Informatik 2 Group 2

Lab 7: Fun with Calculators 2

Bruno Dinis (570637)& Bartholomäus Berresheim (s0568624)

Index

Introduction

In this weeks lab, our task was to combine our calculator from the fifth lab and our work from the sixth lab, so that in the end we have a calculator that takes the precedence of operators into account.

Lab exercises

1. Make a copy of one of your Calculators. Make sure that it works before you begin! If neither of you got a Calculator to work, ask colleagues for permission to use theirs. Give them credit in your report!

We made a copy from Bartholomäus's Lab 5 calculator, which was working as intended.

2. Rework it to accept a long String of single digits separated by operators that have precedence. The bored may use multi-digit numbers and floating-point or scientific notation, if they please.

To accomplish the task we were given, we had to rework most of the CalcEngine methods and the redisplay() method from the UserInterface class. We started by creating a new protected String field called "calculation" in CalcEngine. We then proceeded by changing the numberPressed() method so that it would add the inputted number to "calculation"

```
public void numberPressed(int number)
{
    calculation += number;
}
```

After that, we changed the operator methods such as the multiplication() method, so that they would add the corresponding operator to "calculation".

```
/**
 * The 'multiplication' button was pressed.
 */
public void multiplication()
{
    calculation += "*";
}

/**
 * The 'division' button was pressed.
 */
public void division()
{
    calculation += "/";
}

/**
 * The 'plus' button was pressed.
 */
public void plus()
{
    calculation += "+";
}

/**
 * The 'minus' button was pressed.
 */
public void minus()
{
    calculation += "-";
}
```

We then proceeded by changing the clear() and redisplay() methods, so that clear() would set "calculation" to "" and redisplay() would display "calculation" by using the setText() method from the JTextField class.

```
public void clear()
{
    calculation = "";
}


protected void redisplay()
{
    display.setText("" + calc.calculation);

}
```

Now if we press the buttons on the calculator, all the corresponding operands or operators get added to "calculation", which in turn gets displayed in the calculators display.

Example:

After pressing '4' '*' '6' '+' '9' '/' '3' '-' '1'

We get:

3. Once you get a String input, add in calls to convert this expression to postfix and evaluate the postfix when = is pressed. Presto, your calculator now takes care of operator priorities, like magic! Just a little bit of mathematical thought, and you can introduce new functionality!

The first thing we did for to complete this task, was to make a copy of Bartholomäus's classes from the Reverse Polish Notation lab and put it in our current package.

We then proceeded by creating a Postfix Object called "postcalc" in our CalcEngine class. After that we reworked the equals() methods from CalcEngine. In the equals() method, we created a new double called "result" and set its value to the result from the Postfix evaluate() method, that had the infixToPostfix() method, with "calculation" as input, as

parameters. Then we converted result to String and used it as value for calculation.

Since the Postfix methods throw StackUnderflow and StackException, we put the previously mentioned expressions into a try and catch.

```
public void equals()
{
    try {
        double result = postcalc.evaluate(postcalc.infixToPostfix(calculation));
        calculation = Double.toString(result);
    } catch (StackUnderflow | StackException e) {
        e.printStackTrace();
    }
}
```

Now if we press the ‘=’ button on the calculator, the String “calculation” is used by the Postfix methods to calculate the result, then gets overwritten by it and displayed.

Example:

If we take the previously inputted ‘4’ ‘*’ ‘6’ ‘+’ ‘9’ ‘/’ ‘3’ ‘-’ ‘1’ and press ‘=’.

We get:

4*6+9/3-1 = 24+3-1 = 26

As we can see the calculator works as intended.

While testing the calculator to see if it was working properly, we noticed that we had to press “Clear” after a calculation, if we wanted to start a new one, instead of directly typing in the new expression. We found that to be rather tedious, so we chose to add an if-statement after the calling of the redisplay() method at the end of the actionPerformed() method. The if statement would check if ‘=’ was pressed and then call the clear() method that would set “calculation” to “”.

```
redisplay();
if(command.equals("=")) {
    calc.clear();
}
```

5. Check that this still works with hexadecimal numbers.

While working on the previous exercices, we noticed that hexadecimal numbers wouldn't work, since they get converted into multi digit integers before being added to "calculation" and since the Postfix methods don't work with multi digit, it would end in a failure when tested. This led to us completing this task (the fifth) before the fourth one.

