

## Assignment 1

# Transitioning from JAVA to C++

**Written Part Due:** 11:59 PM, Wednesday, September 13, 2006

**Coding Part Due:** 11:59 PM, Wednesday, September 13, 2006

**Review Part Due:** 5:00 PM, Friday, September 15, 2006

## Preliminaries

Make sure you've done all the assigned reading before beginning this assignment.

Remember, you should work on this entire assignment as a pair, including the written part. You can read the assignment and think about it before you meet, however.

Get the files for this assignment by executing `svn update`. This command should check out the following files and directories:

<code>HW1-Answers.txt</code>	<code>shuffler/LineShuffler.java</code>
<code>HW1-Rubric.txt</code>	<code>shuffler/shuffle.cpp</code>
<code>palindromes/Makefile</code>	<code>shuffler/lineshuffler.cpp</code>
<code>palindromes/Palindrome.java</code>	<code>shuffler/lineshuffler.hpp</code>
<code>palindromes/palindrome.cpp</code>	<code>shuffler/random.cpp</code>
<code>shuffler/Makefile</code>	<code>shuffler/random.hpp</code>

The `palindromes` directory contains C++ and Java implementations of a program that determines whether a word entered by the user is a palindrome (the same forwards as backwards). Both programs can be built by entering the `palindromes` directory and typing `make` (the same command will rebuild the executables if you edit the files). The C++ version can be run by typing `./palindrome`; the Java version can be run by typing `java Palindrome`. The two versions should operate identically.

The `shuffle` directory contains C++ and Java implementations of a program that reads several lines from the user and the outputs them in random order. Both programs can be built by entering the `shuffle` directory and typing `make`. The C++ version can be run by typing `./shuffle`; the Java version can be run by typing `java LineShuffler`. Both implementations should operate identically.

You will use the file `HW1-Answers.txt` for your answers to the written questions. Your grade for this assignment will be determined using the rubric in the file `HW3-Rubric.txt`.

## Written Part

Answers to the questions in this part should be one sentence (or, in some cases, one line of code) only. You can make modifications to the code for the assignment to

answer these questions, but should always return the code to its previous (working) state before moving on to the next question.

W1. The C++ version of the palindrome program does not define a class, whereas the Java version does (defining a class called *Palindrome*).

- (a) When the Java version executes, how many objects of class *Palindrome* are created? (If you are not sure, you could provide a constructor for the class that outputs a message and count how many messages appear.)
- (b) The C++ version could have been written to place most of the code inside a class like the Java version, but wasn't. Why?

W2. When building the palindrome program (by typing `make`),

- (a) What command line is executed by `make` to compile the C++ code?
- (b) What do each of the command-line options mean? (You may want to type `man g++` to look up information, but if you do, remember that you don't need to read the entire manual page.)

W3. Both the C++ and Java versions of the palindrome program use a string class provided by the language (*string* in C++; *String* in Java). Based on the code for the palindrome program alone,

- (a) What string method behaves exactly the same in C++ and Java?
- (b) What string operation exists in both C++ and Java but is accessed differently?

W4. What is the the C++ equivalent to the following Java code?

```
System.out.println("Hello World!");
```

W5. Rearrange the C++ file `palindrome.cpp` to place the function definition of `main` above the definition of the `isPalindrome` function.

- (a) What compiler error is given by the compiler (`g++`) when you run `make`?
- (b) Would the Java compiler give an error if a similar rearrangement were made in the Java source?
- (c) Why doesn't C++ handle the code rearrangement the same way as Java?
- (d) Adding a *function declaration* for `isPalindrome` can eliminate the error. What is the correct declaration?

W6. In the line-shuffling example, the Java code (in `LineShuffler.java`) contains the line

```
LineShuffler randomLines = new LineShuffler();
```

whereas the C++ implementation (in `shuffle.cpp`) achieves similar results using

```
LineShuffler randomLines;
```

- If you change the Java code for this declaration so that it looks like the C++ code and recompile the program, what error message does the compiler give?
- What does this error message mean?
- What error message does the compiler give if you change the C++ code to

```
LineShuffler randomLines = new LineShuffler;
```

(Unlike Java, C++ can omit the “()” in **new** when the constructor takes no arguments.)

- What does this error message mean?

W7. What does the line **using namespace std**; do?

W8. In Java, a class is defined by a single file (ending with `.java`), but in C++ a class is usually defined by two files, a *header file* (ending in `.hpp`) and an *implementation file* (ending in `.cpp`).

- What does the line `#include "lineshuffler.hpp"` actually do?
- Why doesn't `shuffle.cpp` contain the line `#include "random.hpp"`?
- Why do the names of all the member functions (a.k.a. methods) defined in the implementation file begin with `LineShuffler::`?

W9. In the header file `lineshuffler.hpp`, the class declaration ends with `};` rather than just with `}`.

- Modify the class declaration so that it ends with `} myShuffler;` and recompile the code. The compiler should pass the code as being correct, even though we would consider it bad coding style in CS 70 for a couple of reasons. (The linker, on the other hand, will generate an error when it tries to make an executable from the object files produced by the compiler.)
  - What does this code mean?  
N.B. You almost certainly won't find an answer to this question in your textbook. Investigate for yourself. Try `x`, `y`, `z` instead of `myShuffler`. Consider what it *could* mean, and then test your guess.
  - Aside from the odd-looking syntax, why is this code bad style?

- (b) Modify the class declaration so that it ends with `}` (i.e., no closing semicolon) and attempt to recompile two of the source files individually by typing `make shuffle.o` and `make lineshuffler.o`.
- What errors does the compiler report for each of these files?
  - One of the files gives a sensible error message, whereas the other file gives a rather obscure message. This latter error shows the compiler is fairly confused. What does it think you are trying to do and why?

W10. The `count_` data member of the *LineShuffler* class is redundant. Explain why by indicating how the number of lines stored in the *LineShuffler* can be quickly determined.

## Coding Part

The palindrome program has several limitations, the foremost being that `isPalindrome` checks only for exact character matches. Thus while it happily confirms that words like “repaper”, “rotator”, “kayak”, “civic”, and “noon” are palindromes, it spurns palindromic phrases and sentences such as “mirror rim”, “stack cats”, “I prefer pi.”, “A man, a plan, a canal; Panama!” and “A man, a plan, a canoe, pasta, heros, rajahs, a coloratura, maps, snipe, percale, macaroni, a gag, a banana bag, a tan, a tag, a banana bag again (or a camel), a crepe, pins, Spam, a rut, a Rolo, cash, a jar, sore hats, a peon, a canal; Panama!” because (1) the palindrome test is case sensitive, and (2) the palindrome test does not ignore non-letters (spaces, punctuation, etc.)

Fix these deficiencies and submit your revised program, keeping in mind both correctness and elegance.

There are two functions provided by C++ (accessible after adding the line `#include <cctype>` to the `#includes` at the top of your file) that may be helpful:

```
char tolower(char c);           // Returns c, converted to lower case.
bool isalpha(char c);          // Returns true, if c is a letter.
```

## Review Part

After the assignment deadline for code/written answers, running `svn update` will reveal the sample solution to the assignment and any test cases we use. Based on this information, review your code, describe any errors you made, and determine what grade you would give yourselves on the assignment.

## Reminders

Remember to check in your code regularly! And in particular, don't forget to check in your code when you're finished.