EJayInterpreterNoFuncsTest.java

```java
import java.io.*;
import java_cup.runtime.*;

/****
 *
 * Simplified test program for CSC 330 Assignment 4.  This does what the full
 * EjayInterpreterTest class does, but instead of constructing a parse tree for
 * a call to the main method, it calls EJayInterpreter.eval on the statements
 * part of the body of main method.  In doing things this way, this simplified
 * test program can test a version of the interpreter that does not have
 * function calls implemented, but does do statements and expressions.
 *
 * This test program also dumps the parse tree and symbol table, prior to
 * dumping the interpreter’s execution memory.
 *
 */
public class EJayInterpreterNoFuncsTest {

    /**
     * See the class comment for documentation.
     */
    public static void main(String[] args) {
        TreeNode tree;
        SymbolTable symtab;
        try {
            EJayParser parser = new EJayParser(
                new EJayLexer(new FileReader(args[0])));

            parser.initSymbolTable(500);
            tree = (TreeNode) parser.parse().value;
            System.out.println(tree);
            System.out.println();
            System.out.println(symtab = parser.getSymbolTable());

            EJayInterpreter interp =
                new EJayInterpreter(symtab.memorySize, 10000);
            interp.eval(grabMainBody(symtab), symtab);
            interp.dumpMemory();
        }
        catch (Exception e) {
            System.out.println("Exception " + e);
            e.printStackTrace();
        }
    }

    /**
     * Extract the executable statements list from the body of the main
     * method.
     */
    protected static TreeNode grabMainBody(SymbolTable symtab) {
        FunctionEntry entry = (FunctionEntry) symtab.lookup("main");
        TreeNode2 body  = (TreeNode2) entry.body;
        TreeNode stmts = body.child2;
        return stmts;
    }

}
```