To solve this problem, we changed the actionPerformed() method, so that when a number or A-F get pressed we set the value of a newly created char "c" to the first char from the "command" String, after that we call the numberPressed() method using "c". For this to work, we had to change the input type of the numberPressed() method from int to char.

```
else if(command.equals("0") ||
        command.equals("1") ||
        command.equals("2") ||
        command.equals("3") ||
        command.equals("4") ||
        command.equals("5") ||
        command.equals("6") ||
        command.equals("7") ||
        command.equals("8") ||
        command.equals("9")) {
    c = (char) (command.charAt(0));
    calc.numberPressed(c);
}
else if(command.equals("A") ||
        command.equals("B") ||
        command.equals("C") ||
        command.equals("D") ||
        command.equals("E") ||
        command.equals("F")) {
    c = (char) (command.charAt(0));
    calc.numberPressed(c);

public void numberPressed(char number)
{
    calculation += number;
}
```

After that, we implemented some changes into the Postfix methods, so that they would recognize A-F as operands. For that we added an OR condition to the if statement that checks for operands at the beginning of both the infixToPostfix() and the evaluate() methods.

```
if ((token > 47 && token < 58) || (token >64 && token<71))
```

Now the letters get treated just like the other operands. They stay as letters throughout the infix to postfix conversion, and only during the

evaluation they get turned from char to integers using the getNumericValue() method. Since we already had implemented this method in the previous lab, there was no need to make any changes.

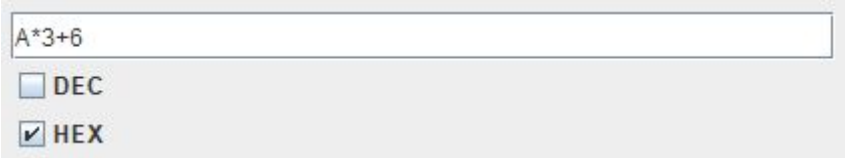
To finish things off, we changed the redisplay() method, so that it turns the final result into Hexadecimal numbers if the corresponding checkbox is checked. For that we added an if statement that would check if the hexaPanel is visible and if the last pressed button was the '=' one. If the conditions are met, the value of a newly created double called "calcD" is set to the double value of "calculation" using the valueOf() method. We then cast the double into an int and set it as value for a newly created int called "calcInt". After that we use "calcInt" to call the toHexString() method, the result of which is then set to upper case using the toUpperCase() method. The final result is then set as value for "calculation".

```
protected void redisplay()
{
    if(hexaPanel.isVisible() == true && command.equals("="))
    {
        double calcD = Double.valueOf(calc.calculation);
        int calcInt = (int) calcD;
        calc.calculation = (Integer.toHexString(calcInt)).toUpperCase();
    }

    display.setText("" + calc.calculation);

}
```

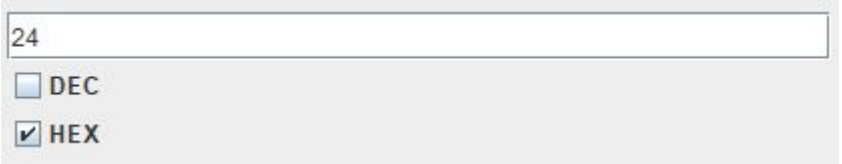
Now if the HEX checkbox is checked and we input an expression, the hexa digits get displayed as letters and the result is also in hexadecimal.

<u>Example:</u>

If we input 'A' '*' '3' '+' '6'



We get:



A*3+6 = 10*3+6 = 30+6 = 36

24 = 2*16 + 4*1 = 32+4 = 36

As we can see, it works as intended.

4. What will be good test cases for the precedence of * over +? Find 15 good test cases and try them out, documenting the results.

To complete this task, first deliberated about what test cases to use and in the end we came up with the following:

"1+7*5", "F-6*5", "6/3+9*2", "3+7*5/4", "A+C/6", "D*8-6*5",

"6+6*6-6/6", "2/A+3*5+F", "F-3*5+A/4", "1-B+8/5-2", "3+3*5+F/A",

"1*2*7-4-B/8-9*E*E", "4+8*7-4-F/2+5*C/3" and

"1-2*3+4/5/6+7*8-9*A+B/C+D*E-F".

