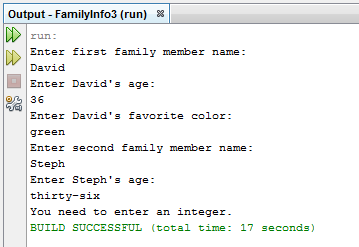
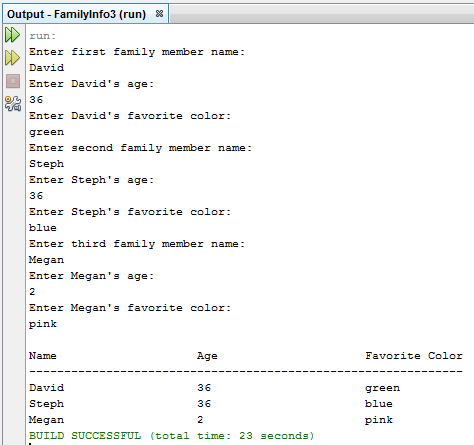
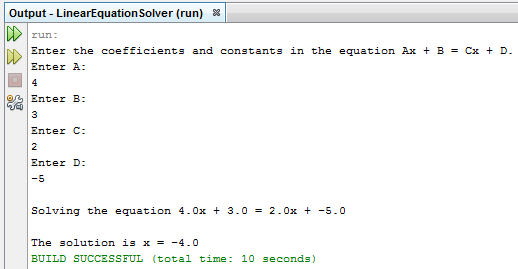
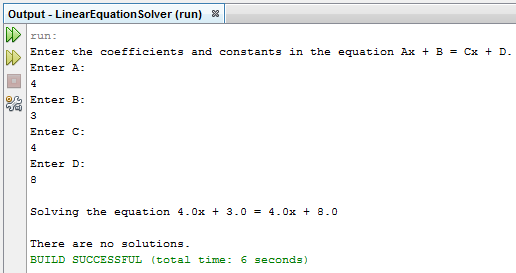
1. In the TextBasedIO program, you wrote a function which read information about three people and printed the results in a table. Your program read in ints (ie it used keyboard.nextInt()) without checking to see if the user had typed an int. If the user doesn’t comply with the desired input format, then your program will crash. Recreate this function in guardedInfoTable (you can just copy-paste your code from TextBasedIO), but use input-guards to make sure that your program doesn’t crash if the user doesn’t follow the input format. Your function just needs to check each time it reads an int, there is an int available. If not, the program can print a message about that and then terminate (ie return). If there is, then it should continue as before. For example, here’s a couple of runs of guardedInfoTable:

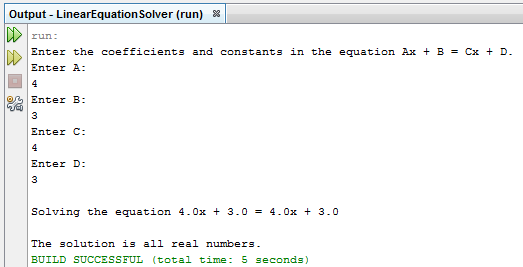




1. Write the function, linearEquationSolver, that solves linear equations of the form Ax + B = Cx + D. Usually, the solution to this equation is x = (D – B) / (A-C). For example, the solution to 4x + 3 = 2x – 5 is x = (-5 - 3)/(4-2) = -2. Sometimes, however, there is no solution. For example, there is no solution to 4x + 3 = 4x + 8. Also, sometimes, the solution is any real value. For example, all real numbers solve the equation 4x + 3 = 4x + 3. Your program must prompt the user to supply the coefficients and constants (A, B, C, and D) in the equation Ax + B = Cx + D. These may be any real number (not necessarily integers). The program should then print out the equation, and the solution. If there are no solutions, it should say so, and if the solution is all real numbers, then it should say that instead. (Hint: what is different about the special case equations? What condition do you need to guard against when using the formula to find the solutions?) Your program should also guard against incorrect input formats.







Small Bonus: Notice that if B and/or D is negative, then, when we print out the equation we get something like 4.0x + -5.0 = 2.0x + -3.0 rather than the more aesthetically pleasing 4.0x – 5.0 = 2.0x – 3.0. Change your program so that when it prints out the equation in these situations, it prints a minus instead of a + -. Also, if any of the constants is 0, you should not print out that term. For example: 4.0x = 2.0x + 3.0 instead of 4.0x + 0.0 = 2.0x + 3.0. You might want to use the function System.out.print which, in contrast to System.out.println, does not end the output line with a newline.

1. Write function quadraticEquationSolver. This function should be very similar to the linearEquationSolver in problem 2. Instead of linear equations of the form Ax + B = Cx + D, it should solve quadratic equations of the form Ax^2 + Bx + C = 0. Recall from Algebra II that the solutions to this equation are x = (-B + sqrt(B^2 – 4AC) ) / (2A) and x = (-B - sqrt(B^2 – 4AC) ) / (2A). Also recall that in some situations, there are 2 real solutions, in some situations there is only one real solution, and in some situations there are 2 complex solutions. Your program should read the coefficients from the user (guarding for incorrect input), print out the actual equation, and, after determining which situation we’re in, print out the solutions.

Note: The static function Math.sqrt takes a double as input and returns the square-root of the number. For example:

double x = 13.0;

double y = Math.sqrt( x ); // y will be ~= 3.6055

