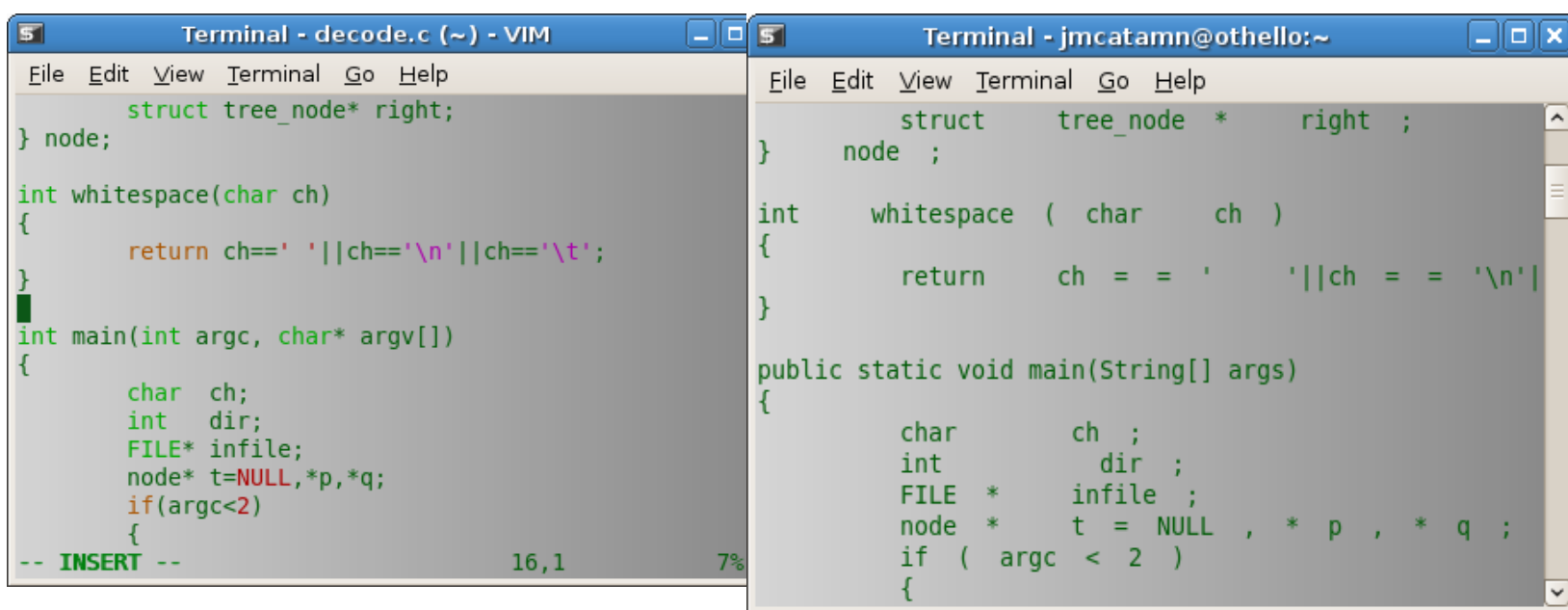# Programming Language Translation

Jamie McAtamney
TJHSST Computer Systems Lab
2007-2008

Abstract

With the modern emphasis on program portability and the new need to run programs on multiple computers in networks or over the Internet, it would be very useful for C programmers to be able to translate either "legacy" C programs or newly written programs into Java to make them more portable; however, currently translation by hand is seen as too tedious and time-consuming, while computer algorithms to do so are not very accurate. A combination of keyword search/replace and algorithms to