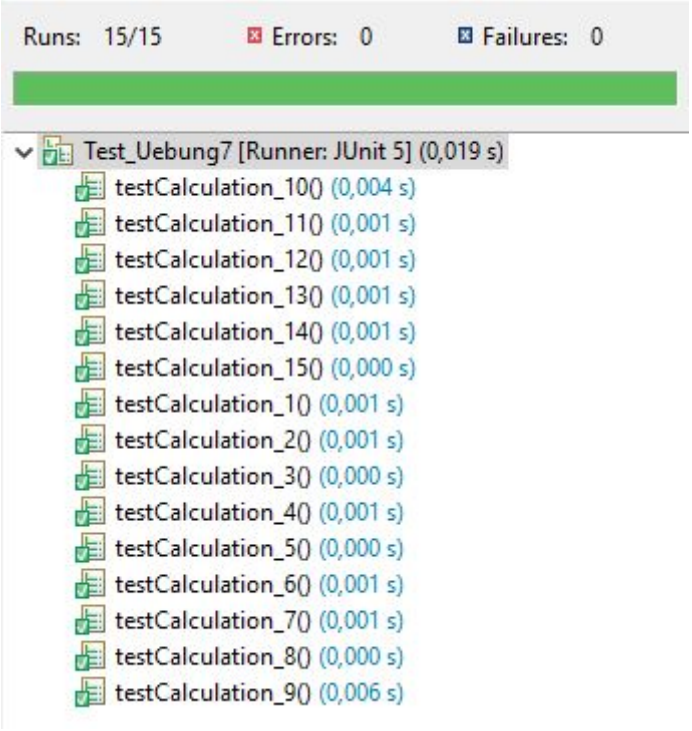
We then created a new JUnit class called Test_Uebung7 in which we created 15 test methods, one for each of the test cases. And then we created one CalcEngine object called "engine" and set it as a field.

Since it would take a lot of space to show the code for each of the test methods, we will explain how we programmed them using one of them as an example, as for the other ones we will show them in the appendix.

For the first test case "1+7*5", we created a method called testCalculation_1(). In it we called the method numberPressed() with '1' as input. We then called the plus() method, followed by the numberPressed() method that had '7' as input. After that we called the multiplication() method, followed by the numberPressed() method with '5' as input. Then we called the equals() method and finally we compare the result with our predetermined result using the assertEquals() method.

```
CalcEngine engine = new CalcEngine();

@Test
void testCalculation_1() {

    engine.numberPressed('1');
    engine.plus();
    engine.numberPressed('7');
    engine.multiplication();
    engine.numberPressed('5');
    engine.equals();
    assertEquals("36.0",engine.calculation);    // 1+7*5
}
```

1+7*5 = 1+35 = 36



As we can see, for all of our test cases the predetermined result corresponded to the calculated one, which means that our calculator works as intended.

We tried for some of the test methods to get the results in hexadecimal numbers by calling the redisplay() method before calling the equals() method, so that "calculation" gets converted into hexadecimal, but instead we got a NullPointerException. This was due to the fact that the if statement in redisplay() that does the conversion to Hexadecimal, checks if the value of the String "command" is equal to '=' and returns false. This is the case because, in our test methods, we don't call the ActionPerformed method using ActionEvents but instead, we directly call the operand/operator methods that we need, which means that "command" never gets a value assigned to it.

In the end we managed to solve this problem by assigning '=' to "command" as its initial value. And by calling redisplay() over a HexUserInterface object, or else hexaPanel would remain null.

```
HexUserInterface gui = new HexUserInterface(engine);

@Test
void testCalculation_3() {

    engine.numberPressed('6');
    engine.division();
    engine.numberPressed('3');
    engine.plus();
    engine.numberPressed('9');
    engine.multiplication();
    engine.numberPressed('2');
    engine.equals();
    gui.redisplay();
    assertEquals("14",engine.calculation);  // 6/3+9*2
}
```

6/3+9*2 = 2+18 = 20
14 = 1*16 + 4*1 = 20

For the bored

1. Now include the exponentiation operator ^ (2^4 = 16). It has a higher priority than either multiplication or division.

To complete this task, we started by adding a new exponentiation button to the buttonPanel in the makeFrame() method.

```
addButton(buttonPanel, "^");
```

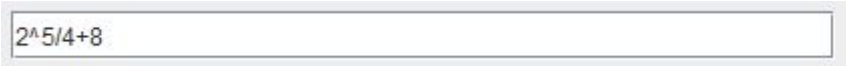
We created a new public void method called exponentiation() in the CalcEngine class. In it we add the exponentiation operator "^" to the String "calculation"

```
public void exponentiation()
{
    calculation += "^";
}
```

And finally we added a new else-if statement to the actionPerformed() method, that checks if command is equal to "^". If true, it then calls the exponentiation() method. Since we already had implemented the exponentiation operator in the Postfix class, in the previous lab, we didn't have to make any changes there.

