CS212 Part 2

Fall 2005

Part 2 of the project consists of two seperate parts. You must first plan out your compiler and submit a design document that outlines your project. You must then implement the compiler.

1 Tasks

1.1 Part 2A: Design Documentation and Review

To help prevent you from procrastinating and enforce better design practices, we require all students to develop, submit, and explain their plans for their code in Part 2B (below). The document will have the following structure:

- Title page: Project title, client (which is CS212), authors (including Net-IDs), date.
- Description of proposed class structure and parsing: Explain your proposed structure for Part 2 in words (about 1 page). You must also explain how you will parse supplied Bali code into Samcode. Refer to the class stubs (next section) to help explain the structure.
- Class headers: Write the proposed class headers.
- Example: On the next page, provide an example of a small, but good, Bali program to test your compiler for Part 2B. Use the supplied Part 2 grammar. Explain why you chose this example.
- AST: Draw the AST for your example.

Additional specifications:

- one-inch margins
- page numbers (title page is page #1, though the page number is not shown)
- all text except code and filenames: double-spaced, 12-point font
- code: fixed-width style font (e.g., Courier New)
- Use one these formats: PDF, ASCII text, or Microsoft Word.

You (and your group) must also schedule a 10-minute meeting with the teaching assistants to review your plans. Details of how to sign up will be announced in lecture and on the newsgroup and website.

1.2 Part 2B: Compiler

In Parts 2 and 3, you will write a Bali compiler that translates Bali code into valid Samcode, using the posted grammar. For Part 2B, we only require that you meet a subset of the syntax and semantics, which is summarized below:

- All code is contained in the main function.
- No function calls.
- No pointers.

So, Part 2 focuses on ALU and control structures. You are welcome to extend your work to include further elements of Bali. You may use any of the Samcode instructions.

- If you do not attempt to tackle functions,
  - You must use relative addressing.
  - Keep the FBR at zero.
  - Do not change or generate code for the FBR.
- Your compiler must use an AST as an intermediate stage between parsing and code generation.
  - The entire AST must be constructed before any code generation begins.
  - The different node types of the AST should correspond to separate classes. Do not confuse this requirement with the fact that all AST classes should be subclasses of a single AST interface or single AST abstract class.
- Your compiler must use a symbol table to keep track of variable locations and variable types.
- You must provide a class called `BaliCompiler` (in file `BaliCompiler.java`) that serves as the starting point for your compiler:
  - `BaliCompiler` must implement the `CS212Compiler` interface.
  - We provide a stubbed version of `BaliCompiler.java`, which you may choose to use.
  - Our `BaliCompiler.java` demonstrates setting up a `PrintWriter` to produce your code as a `String`, which the `CS212Compiler` interface requires.
- You must submit all other Java programs that constitute the classes and interfaces that you use to write your compiler, which will have whatever names your choose. Only the starting point (`BaliCompiler.java`) has a specified name.
- The code you submit should not include a `main` function. We will be testing your compiler by using the method provided by the `CS212Compiler` interface. We supply a main class `BC.java` which you should use to test your compiler. Your compiler is required to work with the provided `CS212Compiler` class and you may not submit a different version of that class. When you create your submission using AMS, several checks are also run on your code to make sure it is compliant with these specifications.
- Your compiler should not produce any output either to a file or to the standard output or error file. It is OK for your program to produce such output while you are debugging, but any such debugging output must be turned off in your submitted version.
- You must generate Samcode that represents a valid translation of the supplied Bali program.
  - Your generated Samcode does not need to contain any comments.
  - Remember to test your Samcode in the SaM Simulator.
- You must adhere to some programming specifications:
  - Your code must comply with the Java 5 specification.
  - You must comment your code and maintain tenets of good programming style.

1.3 Error Handling

There are two kinds of errors that can occur in Bali programs:

- *syntax errors*: code that violates the rules of a language's grammar.
- *semantic errors*: code that violates the rules of language's semantics.

For example, a missing semicolon or an unmatched parenthesis is a syntax error, whereas a type mismatch or an undeclared variable is a semantic error. For this part of the project you will only need to handle syntatical errors.

Rather than printing an error message to `System.out` (or `System.err`), errors in supplied Bali programs can be handled most clearly and cleanly by using *exceptions*. We have supplied exception classes corresponding to the two kinds of errors that can occur:

- `BaliSyntaxException` for syntax errors.
- `BaliSemanticException` for semantic errors.
- `IllegalBaliException`, an abstract root class for both the syntatic and semantic exceptions.

The `SamTokenizer` includes several ways to examine/consume tokens. Depending on how you write your code, you may also generate a `TokenizerException`, which is a runtime exception thrown by the tokenizer when various mismatches occur. If you have correctly caught all syntatical errors, your compiler should not generate `TokenizerException`s.

`BC.java`, the main program that runs your compiler, is designed so that it catches `IllegalBaliException`s and then prints the exception's message.

2 Submission Specifications

Part 2A:

- Refer to the problem statement for the format of the document.

Part2B:

- You must use the AMS submission application provided through the *Source* link on CMS. Refer to the CS212 website for information on using AMS.
- AMS will check your code to make sure it compiles and that your `BaliCompiler` class implements the `CS212Compiler` interface, and is marked *public*.
- Do not submit any of the files that we have provided. We will compile your code and run it with the reference copies posted on the website.
- Do not include extraneous text in your programs, such as debugging statements or superfluous comments.
- Include any special features or issues about your code that you need to communicate to the grading staff in the readme section of AMS.

3 Academic Integrity

You may work in groups of 1, 2, or 3 people. Refer to the course website for further rules on partners in Academic Integrity. You are responsible for knowing, understanding, and following these rules.