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: January 26, 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). It fails the first two tasks and succeeds in the third (with one issue).

Task 1: OO feature analysis

The selection of evaluated features is incomplete on a very basic level (e.g. no dynamic dispatch, no mixins, no prototypes). More importantly, the evaluation itself is superficial. Each feature is described by just a paragraph of text describing it in structural or semantic terms. There is no in-depth analysis of how the feature is implemented in existing languages, how the feature interacts with other mechanisms of the language, nor how it impacts the ergonomics or power of the language. Overall, the analysis does not bring any value apart from listing and explaining terminology.

Task 2: Language design

The language is meant to be used in education. The use in a specific course is a linchpin of all design decisions in the thesis, but the exact requirements of the course, or any specific design concerns are never defined. The thesis settles for proposing a language "consisting only of fundamental features" which I deduce to mean providing mechanisms for encapsulation, inheritance, and polymorphism, supposedly respectively fulfilled by adding classes, single-inheritance, and virtual methods to the language. Arguably these are not fundamental (e.g. prototype-based OO systems do not need classes). On the other hand, these are also insufficient, since there is no mechanism for ensuring
encapsulation---"classes alone," as the author states, "do not bring much value," and they do not provide mechanisms for controlling access to internal members of objects necessary for encapsulation. In addition, the implementation also provides the ability to define abstract methods (methods with no bodies), which are not discussed in the design section of the thesis.

The remaining features introduced in the analysis are discarded as too complex or unnecessary. Necessity is couched in terms of whether a feature leads to changes in the output, which is both naive and incorrect. The aforementioned access modifiers do not produce additional output if encapsulation is maintained, but generate compilation errors, if encapsulation is pierced. Complexity in itself is not intrinsically a good reason to omit a feature.

The design of the syntax of the introduced extension is also described only superficially. It gives the impression of incongruity with TinyC and is apparently lifted from C++ without further consideration. This might be a good choice, but it is insufficiently explored to make this case.

All in all, there is a general lack of sufficient exploration of the author's design choices.

Task 3: transpiler implementation

The implementation of the transpiler is fine within the design framework laid out in the thesis and the loose specification laid out in the task description. The implementation is clean and the transpiler works. My objection is that the designed language is relatively feature poor, so the resulting transpiler is small.

2. Main written part

The work reads like a first draft. As a completed thesis, the work is superficial, incomplete, and riddled with errors. It needs more work on a very basic level.

Development

By far the biggest problem of the written part of the thesis is its general underdevelopment, causing the work to fall short when conveying ideas. Many sections are single paragraphs that introduce various concepts, but do not go further into analysis or more detailed description, often omitting key points. As a result the introduction fails to convey the goals, scope, and parameters for success (i.e. a thesis statement). The design and state of the art portions are brief and superficial. The implementation section does not sufficiently explain how the proposed transpiler works, leaving many obvious questions unanswered. Conclusions seem only pro forma. There are elements that are outright missing, for example a description of TinyC, the output language of the transpiler.

Length

While the length of a thesis does not necessarily signify quality in general, the lack of depth of this particular work is underscored by its extreme brevity.

Structure

The structure of chapters is very confusing. While there is an overall outline that the thesis is following, the internal structure of each chapter is chaotic, making them read like collections of loose thoughts without conveying a point.
The structure is also incoherent, with terms of concepts being used before they are introduced in separate sections much later on.

Others

The text is peppered with numerous grammatical, typographical, typesetting, stylistic, and punctuation errors. I will pass on detailed notes to the student.

3. Non-written part, attachments 75/100 (C)

Standalone, the implementation is relatively bare-bones, but it is within the specification laid out by the design and task of the thesis. The code is well written and the transpiler works as expected.

Implementation notes

The one specific problem with the implementation is the behavior of virtual methods called from within parent methods called from within a child object via the base keyword. These should execute the implementations defined by the child, but due to the association of virtual tables with classes rather than objects, they call implementations defined by the parent.

Implementing virtual tables as structs rather than arrays is a nice touch.

Ergonomics

If the implementation is to be used in a classroom setting, it should produce more beginner friendly error messages. The transpiler also does not preserve the original whitespace. On the other hand the author does take some steps towards making the transpiler user friendly, like the ability to print out code with syntax highlighting.

4. Evaluation of results, publication outputs and awards 85/100 (B)

The transpiler is meant to be used as part of a course. I do not know about the details of the course. In the abstract, the work is fit for purpose, although the TinyC+ language may need to be extended depending on the exact scope of the course.

The overall evaluation 40/100 (F)

While the implementation is sufficient, the written portion of the work is not.

The written portion of the thesis should convey clearly the motivation and goals of the work.

In a thesis centered around an implementation, the written portion should describe the design and the implementation, pointing to interesting features and solutions.

Nevertheless, the work is promising and I would encourage the student to spend more time on the thesis, especially the written portion of it, and to subsequently resubmit.
Questions for the defense

What are the pros and cons of associating virtual tables with classes, rather than objects?
How does the design used in the implementation impact other features?
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 fulfillment 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.