Now if we try calculating using operators of each precedence, we get the correct result.

Example:
If we input ‘2’ ‘^’ ‘5’ ‘/’ ‘4’ ‘+’ ‘8’



We get:

2^5/4+8 = 32/4+8 = 8+8 = 16

As we can see, the exponentiation operator works as intended.

Reflection

Bruno:
This week’s lab was actually my first lab to almost get finished in class, which is very rare to happen. The exercices were pretty straight forward since both the codes we used were already functioning perfectly. For me it was good practicing to stitch two separate projects together since I’ve always had trouble with that. I must say this lab wouldn't have gone so well without having Barth’s code and without him being so well prepared.

Bartholomäus:
This week's lab was one of the rare ones where we managed to finish most of the tasks during class. This was probably due to the fact that we already had a pre existing code that I was familiar with, which meant we didn’t have to take our time to read through it to make heads or tails out of it, or we didn't have to start from scratch and think about how we should go on about coding. The most time consuming part was coming up with some good test cases.

Code

Since this labs code is a lot longer in comparison to the previous ones, we went through it and removed the parts that weren’t needed. Such as the calculateResult() method from the CalcEngine class, that has become obsolete due to the Postfix class.

```
package ubung_7;
/**
```

```
 * The main part of the calculator doing the calculations.
 *
 * @author  David J. Barnes and Michael Kolling
 * @version 2008.03.30
 */
public class CalcEngine
{

    protected String calculation = "";
    protected Postfix postcalc = new Postfix();

    /**
     * Create a CalcEngine.
     */
    public CalcEngine()
    {
        clear();
    }

    /**
     * A number button was pressed.
     * Either start a new operand, or incorporate this number as
     * the least significant digit of an existing one.
     * @param number The number pressed on the calculator.
     */

    public void numberPressed(char number)
    {
     calculation += number;
    }

    /**
     * The 'exponentiation' button was pressed.
     */
    public void exponentiation()
    {
     calculation += "^";
     }

    /**
     * The 'multiplication' button was pressed.
```

```
     */
    public void multiplication()
    {
     calculation += "*";
    }

    /**
     * The 'division' button was pressed.
     */
    public void division()
    {
     calculation += "/";
    }

    /**
     * The 'plus' button was pressed.
     */
    public void plus()
    {
     calculation += "+";
    }

    /**
     * The 'minus' button was pressed.
     */
    public void minus()
    {
     calculation += "-";
    }

    /**
     * The '=' button was pressed.
     */
    public void equals()
    {
     try {
                double result =
postcalc.evaluate(postcalc.infixToPostfix(calculation));
                calculation = Double.toString(result);
           } catch (StackUnderflow | StackException e) {
                e.printStackTrace();
```

```
            }
        }


        /**
         * The 'Clear' button was pressed.
         * Reset everything to a starting state.
         */
        public void clear()
        {
         calculation = "";
        }

        /**
         * @return The title of this calculation engine.
         */
        public String getTitle()
        {
            return "Java Calculator";
        }

        /**
         * @return The author of this engine.
         */
        public String getAuthor()
        {
            return "David J. Barnes and Michael Kolling";
        }

        /**
         * @return The version number of this engine.
         */
        public String getVersion()
        {
           return "Version 1.2";
        }
}
```

```java
package ubung_7;
/**
 * The main class of a simple calculator. Create one of these and
you'll
 * get the calculator on screen.
 *
 * @author David J. Barnes and Michael Kolling
 * @version 2008.03.30
 */
public class Calculator
{
    protected CalcEngine engine;
    protected HexUserInterface gui;

    /**
     * Create a new calculator and show it.
     */
    public Calculator()
    {
        engine = new CalcEngine();
        gui = new HexUserInterface(engine);
    }

    /**
     * In case the window was closed, show it again.
     */
    public void show()
    {
        gui.setVisible(true);
    }
}
```

```java
package ubung_7;
import java.awt.*;
import javax.swing.*;
/**
 * Write a description of class HexUserInterface here.
 *
 * @author (your name)
```

```
 * @version (a version number or a date)
 */
public class HexUserInterface extends UserInterface
{

    protected JPanel FcontentPane =
(JPanel)frame.getContentPane();

      public HexUserInterface(CalcEngine engine) {
            super(engine);
            makePanel();
            FcontentPane.add(hexaPanel, BorderLayout.WEST);
            frame.pack();