translate C structs to Java classes and C "include" modules to Java "import" modules can help alleviate or solve the problem of tedious or inaccurate translations specifically between C and Java.
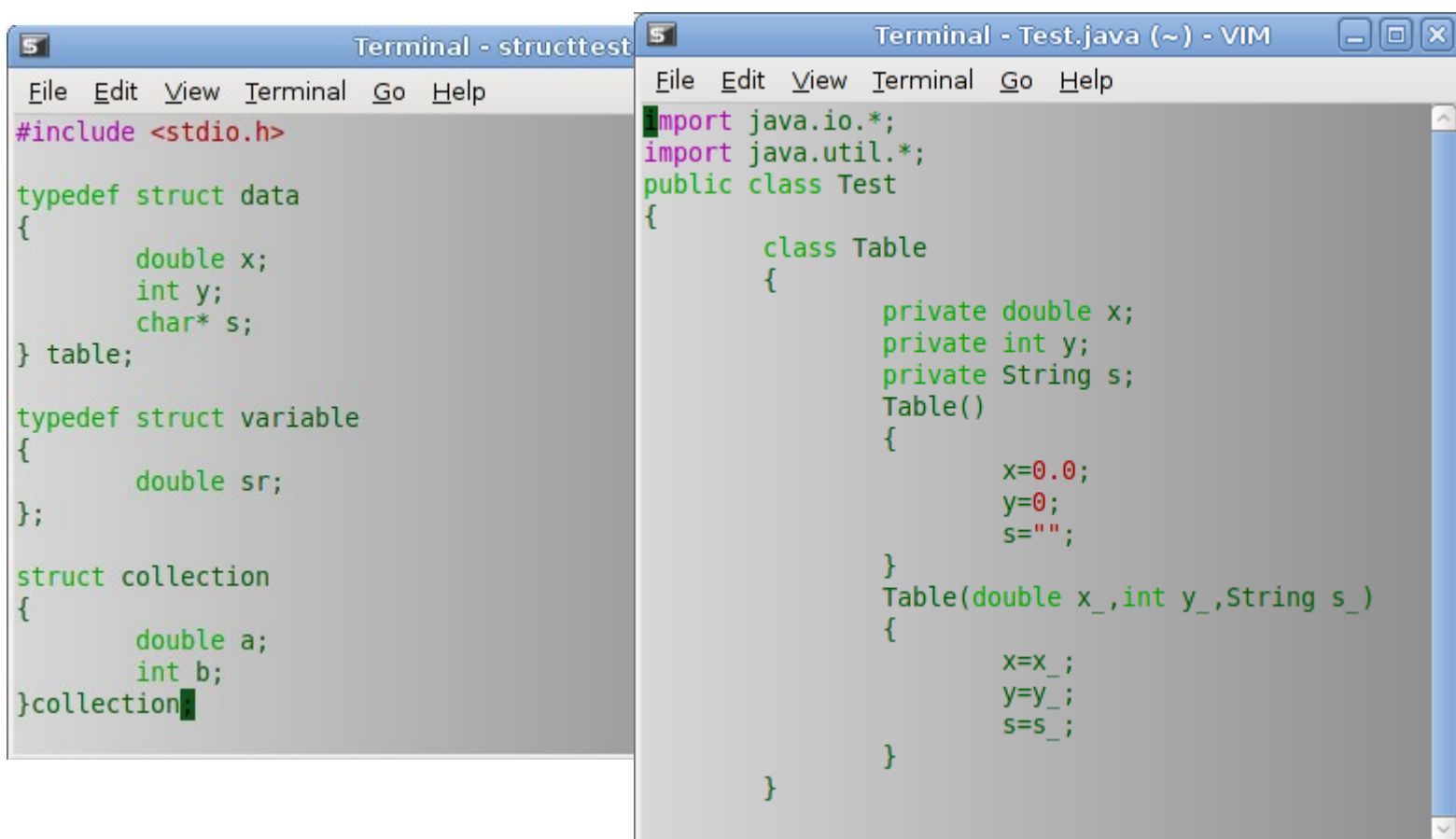
The first step of the translation process from C to Java: The original file (left) is read in and tokenized, and any syntax-independent lines—such as the main() method—are translated (right).
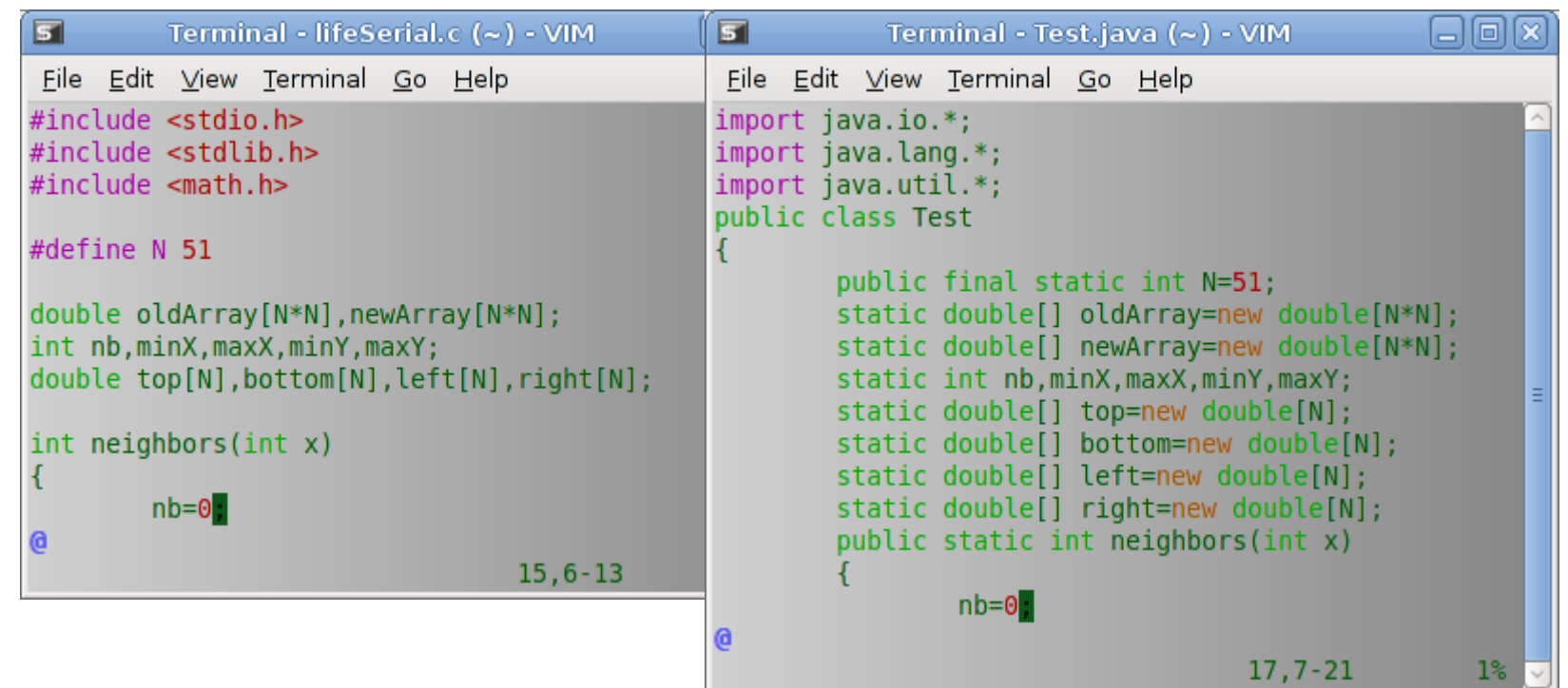
Structs in the original file (left) are translated to inner classes, with one example given above (right). Note the addition of two constructors for the inner class.

