Question 2 *(30 points)* Given the application, Exam2Nonsense consisting of Exam2Nonsense.java and Exam2NonsenseMethods.java (which is found on the Reference pages), carry out the activities in part a and b. Be sure to check the entire program before trying to trace and be careful since there are prints in some of the methods. There are multiple methods in this application.

Part a was to trace the code and write the output.

Part b (10 points) – Write all of the line numbers and specific features requested in the question. You must provide **ONE** line numbers and only **ONE** example. Be sure to include the character in front of the line number. NOTE: I have altered the original exam question to add in a few additional items.

Feature	**ONE** line numbers	**One** example
Primitive variable declaration		
Object instantiation		
Return type		
Method invocation (call)		
Parameter (formal parameter)		
Boolean expression		
Argument (actual parameter)		
An update statement for a loop		
A primitive literal		
A reference type literal		
A constant declaration		
A reference variable declaration		
A reference variable instantiation		
A return statement		
A return type		

Reference

```
A-1.  public class Exam2Nonsense
A-2.  {
A-3.          public static void main(String[] args)
A-4.          {
A-5.                  Exam2NonsenseMethods test;
A-6.                  double target;
A-7.                  double start;
A-8.
A-9.                  test = new Exam2NonsenseMethods();
A-10.
A-11.                 target = 2.5;
A-12.                 start = -2;
A-13.
A-14.                 System.out.printf("Target: %.2f Start: %.2f\n", target, start);
A-15.                 start = test.nonsense(target, start);
A-16.
A-17.                 System.out.printf("Target: %.2f NewValue: %.2f\n", target, start);
A-18.         }
A-19. }
```

```
B-1.  public class Exam2NonsenseMethods
B-2.  {
B-3.          private final double PRECISION = 2;
B-4.
B-5.          public boolean isClose(double expected, double actual)
B-6.          {
B-7.                  boolean closeEnough;
B-8.
B-9.                  if (expected - actual == 0)
B-10.                 {
B-11.                         closeEnough = true;
B-12.                 }
B-13.                 else if(Math.abs(expected - actual) <= PRECISION)
B-14.                 {
B-15.                         closeEnough = true;
B-16.                 }
B-17.                 else
B-18.                 {
B-19.                         closeEnough = false;
B-20.                 }
B-21.                 return closeEnough;
B-22.         }
B-23.
B-24.         public double hilo(double expected, double actual)
B-25.         {
B-26.                 double newActual;
B-27.
B-28.                 if (Math.abs(expected - actual) > PRECISION && expected > actual)
B-29.                         newActual = actual + PRECISION;
B-30.                 else if (Math.abs(expected - actual) > PRECISION && expected < actual)
B-31.                         newActual = actual - PRECISION;
B-32.                 else
B-33.                         newActual = actual;
B-34.                 System.out.printf("newActual: %.2f\n", newActual);
B-35.                 return newActual;
B-36.         }
B-37.
B-38.         public double nonsense(double expected, double actual)
B-39.         {
B-40.                 int iterations;
B-41.                 iterations = 0;
B-42.
B-43.                 while (!isClose(expected, actual))
B-44.                 {
B-45.                         iterations++;
B-46.                         actual = hilo(expected, actual);
B-47.                 }
B-48.                 System.out.printf("iterations: %d\n", iterations);
B-49.
B-50.                 return actual;
B-51.         }
B-52. }
```