```java
// joi/2/linear/Temperatures.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * Temperature conversion program,
 * for exercising LinearEquation objects.
 *
 * @version 2
 */

public class Temperatures
{
    /**
     * First a hardcoded test of Celsius-Fahrenheit conversion,
     * then a loop allowing the user to test interactively.
     */

    public static void main( String[] args )
    {
        Terminal terminal = new Terminal();

        // create a Celsius to Fahrenheit converter
        LinearEquation c2f = new LinearEquation( 9.0/5.0, 32.0 );

        // ask it to tell us its inverse, for F to C
        LinearEquation f2c = c2f.getInverse();

        ///////////////////////////////////////////////////
        // Testing style 1: Hard coded, self-documenting //
        ///////////////////////////////////////////////////

        terminal.println( "Hard coded self documenting tests:" );
        terminal.print( "c2f.compute( 0.0 ), should see 32.0: ");
        terminal.println( c2f.compute( 0.0 ) );
        terminal.print( "f2c.compute( 212.0 ), should see 100.0: ");
        terminal.println( f2c.compute( 212.0 ) );

        //////////////////////////////////
        // Testing style 2: Interactive //
        //////////////////////////////////

        terminal.println();
        terminal.println( "Interactive tests:" );
        while ( terminal.readYesOrNo("more?") ) {
            double degreesCelsius  =
                terminal.readDouble( "Celsius: " );
            terminal.println(" = "
                             + c2f.compute( degreesCelsius )
                             + " degrees Fahrenheit" );
            double degreesFahrenheit  =
                terminal.readDouble( "degrees Fahrenheit: " );
            terminal.println(" = "
                             + f2c.compute( degreesFahrenheit )
                             + " degrees Celsius" );
```

```java
        }
    }
}
```

```java
// joi/2/linear/LinearEquation.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A LinearEquation models equations of the form y = mx + b.
 *
 * @version 2
 */

public class LinearEquation
{
    private double m;      // The equations’s slope
    private double b;      // The equations’s y-intercept

    /**
     * Construct a LinearEquation from a slope and y-intercept.
     *
     * @param m the slope.
     * @param b the y-intercept.
     */

    public LinearEquation( double m, double b )
    {
        this.m = m;
        this.b = b;
    }

    /**
     * Construct a LinearEquation from two points.
     *
     * @param x1 the x coordinate of the first point
     * @param y1 the y coordinate of the first point
     * @param x2 the x coordinate of the second point
     * @param y2 the y coordinate of the second point
     */

    public LinearEquation( double x1, double y1,
                           double x2, double y2 )
    {
        m = (y2 - y1) / (x2 - x1);
        b = y1 - x1 * m;
    }

    /**
     * Compute y, given x.
     *
     * @param x the input value.
     * @return the corresponding value of y: mx+b.
     */

    public double compute( double x )
    {
        return m*x + b;
    }
```

```java

    /**
     * Compute the inverse of this linear equation.
     *
     * @return the LinearEquation object you get by "solving for x".
     */

    public LinearEquation getInverse()
    {
        return new LinearEquation( 1.0/m, -b/m );
    }
}
```