Software 1 - Submission Guidelines

General

The following page describes the submission guidelines for the assignments in the course 'Software 1'. The content of the various assignments will be separately detailed. The assignments will be checked both by a person and by an automatic program. Therefore, it is very important to follow carefully the guidelines below. Failing to follow the exact names, directories, files, times, etc. might lead to the failure of the automatic checking mechanism, which in turn, will cause a significant reduction in the exercise grade.

Procedural Issues

- A student must submit about 80% of the assignments. Specifically, a student is allowed to skip at most 2 non-mandatory assignments.
- A student should submit the assignments by himself/herself. You may share ideas, but you may not share code lines! Severe actions will be taken against students whose work will be suspected as copied!
- Homework should be submitted on time. However, each student has a total grace period of five days for all assignments. Students with a justified reason for late submission of an assignment (e.g. sickness, military service) are asked to submit the following form and present an official confirmation to the exercise checker in order to receive additional time for submission.
- The assignments will be programs in Java 1.5 and should generate no warnings whatsoever.
- Deliverables – Unless specified differently, each assignment should be submitted as a .zip file via the VirtualTAU system (http://virtual.tau.ac.il). Besides source code files (.java files), the zip file should contain a WORD file with a copy of each .java file. Instructions for using the VirtualTAU system are available at at http://virtual2002.tau.ac.il/upload/misc/main1.html.

Questions and Support

Unclear issues or problems regarding the UNIX environment, the nova machine, or any other system relating issues should be directed to the system consultants. They can be accessed via email (~system) or in person in the advisors room on the first floor in Schreiber building, room 019.

Other questions, regarding procedural or programming issues should be coped with in the following order:

1. Read the relevant exercise directions
2. Take a moment and think again whether the question makes sense and cannot be solved alone
3. Questions about the checking procedures, grades, directories and files that were misplaced, etc. please send to the exercise checker. If all else fails, send an email to the course mailing list stating the question shortly and clearly and one of the course's staff will answer it. Such an email should be in plain English only and its subject field should be Software 1. Students are also welcomed to answer questions sent to the mailing list.

4. Exercises should be read upon receiving them. Questions regarding the exercise should be submitted up to 5 days before the submission deadline, to allow sufficient time to answer.

Grading Criteria

The grades will be composed of the following parts:

1. **Submission on time**. Students should submit the exercise on time, except for justified reasons and a total grace period of 5 days. As a rule of thumb, there will always be an 'objective' problem in the day preceding the submission; a power failure, a printer malfunction, printer running out of paper, etc. Therefore, prepare your homework way in advance. Leave plenty of time for debugging, testing, and dealing with unexpected problems. Unexpected problems are the most expected things in programming projects!
2. **Correctness of the program**. The correctness will be checked by an automatic mechanism, and therefore a special care should be taken meeting the exact syntactic requirements.
3. **Design**. The programs should be well designed using the concepts studied in class (e.g., object-oriented, design by contract)
4. **Implementation efficiency**. Your programs are expected to be reasonably efficient.
5. **Documentation**. The inline documentation is composed mainly of a reasonable amount of comments (when needed). Make sure that each file and function includes a short paragraph explaining its purpose, IO and method of work.
6. **Readability and Clarity**. Naming - Use intelligent naming for variables, methods and classes. Modularity - divide code into methods and classes. Comments.
7. **Indentation**. Sometimes a Tab is better than thousands words.
8. **Honesty**. No extra points will be given for honesty. Any suspicion of deception, however, will cause immediate and severe steps.