COLBY COLLEGE
Computer Science Department

CS231 **Allen B. Downey**
Fall 1999

Assignment 5: Stacks

Due: Friday 22 October
Reading: Standish Chapter 6.1 – 6.6.

File I/O

The following program demonstrates basic file input in Java.

```
import java.io.*;

public class Balance {

  public static void parseLine (String line) {
    System.out.println (line);
  }

  // checkFile: reads each line of the given file and passes it to parseLine
  public static void checkFile (String filename)
                    throws FileNotFoundException, IOException {
    String s;

    // open the file and create a BufferedReader for it
    BufferedReader in =
      new BufferedReader (
         new FileReader (filename));

    // use the BufferedReader to get lines from the file
    while (true) {
      s = in.readLine();
      if (s == null) break;
      parseLine (s);
    }
  }

  public static void main (String[] args)
                    throws FileNotFoundException, IOException {
    checkFile ("Balance.java");
  }
}
```

1. Type in or cut and paste the above program, and run it. It should print out its own source code file.

2. Modify this program so that it prints each line backwards. This is easy to do because the elements of a `String` are indexed and we know how to traverse things backwards.

3. Modify this program so that it prints the lines in the file in reverse order. This is not as easy to do because the lines we read from a file are not indexed. They are **streamed**.

   One possibility is that we could index the lines of the file by putting them into an array, but we should not assume that we know ahead of time how many lines there are in the file. Thus, a better solution might make use of what the book calls a "linear data structure."

Balance checking

Starting with the code from the previous section, we would like to write a program that reads a Java source code file and that checks whether the the parentheses, braces, and brackets in the program are balanced.

1. Write a paragraph or two explaining how you would use the Stack ADT to check balance. Why isn't it enough to just count the number of open and close parentheses and make sure they match? Why is a stack well-suited to this task?

2. Once you have an algorithm, make a plan that details at least three intermediate stages between where the program is now and where it is going. Each stage should be a small modification of its predecessor, it should use a new feature or add a new function, and it should be testable.

3. After we have discussed the planning stage in class, go ahead and implement it.