

```

1 // fo1/2/linear/Temperatures.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 /**
7  * Temperature conversion program,
8  * for exercising LinearEquation objects.
9  *
10 * @version 2
11 */
12
13 public class Temperatures
14 {
15     /**
16      * First a hardcoded test of Celsius-Fahrenheit conversion,
17      * then a loop allowing the user to test interactively.
18      */
19
20     public static void main( String[] args )
21     {
22         Terminal terminal = new Terminal();
23
24         // create a Celsius to Fahrenheit converter
25         LinearEquation c2f = new LinearEquation( 9.0/5.0, 32.0 );
26
27         // ask it to tell us its inverse, for F to C
28         LinearEquation f2c = c2f.getInverse();
29
30         ////////////////////////////////////////////////////////////////////
31         // Testing style 1: Hard coded, self-documenting //
32         ////////////////////////////////////////////////////////////////////
33
34         terminal.println( "Hard coded self documenting tests:" );
35         terminal.print( "c2f.compute( 0.0 ), should see 32.0: " );
36         terminal.println( c2f.compute( 0.0 ) );
37         terminal.print( "f2c.compute( 212.0 ), should see 100.0: " );
38         terminal.println( f2c.compute( 212.0 ) );
39
40         ////////////////////////////////////////////////////////////////////
41         // Testing style 2: Interactive //
42         ////////////////////////////////////////////////////////////////////
43
44         terminal.println();
45         terminal.println( "Interactive tests:" );
46         while ( terminal.readYesOrNo( "more?" ) ) {
47             double degreesCelsius =
48                 terminal.readDouble( "Celsius: " );
49             terminal.println( " = "
50                 + c2f.compute( degreesCelsius )
51                 + " degrees Fahrenheit" );
52             double degreesFahrenheit =
53                 terminal.readDouble( "degrees Fahrenheit: " );
54             terminal.println( " = "
55                 + f2c.compute( degreesFahrenheit )
56                 + " degrees Celsius" );

```

```

57     }
58 }
59 }

```

```

1 // fo1/2/LinearEquation.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5 /**
6  * A LinearEquation models equations of the form  $y = mx + b$ .
7  *
8  * @version 2
9  */
10
11 public class LinearEquation
12 {
13     private double m; // The equations's slope
14     private double b; // The equations's y-intercept
15
16     /**
17      * Construct a LinearEquation from a slope and y-intercept.
18      *
19      * @param m the slope.
20      * @param b the y-intercept.
21      */
22
23     public LinearEquation( double m, double b )
24     {
25         this.m = m;
26         this.b = b;
27     }
28
29     /**
30      * Construct a LinearEquation from two points.
31      *
32      * @param x1 the x coordinate of the first point
33      * @param y1 the y coordinate of the first point
34      * @param x2 the x coordinate of the second point
35      * @param y2 the y coordinate of the second point
36      */
37
38     public LinearEquation( double x1, double y1,
39                           double x2, double y2 )
40     {
41         m = (y2 - y1) / (x2 - x1);
42         b = y1 - x1 * m;
43     }
44
45     /**
46      * Compute Y, given x.
47      *
48      * @param x the input value.
49      * @return the corresponding value of y: mx+b.
50      */
51
52     public double compute( double x )
53     {
54         return m*x + b;
55     }
56

```

```

57
58     /**
59      * Compute the inverse of this linear equation.
60      *
61      * @return the LinearEquation object you get by "solving for x".
62      */
63
64     public LinearEquation getInverse()
65     {
66         return new LinearEquation( 1.0/m, -b/m );
67     }
68 }

```