Homework Turnin

Email:	rgalanos@fcps.edu
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	02-05
Receipt ID:	45280e89df51259e4e0913f697d134e2

Turnin Successful!

The following file(s) were received:

ExpressionEvaluator.java (4076 bytes)

```
//Horstman, _Java Concepts for AP Computer Science_, p. 611-117
import java.util.*;

/**
   This program calculates the value of an expression
   consisting of numbers, arithmetic operators, and parentheses.
*/
public class ExpressionEvaluator   {
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter an expression: ");
      // 3+4*5              23
      // (3+4)*5            35
      // (4+5)-5*3          -6
      // (3+4)*(5+6)        77
      // (3*(4+5)-2)/5       5
      //  2*3*4-9/3         21

      String input = in.nextLine().trim();
      Evaluator e = new Evaluator(input);
      int value = e.getExpressionValue();
      System.out.println(input + " = " + value);
   }
}
   ////////////////////////////////////////////////////////////
//name:    date:
class Evaluator
{
   private ExpressionTokenizer tokenizer;

   public Evaluator(String anExpression)
   {
      tokenizer = new ExpressionTokenizer(anExpression);
   }
   /**
      Evaluates the expression.
      @return the value of the expression.
   */
   public int getExpressionValue()
   {
      int value = getTermValue();
      String next = tokenizer.peekToken();

      if("+".equals(next))
      {
        tokenizer.nextToken();
         value = value + getExpressionValue();
      }
      else if("-".equals(next))
      {
          tokenizer.nextToken();
          value = value - getExpressionValue();
       }

       return value;
    }

    /**
       Evaluates the next term found in the expression.
       @return the value of the term
    */
    public int getTermValue()
    {
       int value = getFactorValue();
       String next = tokenizer.peekToken();

       if("*".equals(next))
       {
         tokenizer.nextToken();
          value = value * getTermValue();
       }
       else if("/".equals(next))
       {
         tokenizer.nextToken();
          value = value / getExpressionValue();
       }

       return value;
    }

    /**
       Evaluates the next factor found in the expression.
       @return the value of the factor
    */
    public int getFactorValue()
    {
       int value;
       String next = tokenizer.peekToken();
       if ("(".equals(next))
       {
          tokenizer.nextToken(); // Discard "("
          value = getExpressionValue();
          tokenizer.nextToken(); // Discard ")"
       }
       else
       {
          value = Integer.parseInt(tokenizer.nextToken());
       }
       return value;
    }
 }
    //////////////////////////////////////
    //Horstman, _Java Concepts for AP Computer Science_, p. 611-117
    /**
    This class breaks up a string describing an expression
    into tokens: numbers, parentheses, and operators.
    */
 class ExpressionTokenizer
 {
    private String input;
    private int start; // The start of the current token
    private int end; // The position after the end of the current token

       /**
       Constructs a tokenizer.
       @param anInput the string to tokenize
       */
    public ExpressionTokenizer(String anInput)
    {
       input = anInput;
       start = 0;
       end = 0;
       nextToken(); // Find the first token
    }

       /**
       Peeks at the next token without consuming it.
       @return the next token or null if there are no more tokens
       */
    public String peekToken()
    {
      if (start >= input.length()) {
         return null; }
      else {
         return input.substring(start, end); }
   }

   /**
      Gets the next token and moves the tokenizer to the following token.
      @return the next token or null if there are no more tokens
   */
   public String nextToken()
   {
      String r = peekToken();
      start = end;
      if (start >= input.length()) {
         return r; }
      if (Character.isDigit(input.charAt(start)))
      {
         end = start + 1;
         while (end < input.length()
               && Character.isDigit(input.charAt(end)))
         {
            end++;
         }
      }
      else
      {
         end = start + 1;
      }
      return r;
   }
}
```