Background

While the differences among programming languages have been studied extensively in comparative languages courses and otherwise, little progress has been made in the area of automated programming language translation. The problems involved with automated translation occur because programming languages are too dissimilar for direct word-for-word translation. Even syntactically similar languages such as C and Java have differences that make simple search-and-replace difficult. For example, while the C char arrays have an analog in Java Strings, because they are two different data structures the methods for accessing them are very different, and this discrepancy must be taken into account. A related difficulty is C's use of pointers: A "string" in C is not simply an array of chars, it is a pointer to an array of chars—expressed as "char*"—which means that one cannot simply copy, compare, or otherwise manipulate strings in the same way one may manipulate ints or chars.

After the translation process: The original file (left) has been translated (right). Note the C #include statements changing to Java import statements, C array declarations changing to Java array constructors, and the addition of a class declaration.

Progress and Results

At this point, several translation modules have been implemented:
- Translates primitive types, such as char* to String
- Translates C preprocessor directives ("#include" to "import" and "#define" to variable declaration)
- Translates C array declarations to Java array constructors
- Translates input and output methods and structures such as input/output streams and files
- Translates method and module headers
- Translates most packages: math, string, stdlib, stdio, and more
- Handles throwing Exceptions
- Determines whether FILE*s are "input" (should become Scanners) or "output" (should become PrintStreams) FILE*s
- Determines whether PrintStreams should use print() or append() based on whether they are opened in C for writing or appending
- Translates basic graphics commands from OpenGL to JOGL
- Translates C structs to Java inner classes
- Translates pointers