# Testaamiseen käytetyt metodi ja tulosteet testiajoista

## *FuncXtoY-luokan sisältämät testimetodit*

```
//--------------------------------------------------------------------------------
// Testaukseen tarkoitetut metodit:

/**
 * Testauksessa hyödynnettävä funktio, joka tarkastaa, ovatko liukuluvut käytännössä
 * samoja.
 * @param val1,val2    testattavat arvot
 */
private static boolean close(double val1, double val2) {
 return Math.abs(val2 / val1 - 1.0) < .001;
}

/**
 * Testausapufunktio, joka tulostaa annetun testin string-esityksen, sekä tiedon testin
 * onnistumisesta ruudulle. Tuloste pilkkoutuu alle 80 merkin mittaisiksi riveiksi.
 * @param s
 * @param succ
 */
private static void printResult(String s, boolean succ) {
 while( s.length() > 70 ) {
        System.out.print(s.substring(0, 70) + "..\n");
        s = ".." + s.substring(70);
 }
 char[] space = new char[75 - s.length()];
 java.util.Arrays.fill(space, ' ');
 s += new String(space);
 System.out.println( s + (succ ? "OK" : "FAIL!"));   }

/**
 * Testausfunktio tutkii, täsmääkö lausekkeen arvo annettuun vertailuarvoon, ja
 * tulostaa verrattavan lausekkeen, tulosarvon sekä täsmäsikö se vertailuarvoon.
 * Tulostaa tuloksen näytölle.
 * @param e            lauseke, jonka arvoa tutkitaan
 * @param val   vertailuarvo
 */
private static void testValue(Expression e, double val) {
 String s = e + " = " + e.getValue() + " (" + val + ")";
 printResult(s, close(e.getValue(), val));
}

/**
 * Lausekkeen sisävävän string-muuttujan paloittelun testaukseen käytettävä metodi.
 * Tulostaa alkuperäisen lausekkeen, sekä paloitellun version palat välilyönnillä
 * eroteltuina.
 * @param s            paloiteltava merkkijono
 */
private static void testSplit(String s) {
 List<String> atoms = Split(s);
 System.out.println(s + "\n => ");
 for( int i = 0; i < atoms.size(); ++i )
        System.out.print(" " + atoms.get(i));
 System.out.println();
}

/**
 * Testausfunktio, joka tulostaa parametrina annetun merkkijonon, ja siitä johdetun
 * kaavan käyttäjän vertailtavaksi.
 * @param s            vertailtava kaava
 */
private static void printConstructTest(String s) {
 System.out.println(s + "\n => " + Construct(Split(s)));
}

/**
 * FuncXtoY-luokan testaukseen tarkoitettu apumetodi. Luo ko. luokan ilmentymän
 * parametrina annetusta lausekkeesta. Laskee lausekkeen arvon annetussa pisteessä,
 * ja vertaa sitä valmiiksi annettuun vertailuarvoon. Tulostaa lausekkeen, lasketun
 * arvon sekä vertailun tuloksen näytölle.
 * @param expr  lauseke, josta funktion generoidaan
 * @param x            piste, jossa funktion arvo lasketaan
 * @param val   vertailuarvo
 */
private static void testFuncXtoY(String expr, double x, double val) {
 FuncXtoY f = new FuncXtoY(expr);
```

```
        String s = "y = " + f.rightSide +  " = " + f.calcValue(x) + " (" + val + ")";
        printResult(s, close(f.calcValue(x), val));
       }

       /**
        * Testaukseen tarkoitettu main-metodi.
        * @param args
        */
       public static void main(String[] args) {
        //Scanner cin = new java.util.Scanner(System.in);

        //String kaava = cin.nextLine();