      }

    public void makePanel(){


        hexaPanel = new JPanel(new GridLayout(4, 2));
              addButton(hexaPanel, "A");
              addButton(hexaPanel, "B");
              addButton(hexaPanel, "C");
              addButton(hexaPanel, "D");

              addButton(hexaPanel, "E");
              addButton(hexaPanel, "F");

    }
}

package ubung_7;

public class Postfix {

      public Postfix() {
            // TODO Auto-generated constructor stub
      }

      char token;

      public String infixToPostfix (String infix) throws
```

```
StackUnderflow, StackException
	{
		String result ="";
		StackAsList stack = new StackAsList();

		infix = infix.replace("\\s+", "");

		for (int i = 0; i < infix.length(); i++)
		{
			token = infix.charAt(i);


			if ((token > 47 && token < 58) || (token >64 &&
token<71))
				{
					result += token;
				}

			else if (token == '(')
			{
				stack.push(token);
			}
			else if (token == ')')
			{
				while (stack.top() != (Character)'(')
				{
					result += stack.top();
					stack.pop();
				}
				stack.pop();
			}
			else if (isOperator(token))

			{
				while (!stack.isEmpty()
&&(!((precedence((char) stack.top()) < precedence(token) )||
						(token == '^' &&
precedence((char) stack.top()) == precedence(token) ))))
				{
					result += stack.top();
					stack.pop();
```

```
                    }

                    stack.push(token);
                }
                else {
                    throw new StackException("Wrong Input");
                }

            }

            while (!stack.isEmpty())
            {
                result += stack.top();
                stack.pop();
            }

            return result;
        }

        public boolean isOperator (char token) {

            if (token == '+' ||
                token == '-' ||
                token == '*' ||
                token == '/' ||
                token == '^')
                return true;

            else
                return false;
        }


        public int precedence (char token)
        {
            switch(token) {

            case '+':
            case '-':
                return 0;
            case '*':
```

```
            case '/':
                  return 1;
            case '^':
                  return 2;
            default:
                  return -1;

            }
      }

      public double evaluate (String pfx) throws StackUnderflow,
StackException {


            double rhs= 0;
            double lhs = 0;
            double tokenInt = 0;

            double result = 0;

            StackAsList stack = new StackAsList();

            pfx = pfx.replace("\\s+", "");

            for (int i = 0; i <pfx.length(); i++)
            {
                  token = pfx.charAt(i);

                  if ((token > 47 && token < 58) || (token >64 &&
token<71))
                  {
                        tokenInt = Character.getNumericValue(token);
                        stack.push(tokenInt);
                  }

                  else if (isOperator(token))
                  {

                        rhs = (double) stack.top();
                        stack.pop();
                        lhs =  (double) stack.top();
```

```
                        stack.pop();

                        if (token == '+')
                              {result = lhs + rhs;
                              stack.push(result);
                              }
                        else if (token == '-')
                              {result = lhs - rhs;
                              stack.push(result);
                              }
                        else if (token == '*')
                              {result = lhs * rhs;
                              stack.push(result);
                              }

                        else if (token == '/')
                              {result =  lhs /rhs;
                              stack.push(result);
                              }
                        else if (token == '^')
                              {result = (double) Math.pow(lhs, rhs);
                              stack.push(result);
                              }
                  }
                  else {
                        throw new StackException("Wrong Input");
                  }
            }

            return (double) stack.top();

      }
}
```

```
package ubung_7;

public class Run {

      public static void main(String[] args) {
            Calculator calculator = new Calculator();
            calculator.show();
```

```
	}

}
```

```
package ubung_7;

public interface Stack<E> {
	public void push (E item);
	public void pop () throws StackUnderflow;
	public Object top () throws StackUnderflow;
	public boolean isEmpty ();
	public void Empty ();
	public void print();
	public String toString();
	}
```

```
package ubung_7;

public class StackAsList implements Stack {
	private Node myStack;
	private class Node {

