

Eiffel library to generate Java bytecodes

DIPLOMA PROJECT PLAN

Project period: 2003-05-26 Monday – 2003-09-25 Thursday

Student name: Daniel Gisel

Email address: daniel@gisel.ch

Supervising Assistant: Karine Arnout

Supervising Professor: Bertrand Meyer

1. PROJECT DESCRIPTION

Overview

One goal of the Java language environment is making “write once, run anywhere” possible. To achieve this, the Java compiler translates Java programs into an intermediate language called Java bytecodes – the platform-independent codes executed by the Java Virtual Machine (JVM) [6]. This means that as long as a computer has a JVM, a program in Java bytecodes can run on it, independent of the operating system or the hardware of the computer.

Even if the JVM and the Java bytecodes are designed and optimized for the Java programming language, the JVM does not assume that the instructions it executes were generated from Java code. It is also possible to compile other programming languages to Java bytecodes.

The Java bytecodes of one class is always stored in class files (name.class). This file contains all information about the class, like the access rights, references to the parent class and all the interfaces the class implements, tables with all the fields and the methods (including their code) of the class and the constant pool. The constant pool is a table of structures representing various string constants, class and interface names, field names and other constants that are referred to within the class file structure and its substructures.

Scope of the work

The main work of my diploma project is to design and implement an Eiffel library which is able to generate Java class files that can be executed by a Java Virtual Machine. To achieve this goal, I will do the following steps.

First I have to create an object oriented data structure that maps the structure of a Java class file. This data structure will contain classes that represent a Java class, a method, a field, the constant pool, and so on. These classes will be fully contracted to catch as many errors as possible in the code generation to avoid invalid class files.

Based on this data structure I will build a layer that allows an easy Java class file creation for the client. In this layer there will be generators for classes, methods and fields.

To test this library and also to give examples how to use the framework, I will create a simple language with a corresponding compiler that uses the library as backend. The language will be similar to the Java language, but much simpler.

Intended results

Library:

The created library will be fully contracted and produce correct Java class files. It will be built up of two layers. The first layer is very close to the structure of the class files and allows direct access to the elements. The second layer implements generators for classes and methods and allows an easy creation of class files.

Compiler:

I will design a simple language similar to Java that is compiled to Java bytecodes. The backend of the compiler will be the library created in the first part of my project. The goal of this compiler is to test the library and to show how to use it.

Documentation:

The documentation will contain three documents. The first one is the report of the whole project. The other two documents are a user and a developer manual for the library.

2. BACKGROUND MATERIAL

Reading list

Eiffel:

- Object-Oriented Software Construction [7]
- Gobo Eiffel Project [2]

Java:

- Byte Code Engineering Library (BCEL) [3]
- The Java Language Specification, 2nd edition [5]
- The Java Virtual Machine Specification, 2nd edition [6]

Compiler:

- Gobo Eiffel Project [2]
- Compilers: Principles, Techniques and Tools, Addison-Wesley [1]
- Advanced Compiler Design and Implementation [8]

3. PROJECT MANAGEMENT

Objectives and priorities

<i>Objective</i>	<i>Priority</i>
Library (lower level)	1
Library (higher level)	2
Compiler	3
Report	1
User manual	3
Developer manual	2

Criteria for success

The main goal of this project is to create a well designed, correct and comprehensible library to generate Java bytecodes. So, the criteria for success are the quality of the software and the documentation. To achieve a high quality, the following points must be satisfied:

Quality of software:

- Design by Contract
- core principles of Object-Oriented Software Construction, 2nd edition [7]
- good design of the interfaces
- completeness:
 - o The library should cover all aspects of a Java class file. It should provide an interface that allows the user to directly work with the structures of a class file (lower level). On the other hand there should also be an interface that makes it possible to easily create a correct class file with a generator (higher level).
 - o The compiler and its corresponding language should be as simple as possible. Their only purpose is to be a tool for testing the library and to give examples how the library can be used.

