Documentation:

- Notebook callout: Document all project work in your notebook. Use the development process as a guide.
- Investigate an Idea
    - Capture any ideas you generated as you explored the number riddle. This might include working through a few different numbers to verify the answer is always three.
        - If you substitute a variable x to have the operation be done on, the result will also be 3. This works because of algebra. The use of algebra can be used to create other number riddles evaluating to other numbers or the initial number.
- Plan
    - See problem 3
- Design
    - Skip for now
- Create and Test
    - See problems 4 and 5
- Evaluate the Solution
    - See problem 6
- Document and Present
    - See problem 7

Problems:

1. Begin by reviewing the requirements this program must meet.
    a. N/A
2. Review your notes from previous activities.
    a. N/A
3. Decompose your problem. The first step in problem decomposition is to identify the major parts of your program and decide what each part will do. You then try to estimate how long you think it will take to develop each part. These will become your milestones—tasks with estimated completion dates.
   Notebook callout: Recreate this milestone chart, predicting your own Estimated Time to Develop, adjusting the times as necessary. Know that many programmers find it difficult to create time estimates, but as with most things, the more you practice, the easier it becomes.
    a.

Milestones	Estimated development time (min)	Actual development time (min)
A class definition header	<1	<1

A `main` method	<1	<1
Variables for testing all types of numbers	<1	1
Algorithms to process the numbers	3	8
Print statements to show the number entered, the calculations, and the final result	4	4

4. Use the embedded code editor below to create your project. Work through each part of your program, referring back to your milestone chart. Test incrementally and pay attention to error messages to correct any mistakes.

   You should write your solution in two phases.
   - First, get the program to print out all the correct expected values for a single int you choose.
   - Second, add variables as necessary for the test cases in Step 5, then modify the code until you see all test cases returning the correct expected values.

   Notebook callout: As you complete each milestone, fill in the Actual Time to Develop column in your milestone chart. Compare it to your Estimated Time to Develop and note any discrepancies and why they occurred.

   a. trinket | Java beta | Run | Share

   NumbersRiddle.java

```
/*
 * Activity 1.1.6
 */
public class NumbersRiddle
{
  public static void main(String[] args) {
    /* Choose any integer, double it, add six,
    divide it in half, and subtract the number
    you started with. The answer is always three! */

    // Test cases
    int positiveI = 2;
    int negativeI = -1;
    int zero = 0;
    int one = 1;
    double positiveD = 1.5;
    double negativeD = -0.5;

    // Run calculations for each value
    calculate(positiveI);
    calculate(negativeI);
    calculate(zero);
    calculate(one);
    calculate(positiveD);
    calculate(negativeD);
  }

  public static void calculate(double startingNumberChosen) {
    // print initial number
    System.out.println("Starting number: " + startingNumberChosen);
    // double number
    double doubleNumber = startingNumberChosen * 2;
    System.out.println(startingNumberChosen + " * 2 = " + doubleNumber);
    // add 6 to previous result
    double addSix = doubleNumber + 6;
    System.out.println(doubleNumber + " + 6 = " + addSix);
    // divide previous result by 2
    double divideHalf = addSix / 2;
    System.out.println(addSix + " / 2 = " + divideHalf);
    // subtract initial number from previous result
    double subtractStarting = divideHalf - startingNumberChosen;
    System.out.println(divideHalf + " - " + startingNumberChosen + " = " + subtractStarting + "\n");
  }
}
```

```
Powered by trinket
Starting number: 2.0
2.0 * 2 = 4.0
4.0 + 6 = 10.0
10.0 / 2 = 5.0
5.0 - 2.0 = 3.0

Starting number: -1.0
-1.0 * 2 = -2.0
-2.0 + 6 = 4.0
4.0 / 2 = 2.0
2.0 - -1.0 = 3.0

Starting number: 0.0
0.0 * 2 = 0.0
0.0 + 6 = 6.0
6.0 / 2 = 3.0
3.0 - 0.0 = 3.0

Starting number: 1.0
1.0 * 2 = 2.0
2.0 + 6 = 8.0
8.0 / 2 = 4.0
4.0 - 1.0 = 3.0

Starting number: 1.5
1.5 * 2 = 3.0
3.0 + 6 = 9.0
9.0 / 2 = 4.5
4.5 - 1.5 = 3.0

Starting number: -0.5
-0.5 * 2 = -1.0
-1.0 + 6 = 5.0
5.0 / 2 = 2.5
2.5 - -0.5 = 3.0
```

   Some discrepancies include how the variables for testing and algorithms both took longer than expected. The variables for test cases took longer than expected since I needed to figure out what to name the variables and their datatypes, along with the numbers I test. Because some of the decisions involved

in making the variable definitions for test cases were needed and were not expected, the time needed for making the test cases was longer than I expected. Also, developing the algorithms processing the numbers took much longer than expected. This is also because I needed to make decisions when writing the algorithm, and I ran into errors when writing my code. Since it took extra time to troubleshoot, my algorithm took much longer to complete since I needed to find the cause of the error and resolve it.

5. Test cases: Create variables and choose numbers that match the following six number types to verify your program works for each.
    a. See 4
6. Evaluate your solution. Does your program function as expected? Does your program meet all the requirements that are in the scoring guidelines? How did your time estimates relate to your actual development time?
    a. My program worked and followed all requirements, and my development time estimates were shorter than the actual development times.
7. Document and collect the following items.
    a. The purpose of your program. Imagine explaining to someone not in your class what the program should do.
        i. The purpose of my program is to demonstrate that for an int or double, if you double, add 6 to, divide in half, and subtract the initial number from it, the answer that is gotten is always 3. If x is a number, the program demonstrates the following equation would work for all integers and decimals:

    (2x+6)/2-x=3
    b. Quality screenshots of your program's code and output using a double. (If you figured this out, then you figured out how to process the double to an int output). You may use more than one screenshot.
        i. See 4
    c. An explanation of the different parts of your program.
        i. The program has a main class where all parts of the code work, and a main method that gets executed when the program is ran. In the main method, 6 variables with different characteristics have been defined containing the numbers to test if the series of operations listed above can produce a result of 3 when the operations are done on each of the numbers listed. Then, there are method calls listed that do the set of operations on each of the test cases, or numbers to test. The calculate method that does the operation to the test cases is listed below the main method, and in its definition, the numerical value used in a function is widened into a double if needed. After the definition, the initial number printed to test the series of operations on is defined. The subsequent code defines variables storing the result of each operation after it is done, and the operation is also printed. The last operation printed prints 3.0, meaning that the test case makes the operations evaluate to 3.
8. Submit your work as directed by your teacher.

a. N/A