		public Node(Object data, Node next) {
			this.data = data;
			this.next = next;
		}
		public Node() {
			data = null;
			next = null;
		}
		Object data;
		Node next;
		public String toString() {
			return data.toString();
		}
	}



	public StackAsList() {
		myStack=null;
```

```
	}

	@Override
	public void push(Object item) {

		Node temp = new Node(item, myStack);
		myStack = temp;
	}

	@Override
	public void pop() throws StackUnderflow {
		if(!isEmpty())
		myStack = myStack.next;
	}

	@Override
	public Object top() throws StackUnderflow {
		// TODO Auto-generated method stub
		return myStack.data;
	}

	@Override
	public boolean isEmpty() {
		if(myStack == null) {
			return true;
		}
		else
			return false;
	}

	@Override
	public void Empty() {
		myStack = null;
	}

	@Override
	public void print() {
		System.out.println(this.toString());
	}
	@Override
	public String toString() {
```

```
            String S_myString = "";
            Node temp = new Node();
            temp = myStack;
            while (isEmpty()) {
                  if(temp.next == null) {
                        break;
                  }
                  temp = temp.next;
                  S_myString += temp.data + "\r\n";
            }
            return S_myString;
      }
}
```

```
package ubung_7;

public class StackException extends Exception
{
    public StackException( String message )
    {
       super( message );
    }
 }
```

```
package ubung_7;

public class StackUnderflow extends Exception {

}
```

```
package ubung_7;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class Test_Uebung7 {

      CalcEngine engine = new CalcEngine();
      HexUserInterface gui = new HexUserInterface(engine);

      @Test
```

```
	void testCalculation_1() {

		engine.numberPressed('1');
		engine.plus();
		engine.numberPressed('7');
		engine.multiplication();
		engine.numberPressed('5');
		engine.equals();
		assertEquals("36.0",engine.calculation);    // 1+7*5
	}

	@Test
	void testCalculation_2() {

		engine.numberPressed('F');
		engine.minus();
		engine.numberPressed('6');
		engine.multiplication();
		engine.numberPressed('5');
		engine.equals();
		assertEquals("-15.0",engine.calculation);   // F-6*5
	}

	@Test
	void testCalculation_3() {

		engine.numberPressed('6');
		engine.division();
		engine.numberPressed('3');
		engine.plus();
		engine.numberPressed('9');
		engine.multiplication();
		engine.numberPressed('2');
		engine.equals();
		gui.redisplay();
		assertEquals("14",engine.calculation);// 6/3+9*2
	}

	@Test
	void testCalculation_4() {
```

```
        engine.numberPressed('3');
        engine.plus();
        engine.numberPressed('7');
        engine.multiplication();
        engine.numberPressed('5');
        engine.division();
        engine.numberPressed('4');
        engine.equals();
        assertEquals("11.75",engine.calculation);   // 3+7*5/4
    }

    @Test
    void testCalculation_5() {

        engine.numberPressed('A');
        engine.plus();
        engine.numberPressed('C');
        engine.division();
        engine.numberPressed('6');
        engine.equals();
        assertEquals("12.0",engine.calculation);    // A+C/6
    }

    @Test
    void testCalculation_6() {

        engine.numberPressed('D');
        engine.multiplication();
        engine.numberPressed('8');
        engine.minus();
        engine.numberPressed('6');
        engine.multiplication();
        engine.numberPressed('5');
        engine.equals();
        assertEquals("74.0",engine.calculation);    // D*8-6*5
    }

    @Test
    void testCalculation_7() {

        engine.numberPressed('6');
```

```
        engine.plus();
        engine.numberPressed('6');
        engine.multiplication();
        engine.numberPressed('6');
        engine.minus();
        engine.numberPressed('6');
        engine.division();
        engine.numberPressed('6');

        engine.equals();
        assertEquals("41.0",engine.calculation);    // 6+6*6-6/6
    }

    @Test
    void testCalculation_8() {

        engine.numberPressed('2');
        engine.division();
        engine.numberPressed('A');
        engine.plus();
        engine.numberPressed('3');
        engine.multiplication();
        engine.numberPressed('5');
        engine.plus();
        engine.numberPressed('F');
        engine.equals();
        assertEquals("30.2",engine.calculation);    // 2/A+3*5+F

    }

    @Test
    void testCalculation_9() {

        engine.numberPressed('F');
        engine.minus();
        engine.numberPressed('3');
        engine.multiplication();
        engine.numberPressed('5');
        engine.plus();
        engine.numberPressed('A');
        engine.division();
```

```
        engine.numberPressed('4');
        engine.equals();
        assertEquals("2.5",engine.calculation);     // F-3*5+A/4
    }

    @Test
    void testCalculation_10() {
        engine.numberPressed('8');
        engine.plus();
        engine.numberPressed('7');
        engine.multiplication();