        //double val1, val2;

        //Expression e;

        // operaatioiden testaus
        System.out.println("Operaatioiden testaus");
        testValue(new Add(CONST_PI, CONST_PI), Math.PI + Math.PI);
        testValue(new Sub(new Constant(1.0), new Constant(-1.0)), 2.0);
        testValue(new Mul(new Constant(2.0), new Constant(-1.5)), -3.0);
        testValue(new Div(new Constant(2.0), new Constant(-1.5)), -2.0/1.5);
        testValue(new Pow(new Constant(-2.0), new Constant(-1.0)), -.5);

        // funktioiden testaus
        System.out.println("\nFunktioiden testaus");
        testValue(new Sin(new Constant(-5)), Math.sin(-5));
        testValue(new Cos(new Constant(-5)), Math.cos(-5));
        testValue(new Tan(new Constant(-5)), Math.tan(-5));
        testValue(new Tan(new Constant(Math.PI/2)), Math.tan(Math.PI/2));
        testValue(new Asin(new Constant(0.1)), Math.asin(0.1));
        testValue(new Acos(new Constant(-1)), Math.acos(-1));
        testValue(new Atan(new Constant(-5)), Math.atan(-5));
        testValue(new Sinh(new Constant(-5)), Math.sinh(-5));
        testValue(new Cosh(new Constant(-5)), Math.cosh(-5));
        testValue(new Tanh(new Constant(-5)), Math.tanh(-5));
        testValue(new Ln(new Constant(100)), Math.log(100));
        testValue(new Lg(new Constant(30)), Math.log10(30));
        testValue(new Exp(new Constant(-5)), Math.exp(-5));
        testValue(new Sqrt(new Constant(30)), Math.sqrt(30));
        testValue(new Cbrt(new Constant(-5)), Math.cbrt(-5));
        testValue(new Abs(new Constant(-5)), Math.abs(-5));

        // Testataan kaavanpilkkojaa
        System.out.println("\nTestataan kaavanpilkkojaa");
        testSplit("1+.2-3E1*-7.406/.01E-4.2^0E++6");
        testSplit(")x+)+-  (n2sin xxx /*+- 2--)^ln -abs help2 cos (((x + sqrt x*-.3E-2)");

        // Testataan kaavantulkintaa kokonaisuudessaan
        System.out.println("\nTestataan kaavantulkintaa kokonaisuudessaan");
        printConstructTest("sin -2");
        printConstructTest("1 + exp 5 ^ 2 * 3");
        printConstructTest("1 + exp 5 ^ 2 ^ 3");
        printConstructTest("cos -2^3");
        printConstructTest("(cos -2)^3");
        printConstructTest("1^sin 2^sin 3^sin 4");
        printConstructTest("-2+3");
        printConstructTest("-2*3");
        printConstructTest("-2/3");
        printConstructTest("-2^3");
        printConstructTest("1+2*3^4");
        printConstructTest("4^3/2-1");
        printConstructTest("-2/3");
        printConstructTest("0-2/3");

        // Testataan koko homman toimintaa
        System.out.println("\nTestataan koko homman toimintaa");
        testFuncXtoY("x+1", 5, 6);
        testFuncXtoY("3*x^2", 4, 48);
        testFuncXtoY("2*x^4 +  x^2/2 + .5*x^-1 + 1", 2, 35.25);
        testFuncXtoY("sin cos x", -1, .514395258);
        testFuncXtoY("x^x^x", 3, 19683);
        testFuncXtoY("x^(x^x)", 3, 7.625597485e12);

//      varX.setValue(1);
//      System.out.println(e.getValue());
//      varX.setValue(2);
//      System.out.println(e.getValue());

//      kaava = "x";
//      FuncXtoY f = new FuncXtoY(kaava);
//      System.out.println(f.calcValue(2));
       }
```

