

Insert here your thesis' task.





**FACULTY  
OF INFORMATION  
TECHNOLOGY  
CTU IN PRAGUE**

Master's thesis

# **SimpleObjectMachine implementation**

*Bc. Rudolf Rovňák*

Department of Theoretical Computer Science

Supervisor: Ing. Petr Máj

November 4, 2020



---

# Acknowledgements

THANKS (remove entirely in case you do not wish to thank anyone)



---

## Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No.121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on November 4, 2020

.....

Czech Technical University in Prague  
Faculty of Information Technology  
© 2020 Rudolf Rovňák. All rights reserved.

*This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).*

### **Citation of this thesis**

Rovňák, Rudolf. *SimpleObjectMachine implementation*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.



---

## Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

**Klíčová slova** Replace with comma-separated list of keywords in Czech.

---

## Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

**Keywords** Replace with comma-separated list of keywords in English.



---

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 State-of-the-art</b>	<b>3</b>
<b>2 Analysis and design</b>	<b>5</b>
2.1 SOM design and features . . . . .	5
2.2 Parsing . . . . .	5
2.3 Interpretation . . . . .	5
2.3.1 AST interpretation . . . . .	6
2.3.2 Bytecode interpretation . . . . .	6
2.4 Optimization . . . . .	6
2.5 Virtual Machine . . . . .	6
2.5.1 Garbage collection . . . . .	6
<b>3 Realisation</b>	<b>7</b>
<b>Conclusion</b>	<b>9</b>
<b>Bibliography</b>	<b>11</b>
<b>A Acronyms</b>	<b>13</b>
<b>B Contents of enclosed CD</b>	<b>15</b>



---

## List of Figures



---

# Introduction





# State-of-the-art



---

# Analysis and design

## 2.1 SOM design and features

- Data types: Integer, Char/String, (Boolean, Float)
- Basic arithmethics (boolean arithmethics?)
- Bitwise operations
- Classes - fields, methods, single inheritance, dynamic dispatch (late binding)

## 2.2 Parsing

TBD, use parser generator (ANTLR)?

## 2.3 Interpretation

Once the source code is parsed, the next step is executing it – this step is called *interpretation*. Interpretation is As per [1], an interpreter for a language L can be defined as a mechanism for the direct execution of all programs from L. It executes each element of the program without reference to other elements.

It is however very rare that any language is interpreted directly. In most cases of non-trivial languages, the interpretation process is preceeded by parsing or compiling into some form of *intermediate representation*. According to [1], this process removes lexical noise (comments, formating), elements can be abstracted/combined (into keywords, operations etc.) and reordered into execution order (for example operators in an algebraic expression).

The choice of intermediate representation is therefore vital. It can determine a lot of aspects of interpretation - from the way of distributing the interpreted program to time and space complexity of the interpreter.

### 2.3.1 AST interpretation

*Abstract syntax tree (AST) is a tree representation of the source code of a computer program that conveys the structure of the source code. Each node in the tree represents a construct occurring in the source code [2].*

As the name suggests, AST represents the source code in the form of a tree. During the transformation from the source code to AST, some information is omitted. Information that is vital for AST's according to [2] is:

- variables – their types, location of their definition/declaration,
- order of commands/operations,
- components of operators and their position (for example left and right operands for a binary operator),
- identifiers and corresponding values.

### 2.3.2 Bytecode interpretation

Using a form of bytecode. Effective, requires:

- designing the bytecode (instructions, bytecode file formats),
- AST to bytecode translation (AST to bytecode instructions),
- actual bytecode interpretation.

Bytecode interpretation permits easier optimization.

## 2.4 Optimization

- dead code elimination,
- constant propagation,
- others...

## 2.5 Virtual Machine

Decide on memory hierarchy, garbage collection...

### 2.5.1 Garbage collection

# Realisation



---

## Conclusion





---

## Bibliography

- [1] Wolczko, M. Execution mechanisms Part I: Interpretation. [online], 2015, [cit. 2020-10-31]. Available from: <https://www.dropbox.com/s/1fav564dvx20qsw/2%20AST%20Interpretation.pdf>
- [2] DeepSource Corp. Abstract Syntax Tree. [cit. 2020-11-4]. Available from: <https://deepsourc.io/glossary/ast/>



## Acronyms

**AST** Abstract syntax tree



## Contents of enclosed CD

	readme.txt .....	the file with CD contents description
	exe .....	the directory with executables
	src .....	the directory of source codes
	wbdcm .....	implementation sources
	thesis .....	the directory of $\text{\LaTeX}$ source codes of the thesis
	text .....	the thesis text directory
	thesis.pdf .....	the thesis text in PDF format
	thesis.ps .....	the thesis text in PS format