        engine.numberPressed('D');
        engine.division();
        engine.numberPressed('4');
        engine.equals();
        assertEquals("30.75",engine.calculation);   // 8+7*D/4
    }

    @Test
    void testCalculation_11() {

        engine.numberPressed('1');
        engine.minus();
        engine.numberPressed('B');
        engine.plus();
        engine.numberPressed('8');
        engine.division();
        engine.numberPressed('5');
        engine.minus();
        engine.numberPressed('2');
        engine.equals();
        assertEquals("-10.4",engine.calculation);   // 1-B+8/5-2
    }

    @Test
    void testCalculation_12() {

        engine.numberPressed('3');
        engine.plus();
        engine.numberPressed('3');
        engine.multiplication();
```

```
            engine.numberPressed('5');
            engine.plus();
            engine.numberPressed('F');
            engine.division();
            engine.numberPressed('A');
            engine.equals();
            assertEquals("19.5",engine.calculation);    // 3+3*5+F/A
      }

      @Test
      void testCalculation_13() {

            engine.numberPressed('1');
            engine.multiplication();
            engine.numberPressed('2');
            engine.multiplication();
            engine.numberPressed('7');
            engine.minus();
            engine.numberPressed('4');
            engine.minus();
            engine.numberPressed('B');
            engine.division();
            engine.numberPressed('8');
            engine.minus();
            engine.numberPressed('9');
            engine.multiplication();
            engine.numberPressed('E');
            engine.multiplication();
            engine.numberPressed('E');
            engine.equals();
            assertEquals("-1755.375",engine.calculation);    //
1*2*7-4-B/8-9*E*E
      }

      @Test
      void testCalculation_14() {

            engine.numberPressed('4');
            engine.plus();
            engine.numberPressed('8');
            engine.multiplication();
```

```
            engine.numberPressed('7');
            engine.minus();
            engine.numberPressed('4');
            engine.minus();
            engine.numberPressed('F');
            engine.division();
            engine.numberPressed('2');
            engine.plus();
            engine.numberPressed('5');
            engine.multiplication();
            engine.numberPressed('C');
            engine.division();
            engine.numberPressed('3');
            engine.equals();
            assertEquals("68.5",engine.calculation);    //
4+8*7-4-F/2+5*C/3
      }

      @Test
      void testCalculation_15() {

            engine.numberPressed('1');
            engine.minus();
            engine.numberPressed('2');
            engine.multiplication();
            engine.numberPressed('3');
            engine.plus();
            engine.numberPressed('4');
            engine.division();
            engine.numberPressed('5');
            engine.division();
            engine.numberPressed('6');
            engine.plus();
            engine.numberPressed('7');
            engine.multiplication();
            engine.numberPressed('8');
            engine.minus();
            engine.numberPressed('9');
            engine.multiplication();
            engine.numberPressed('A');
            engine.plus();
```

```
            engine.numberPressed('B');
            engine.division();
            engine.numberPressed('C');
            engine.plus();
            engine.numberPressed('D');
            engine.multiplication();
            engine.numberPressed('E');
            engine.minus();
            engine.numberPressed('F');

            engine.equals();
            assertEquals("129.05",engine.calculation);  //
1-2*3+4/5/6+7*8-9*A+B/C+D*E-F
      }
}

package ubung_7;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

/**
 * A graphical user interface for the calculator. No calculation
is being
 * done here. This class is responsible just for putting up the
display on
 * screen. It then refers to the "CalcEngine" to do all the real
work.
 *
 * @author David J. Barnes and Michael Kolling
 * @version 2008.03.30
 */
public class UserInterface
    implements ActionListener
{
    protected CalcEngine calc;
    protected boolean showingAuthor;
    protected JFrame frame;
    protected JTextField display;
    protected JLabel status;
```

```
    protected JPanel hexaPanel;
    protected ButtonGroup buttonG = new ButtonGroup();

    private String command = "=";
    /**
     * Create a user interface.
     * @param engine The calculator engine.
     */
    public UserInterface(CalcEngine engine)
    {
        calc = engine;
        showingAuthor = true;
        makeFrame();

        frame.setVisible(true);

    }

    /**
     * Set the visibility of the interface.
     * @param visible true if the interface is to be made visible,
false otherwise.
     */
    public void setVisible(boolean visible)
