

Take-home exam for IOOR

Autumn term 2007

General information

- This exam gives one Swedish credit
- Possible grades are **U,G,VG for SU, and U,3-5 for KTH etc. etc.**
- Handed out at 2007-01-23
- To be handed in by email before 2007-11-29, 23:59:59 (don't be late).
- You should use the literature, but you *may not copy* any answer, not discuss these questions with anyone else, or allow anyone to aid you in the answering of these questions. *Your answers must be products of your own mind. Omitting references to direct quotations is plagiarism.*
- **NOTE:** The sum of your answers must not exceed 10 pages (L^AT_EX, article, 10pt) or 3500 words (whichever you prefer), *excluding* title page and bibliography. Divide this between your answers as you please. You may not exceed this limit!

There are no deliberate trick questions in this exam. However:

1. Read the questions with a critical mind as there might be more to the question than what is explicitly stated. For example, for the question “argue why X is better than Y ”, it might be the case that X is sometimes better than Y and sometimes worse, depending on Z , or something else.
2. Questions need not have a single correct answer and can seldom be answered by a single statement. The questions are meant to be “thought provoking”—we hope that your knowledge and OO-skills will be made visible in the *reasoning* in your answers.
3. Sometimes it is absolutely necessary to make some (or even huge) delimitations (avgränsningar) to be able to produce a sensible answer.

Your ability to act in accordance with the above is a good sign of your understanding of the course's contents.

How to pass this exam

In this exam, *you are to show us* that you fulfil the criteria listed in the goals section under “Course Goals” on the web, modulo, perhaps, the fifth goal, about writing actual programs.

If your answers look like the ones you give at closed exams (salstenta), they are most likely not good enough. Argue, reason, convince, and elaborate! Simple yes/no-answers most likely won't fly. At the same time, core-dumping is not

acceptable. A long answer with a lot of irrelevant remarks is worse than one that is short and to-the-point, but misses something. Your ability to keep irrelevant information out of the answers will affect your grade.

In order to pass the exam, you must pass all questions. This is standard for take-home exams as you have several days to complete the exam, and are allowed to use any books, web pages etc., *as long as you quote your sources*. **Stuff about grades.**

Even though your answers are not research papers, avoid overly use of “weasel words”¹.

Handing in the exam

You should hand in the exam *by email* (see previous page for time and date) to `ioor@dsv.su.se`. We want a nicely formatted *PDF*-file containing your answers with your name, email and social security number clearly printed on the (otherwise empty) title page. Good language, good presentation and typesetting will of course improve your marks. You should receive a short message within a few hours after submission to ensure you that it was received correctly. (If you write in L^AT_EX, please also submit your source files.)

Words of advice

Start by disassembling the exam questions into a form that fits the workings of your brain. Generally, each exam question is a collection of sub-questions about a specific topic. One reason for using this form is that we like the answer to be in the shape of a short essay where many issues are *nicely tied together* (and related), as opposed to 10-15 short unrelated paragraphs with blurted out answers to smaller questions.

It is generally not possible to score VG or 5 without addressing all sub-questions. One possible strategy for answering is to list all sub-questions and tick off the ones you’ve answered to keep track of your progress. Unless you aim for the highest marks, don’t be too afraid if you don’t tick them all off.

A good way of doing a take-home exam is to complete the exam the first or second day of the exam period, let it rest for an entire day, and then go back to polish your arguments. As you are most likely aware, this is a good strategy for any text you produce.

And, importantly: a longer question does not imply longer answer.

And, finally, the questions

1. *Different Types of OO*

Given a *fairly large* implementation task, reason about the effects (on design, productivity, performance, maintenance, etc.—both in the small and in the large) of choosing:

- (a) A pure vs. a non-pure OOPL
- (b) A prototype-based vs. a class-based OOPL

¹http://en.wikipedia.org/wiki/Wikipedia:Avoid_weasel_words

- (c) A language with multiple or mixin inheritance vs. a language with single inheritance²

What are the possible effects? When, why, under what assumptions, etc.? Don't comment on the APIs available etc., but concentrate on language issues. Also, make a short note about whether your reasoning would be affected if it was a smaller implementation task.

In addition to this, state your personal preferences for each choice above (what you think would be best, not what you are most comfortable with), motivate them briefly and comment on how this may have influenced your answer to this questions.

2. *Different Constructs*

In Objective-C, there is a concept called categories. Compare categories, aspect-oriented programming, mixins/traits and metaclasses. Are there any "deeper similarities" (e.g., not syntactic) between some or all of them? Compare their respective expressive power and where they can be useful. Does one void the need of another? Why?

3. *Different Types of Typing*

Discuss structural and nominal typing. What are the benefits of each? Is one more ad hoc than another? Explain if (and why/how) they can be used side-by-side in a single programming language, or are they mutually exclusive? Are there any situations where structural typing fits a problem better than nominal typing and vice versa? Is there a difference between structural typing and dynamic typing?

Good luck!

Beatrice and Tobias

²Single inheritance does not allow multiple interface inheritance in this question.