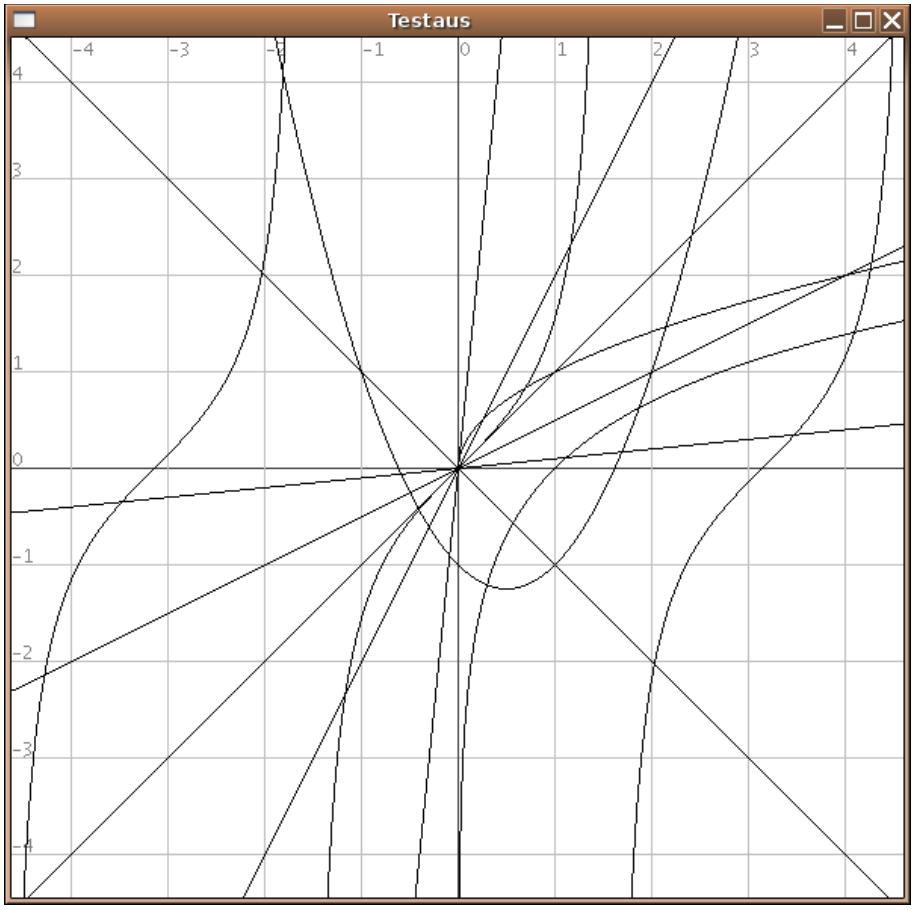
## Testin tulos

```
Operaatioiden testaus
(3.141592653589793 + 3.141592653589793) = 6.283185307179586 (6.2831853..
..07179586)                                                              OK
(1.0 - -1.0) = 2.0 (2.0)                                                 OK
(2.0 * -1.5) = -3.0 (-3.0)                                               OK
(2.0 / -1.5) = -1.3333333333333333 (-1.3333333333333333)                 OK
(-2.0 ^ -1.0) = -0.5 (-0.5)                                              OK

Funktioiden testaus
(sin -5.0) = 0.9589242746631385 (0.9589242746631385)                     OK
(cos -5.0) = 0.28366218546322625 (0.28366218546322625)                   OK
(tan -5.0) = 3.380515006246586 (3.380515006246586)                       OK
(tan 1.5707963267948966) = 1.633123935319537E16 (1.633123935319537E16)   OK
(arcsin 0.1) = 0.1001674211615598 (0.1001674211615598)                   OK
(arccos -1.0) = 3.141592653589793 (3.141592653589793)                    OK
(arctan -5.0) = -1.373400766945016 (-1.373400766945016)                  OK
(sinh -5.0) = -74.20321057778875 (-74.20321057778875)                    OK
(cosh -5.0) = 74.20994852478785 (74.20994852478785)                      OK
(tanh -5.0) = -0.9999092042625951 (-0.9999092042625951)                  OK
(ln 100.0) = 4.605170185988092 (4.605170185988092)                       OK
(lg 30.0) = 1.4771212547196624 (1.4771212547196624)                      OK
(exp -5.0) = 0.006737946999085467 (0.006737946999085467)                 OK
(sqrt 30.0) = 5.477225575051661 (5.477225575051661)                      OK
(cbrt -5.0) = -1.709975946676697 (-1.709975946676697)                    OK
|-5.0| = 5.0 (5.0)                                                       OK

Testataan kaavanpilkkojaa
1+.2-3E1*-7.406/.01E-4.2^0E++6
 =>
 1 + .2 - 3E1 * - 7.406 / .01E-4.2 ^ 0E+ + 6
)x+)+-  (n2sin xxx /*+- 2--)^ln -abs help2 cos (((x + sqrt x*-.3E-2)
 =>
 ) x + ) + - ( n2sin xxx / * + - 2 - - ) ^ ln - abs help2 cos ( ( ( x + sqrt x * - .3E-2 )

Testataan kaavantulkintaa kokonaisuudessaan
sin -2
 => (sin (-2.0))
1 + exp 5 ^ 2 * 3
 => (1.0 + ((exp (5.0 ^ 2.0)) * 3.0))
1 + exp 5 ^ 2 ^ 3
 => (1.0 + (exp ((5.0 ^ 2.0) ^ 3.0)))
cos -2^3
 => (cos (-(2.0 ^ 3.0)))
(cos -2)^3
 => ((cos (-2.0)) ^ 3.0)
1^sin 2^sin 3^sin 4
 => (1.0 ^ (sin (2.0 ^ (sin (3.0 ^ (sin 4.0))))))
-2+3
 => ((-2.0) + 3.0)
-2*3
 => ((-2.0) * 3.0)
-2/3
 => ((-2.0) / 3.0)
-2^3
 => (-(2.0 ^ 3.0))
1+2*3^4
 => (1.0 + (2.0 * (3.0 ^ 4.0)))
4^3/2-1
 => (((4.0 ^ 3.0) / 2.0) - 1.0)
-2/3
 => ((-2.0) / 3.0)
0-2/3
 => (0.0 - (2.0 / 3.0))

Testataan koko homman toimintaa
y = (0.0 + 1.0) = 6.0 (6.0)                                              OK
y = (3.0 * (5.0 ^ 2.0)) = 48.0 (48.0)                                    OK
y = ((((2.0 * (4.0 ^ 4.0)) + ((4.0 ^ 2.0) / 2.0)) + (0.5 * (4.0 ^ (-1...
..0)))) + 1.0) = 35.25 (35.25)                                           OK
y = (sin (cos 2.0)) = 0.5143952585235492 (0.514395258)                   OK
y = ((-1.0 ^ -1.0) ^ -1.0) = 19683.0 (19683.0)                           OK
y = (3.0 ^ (3.0 ^ 3.0)) = 7.625597484987E12 (7.625597485E12)             OK
```