    {
        frame.setVisible(visible);
    }

    /**
     * Make the frame for the user interface.
     */
    protected void makeFrame()
    {
        frame = new JFrame(calc.getTitle());

        JPanel contentPane = (JPanel)frame.getContentPane();
        contentPane.setLayout(new BorderLayout(0, 8));
        contentPane.setBorder(new EmptyBorder( 10, 10, 10, 10));

        JPanel buttonPanel = new JPanel(new GridLayout(4, 5));
```

```
        addButton(buttonPanel, "7");
        addButton(buttonPanel, "8");
        addButton(buttonPanel, "9");
        addButton(buttonPanel, "Clear");
        addButton(buttonPanel, "?");

        addButton(buttonPanel, "4");
        addButton(buttonPanel, "5");
        addButton(buttonPanel, "6");
        addButton(buttonPanel, "*");
        addButton(buttonPanel, "/");

        addButton(buttonPanel, "1");
        addButton(buttonPanel, "2");
        addButton(buttonPanel, "3");
        addButton(buttonPanel, "+");
        addButton(buttonPanel, "-");

        addButton(buttonPanel, "0");
        buttonPanel.add(new JLabel(" "));
        buttonPanel.add(new JLabel(" "));
        addButton(buttonPanel, "^");
        addButton(buttonPanel, "=");

        contentPane.add(buttonPanel, BorderLayout.CENTER);



        JPanel checkboxPanel = new JPanel(new GridLayout(3,1));
        display = new JTextField();
        checkboxPanel.add(display);

        addCheckbox(checkboxPanel, "DEC", false);
        addCheckbox(checkboxPanel, "HEX", true);

        contentPane.add(checkboxPanel, BorderLayout.NORTH);

        status = new JLabel(calc.getAuthor());
        contentPane.add(status, BorderLayout.SOUTH);
        frame.pack();
    }
```

```
    /**
     * Add a button to the button panel.
     * @param panel The panel to receive the button.
     * @param buttonText The text for the button.
     */
    protected void addCheckbox(Container panel, String
checkboxText, boolean selected)
    {
      JCheckBox checkbox = new JCheckBox(checkboxText, selected);
      checkbox.addActionListener(this);
      buttonG.add(checkbox);
         panel.add(checkbox);
    }

    protected void addButton(Container panel, String buttonText)
    {
        JButton button = new JButton(buttonText);
        button.addActionListener(this);
        panel.add(button);
    }

    /**
     * An interface action has been performed.
     * Find out what it was and handle it.
     * @param event The event that has occured.
     */
    public void actionPerformed(ActionEvent event)
    {
        command = event.getActionCommand();
        char c;

        if(command.equals("DEC"))
        {hexaPanel.setVisible(false);}

        else if(command.equals("HEX"))
        {hexaPanel.setVisible(true);}

        else if(command.equals("0") ||
                command.equals("1") ||
                command.equals("2") ||
```

```
            command.equals("3") ||
            command.equals("4") ||
            command.equals("5") ||
            command.equals("6") ||
            command.equals("7") ||
            command.equals("8") ||
            command.equals("9")) {
       c = (char) (command.charAt(0));
        calc.numberPressed(c);
    }
    else if(command.equals("A") ||
            command.equals("B") ||
            command.equals("C") ||
            command.equals("D") ||
            command.equals("E") ||
            command.equals("F")) {
       c = (char) (command.charAt(0));
        calc.numberPressed(c);
    }
    else if(command.equals("^")) {
        calc.exponentiation();
    }
    else if(command.equals("*")) {
        calc.multiplication();
    }
    else if(command.equals("/")) {
       calc.division();
    }
    else if(command.equals("+")) {
        calc.plus();
    }
    else if(command.equals("-")) {
        calc.minus();
    }
    else if(command.equals("=")) {
        calc.equals();

    }
    else if(command.equals("Clear")) {
        calc.clear();
    }
```

```
        else if(command.equals("?")) {
            showInfo();
        }
        // else unknown command.

        redisplay();
        if(command.equals("=")) {
           calc.clear();
        }
    }

    /**
     * Update the interface display to show the current value of
the
     * calculator.
     */
    protected void redisplay()
    {
     if(hexaPanel.isVisible() == true && command.equals("="))
     {
           double calcD = Double.valueOf(calc.calculation);
           int calcInt = (int) calcD;
           calc.calculation =
(Integer.toHexString(calcInt)).toUpperCase();
      }

      display.setText("" + calc.calculation);

    }

    /**
     * Toggle the info display in the calculator's status area
between the
     * author and version information.
     */
    protected void showInfo()
    {
        if(showingAuthor)
            status.setText(calc.getVersion());
        else
            status.setText(calc.getAuthor());
```

```
            showingAuthor = !showingAuthor;
        }
    }
```