FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Review report of a final thesis

Reviewer:	Ing. Konrad Siek, Ph.D.
Student:	Bc. Rasul Seidagul
Thesis title:	Implementation of Object Oriented Languages
Branch / specialization:	Web and Software Engineering
Created on:	25 August 2022

Evaluation criteria

1. Fulfillment of the assignment

▸ [1] **assignment fulfilled**
[2] assignment fulfilled with minor objections
[3] assignment fulfilled with major objections
[4] assignment not fulfilled

The thesis sets out to fulfill three tasks: analyze object oriented (OO) features of programming languages, design an educational OO language based on the analysis (TinyC+), and implement a transpiler from this language into a simplified C-like language (TinyC). I am convinced it succeeds in all three.

2. Main written part

75 / 100 (C)

The structure is logical and follows the tasks of the assignment, with consecutive chapters presenting an overview of OOP languages, intended features of the transpiler, and the implementation. The only thing missing was a chapter relating the implementation to its goals somehow. As such, figuring out how the work fulfills its educational aims was left as an exercise for the reader.

The student does a good job introducing concepts and implementation details in manageable chunks. The thesis uses examples well, both code and graphics, in support of its points. It also made good use of notes to point out interesting issue.

Nevertheless, the text is marred with linguistic problems. While I did not have trouble understanding the prose, the document is peppered with niggling grammatical errors and misused words. It is also missing quote a few obvious citations where the author explicitly refers to some fact (eg. "Such ambiguity is a well-known issue referred to as the diamond problem").

As a typesetting note, inline code was presented in literary quotes, where it should have been given in distinctive code blocks.

3. Non-written part, attachments 90/100 (A)

TinyC+ works in that it was capable of transpiling tinyC++ examples I made up into sensible-looking TinyC code.

The implementation largely uses conventional solutions to OOP problems. This is welcome, since the transpiler is meant to be eductional and this produces no surprises. Implementing virtual tables as structs rather than arrays is a nice touch.

The internals of the implementation are straightforward: the parser produces an AST and the transpiler does two passes over it: a type/sanity check and code generation. Each pass is implemented with a visitor as is standard practice.

The code itself is tidy and relatively easy to follow.
There is also some effort at providing in-code documentation, although in certain parts of the code it remains sparingly restricted to section headers in larger functions.

My major concerns for the previous version of the implementation were addressed: virtual method invocations are more conventional, error messages are more user-friendly.

Minor annoyances: when trying out the code form the TinyC+ repository, I had to do a few things for the code to compile. The README file should have been more helpful in this.

4. Evaluation of results, publication outputs and awards 90/100 (A)

The transpiler is meant to be used as part of a course. It looks reasonably fit for purpose, although the TinyC+ language may need to be extended depending on the exact scope of the course.

The overall evaluation 80/100 (B)

This second iteration of the students work shows a marked improvement from the previous and most of the significant point I had raised were resolved.

The student's work describes the landscape of OOP features in four mainstay OO languages and presents a case for the features that are selected for TinyC+ reasonably well. The text does a good job of explaining how these features were implemented in the form a transpiler from a toy C++-like language to a toy C-like language. The implementation is competent and easy to follow. The implemented solutions do not surprise, which is good the intended educational application.

The written portion of the work leaves something to be desired, but is nevertheless legible and much improved from the previous iteration.

Questions for the defense

No questions.

Instructions

Fulfillment of the assignment

Assess whether the submitted FT defines the objectives sufficiently and in line with the assignment; whether the objectives are formulated correctly and fulfilled sufficiently. In the comment, specify the points of the assignment that have not been met, assess the severity, impact, and, if appropriate, also the cause of the deficiencies. If the assignment differs substantially from the standards for the FT or if the student has developed the FT beyond the assignment, describe the way it got reflected on the quality of the assignment's fulfilment and the way it affected your final evaluation.

Main written part

Evaluate whether the extent of the FT is adequate to its content and scope: are all the parts of the FT contentful and necessary? Next, consider whether the submitted FT is actually correct – are there factual errors or inaccuracies?

Evaluate the logical structure of the FT, the thematic flow between chapters and whether the text is comprehensible to the reader. Assess whether the formal notations in the FT are used correctly. Assess the typographic and language aspects of the FT, follow the Dean's Directive No. 52/2021, Art. 3.

Evaluate whether the relevant sources are properly used, quoted and cited. Verify that all quotes are properly distinguished from the results achieved in the FT, thus, that the citation ethics has not been violated and that the citations are complete and in accordance with citation practices and standards. Finally, evaluate whether the software and other copyrighted works have been used in accordance with their license terms.

Non-written part, attachments

Depending on the nature of the FT, comment on the non-written part of the thesis. For example: SW work – the overall quality of the program. Is the technology used (from the development to deployment) suitable and adequate? HW – functional sample. Evaluate the technology and tools used. Research and experimental work – repeatability of the experiment.

Evaluation of results, publication outputs and awards

Depending on the nature of the thesis, estimate whether the thesis results could be deployed in practice; alternatively, evaluate whether the results of the FT extend the already published/known results or whether they bring in completely new findings.

The overall evaluation

Summarize which of the aspects of the FT affected your grading process the most. The overall grade does not need to be an arithmetic mean (or other value) calculated from the evaluation in the previous criteria. Generally, a well-fulfilled assignment is assessed by grade A.