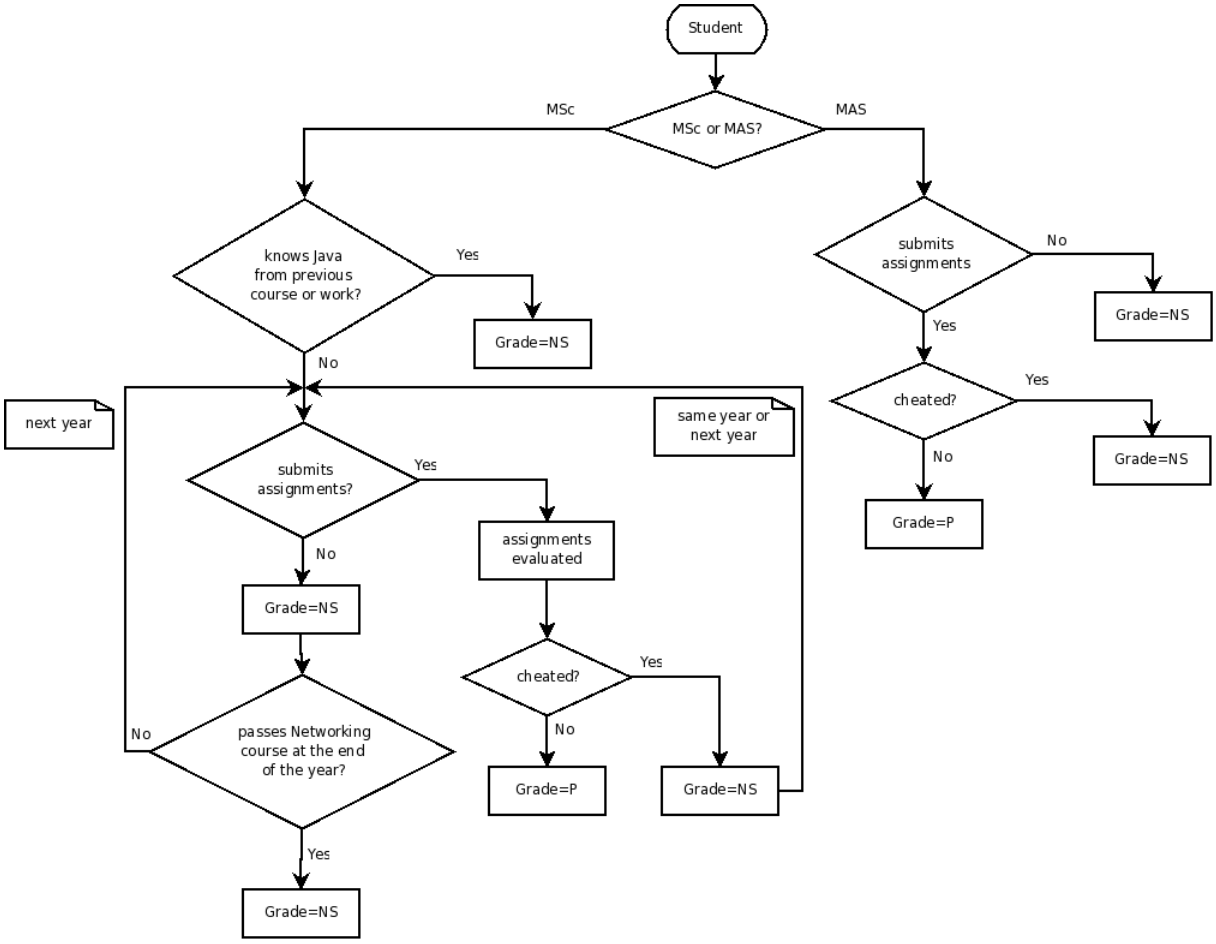

Figure 1: Attendance and Grading Policy

2.2 How to edit the Makefile

Each exercise folder contains a Makefile to compile the exercise. The Makefile of the exercise should be edited to include a list of the source files (*.java), and **PROGRAM** variable should be set to the class that contains the **main** method. For example, the Makefile for *10-vector-assignment* looks as follows:

```
JAVA_SRC = SortedVectorTest.java SortedVector.java SortedCollection.java
PROGRAM = SortedVectorTest
include ../java.Makefile.std
```

The java.Makefile.std file

This file is included by the Makefiles of all the exercises and provides the capability to work in a uniform way with the exercises.

2.3 How to submit the assignments

- The exercise folders whose name end with *-assignment* are the ones that you are supposed to submit.

- In most of the exercise folders, there is a test file (e.g. *SortedVectorTest.java*) that shows an example usage of the classes you are expected to write. Don't modify the test file!
- In most of the exercise folders, there is a *correct-output.txt* file that shows what your code should produce when the test file is run by **make run**. Don't modify the correct output file!
- Before submitting your code, check that it produces the same output as *correct-output.txt* by **make check**.
- Clean your *.class files before submitting your code by **make clean**.
- In order to submit your code, you should compress each assignment folder and send the compressed files to *derino@alari.ch*.

  Here is an example:

  For the assignment *10-vector-assignment*, the files you will have in the exercise folder after solving the assignment are:

  - Makefile
  - SortedCollection.java
  - SortedVector.java
  - SortedVectorTest.java
  - correct-output.txt

  Within the directory that contains the assignment folder, you can issue the following command to create a compressed file for this assignment:

  # tar czf 10-vector-assignment.tar.gz 10-vector-assignment/

  Do this for all the assignments and send me each of your compressed (*.tar.gz or *.zip) files. Don't send me Eclipse or NetBeans project folders!

3 Cheating Policy

By submitting your assignments, you confirm that you agree with the below cheating policy. Note that the assignments are to be done individually.

(Original text: http://www.andrew.cmu.edu/course/15-CSAP/cheatingpolicy.html)

Program plagiarism will be suspected if an assignment that calls for independent development and implementation of a program results in two or more solutions so similar that one solution can be converted to the other(s) by a series of simple commands;

Cheating will be suspected if a student who completed an assignment independently cannot explain both the intricacies of the solution and the techniques used to generate that solution.

If cheating has occurred, the student will fail the course receiving the grade NP.

Examples of Cheating:

- Turning in someone else's work, in whole or in part, as your own (with or without his/her knowledge).
- Turning in a completely duplicated assignment is a flagrant offense.
- Allowing another student to turn in your work as his/her own.

- Several people writing one assignment and turning in multiple copies, all represented (implicitly or explicitly) as individual work.
- Stealing an examination or solution from the instructor.

Examples of Not Cheating:

- Turning in work done alone or with the help of the course's staff.
- Submitting one assignment for a group of students if group work is explicitly permitted (or required).
- Getting or giving help about using the computers.
- Getting or giving help about solving minor syntax errors.
- High level discussions of course material for better understanding.
- Discussing assignments to better understand them.

4 Getting help

If you have any questions, you can send me an e-mail (derino@alari.ch) or come to my office (Room 162).