Quality of documentation:

- good structure
- understandability
- completeness
- covers the final state of the work

Method of work

I will use the following technologies for my project:

- Eiffel programming language
- Gobo Eiffel Lex and Yacc [2] for implementing the scanner and parser of the compiler
- Gobo Eiffel Test [2] for implementing the test cases
- Java application launcher (java) [9] to execute the generated Java class files
- Java class file disassembler (javap) [9] to analyze the generated Java class files

Quality management

Documentation

- Report: The report contains all aspects of the project: the design of the library, its implementation, the design of the generator classes, their implementation, the grammar of the language, a description of the compiler, report of the tests and theoretical background for all the mentioned parts.
- User manual: The user manual is a short description of the interface, in addition to the contracts.
- Developer manual: The developer manual describes the ideas of the implementation of the library (in addition to the comments in the code) and lists possible extensions and the limits of the library.

Validation steps

Every Friday I will send Karine a short report of the work I have done during the week. I will also discuss my ideas and solutions with Karine and with Emmanuel Stapf to work towards the right goal.

4. PLAN WITH MILESTONES

Project steps

- Design a data structure to map the structures of a Java class file
- Implement the data structure
- Design classes to easily create Java byte codes (for example class generator, method generator, and so on)
- Implement classes
- Create a language and a corresponding compiler to test the framework as compiler backend
- Three presentations (introductory presentation, intermediary presentation, final presentation)

Parallel to all these points I will write the documentation.

Deadline

2003-09-25 Thursday
17 weeks and 4 days = 89 days

Tentative schedule

			2003-05-26	2003-06-02	2003-06-09	2003-06-16	2003-06-23	2003-06-30	2003-07-07	2003-07-14	2003-07-21	2003-07-28	2003-08-04	2003-08-11	2003-08-18	2003-08-25	2003-09-01	2003-09-08	2003-09-15	2003-09-22
	Start	End	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Thesis	22	39	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
Introductory presentation	23	23		█																
Intermediary presentation	32	32											█							
Final presentation	38	39																	█	█
Final corrections	38	38																	█	
Reserve, unexpected events	36	37															█	█		
Report																				
Writing report	22	35	█	█	█	█	█	█	█	█	█	█	█	█	█	█				
Writing user manual	29	35								█	█	█	█	█	█					
Writing developer manual	26	35				█	█	█	█	█	█	█	█	█						
Bytecode generator																				
Designing interface to Java bytecodes	22	22	█																	
Implementing interface	23	25		█	█	█														
Designing generator classes	27	27						█												
Implementing generator classes	28	29							█	█										
Creating test-language	31	31										█								
Implementing compiler	32	33											█	█						
Testing the whole software	34	34													█					

REFERENCES

- [1] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman: *Compilers: Principles, Techniques and Tools*, Addison-Wesley, 1986.
- [2] Eric Bezault: *Gobo Eiffel Project*, Online at <http://www.gobosoft.com>, consulted in April 2003.
- [3] *Byte Code Engineering Library (BCEL)*, Online at: <http://jakarta.apache.org/bcel/>, consulted in May 2003.
- [4] Chair of Software Engineering: *Semester-/Diplomarbeiten*, Online at: <http://se.inf.ethz.ch/projects/index.html>, consulted in March 2003.
- [5] James Gosling, Bill Joy, Guy Steele, Gilad Bracha: *The Java Language Specification, 2nd edition*, Addison-Wesley, 2000, <http://java.sun.com/docs/books/jls/>.
- [6] Tim Lindholm, Frank Yellin: *The Java Virtual Machine Specification, 2nd edition*, Addison-Wesley, 1999, <http://java.sun.com/docs/books/vmspec/>.
- [7] Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.
- [8] Seven Muchnick, *Advanced Compiler Design and Implementation*, Morgan Kaufmann Publishers, 1997.
- [9] Java 2 Platform, available from <http://java.sun.com>.