## *SurfaceXY-luokan sisältämät testimetodit*

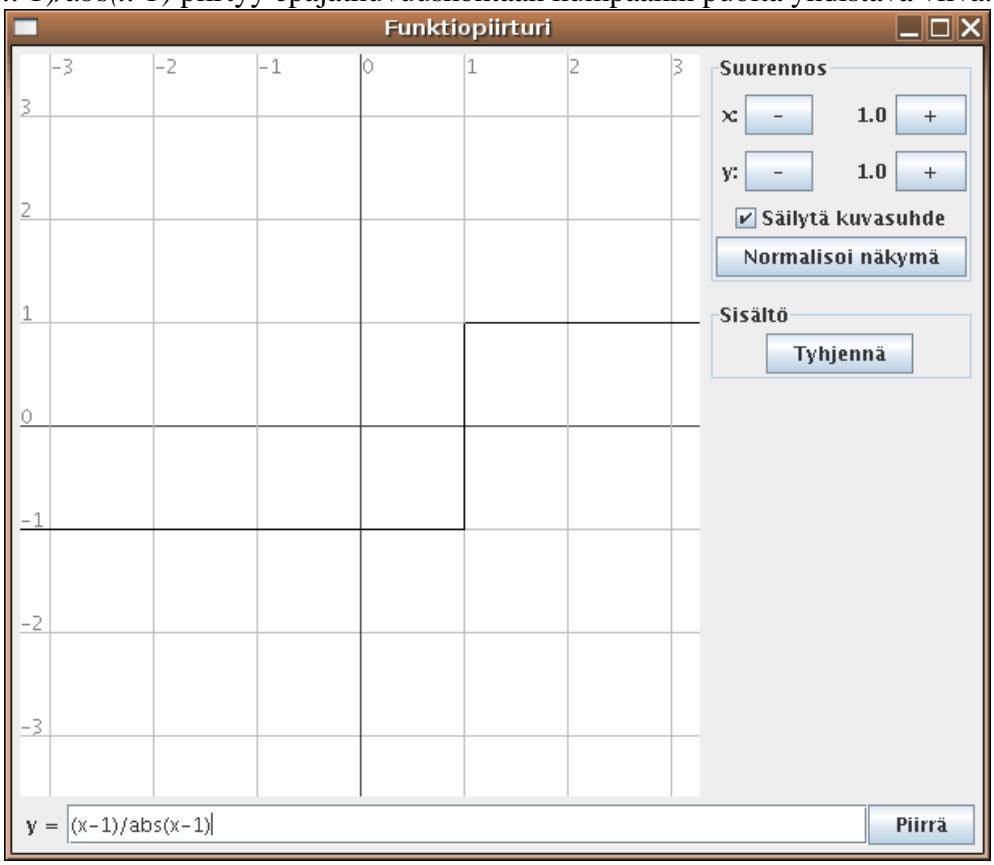
```
/**
 * Testausmetodi. Piirtelee muutamia kuvaajia ruudulle.
 * @param args  ohjelmalle välitettävät parametrit (ei vaikutusta)
 */
public static void main(String[] args) {
 JFrame frame = new JFrame("Testaus");
 SurfaceXY s = new SurfaceXY();
 frame.getContentPane().add(s);
 frame.setPreferredSize(new Dimension(600, 600));
 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 frame.pack();
 frame.setVisible(true);
 s.addFunction(new FuncXtoY("x"));
 s.addFunction(new FuncXtoY("2*x"));
 s.addFunction(new FuncXtoY("10*x"));
 s.addFunction(new FuncXtoY(".5*x"));
 s.addFunction(new FuncXtoY(".1*x"));
 s.addFunction(new FuncXtoY("-x"));
 s.addFunction(new FuncXtoY("x^2-x-1"));
 s.addFunction(new FuncXtoY("tan x"));
 s.addFunction(new FuncXtoY("sqrt x"));
 s.addFunction(new FuncXtoY("ln x"));
 s.addFunction(new FuncXtoY("tan x"));
}
```

## Testin tulos



### *Kokonaisuuden testauksessa havaittuja virheitä*

Funktiolla *(x-1)/abs(x-1)* piirtyy epäjatkuvuuskohtaan kumpaakin puolta yhdistävä viiva.




Myös tangenttifunktiolla on mahdollista saada epäjatkuvuuskohdat piirtymään väärin.