1 Which of the following are not valid Java identifiers? Why?

1. `Factorial`
2. `anExtremelyLongIdentifierIfYouAskMe`
3. `2ndLevel`
4. `level2`
5. `MAX_SIZE`
6. `highest$`
7. `hook&ladder`

Item 3 is not a valid Java identifier because it started with a number.
*Item 5 is not a valid Java identifier because it contains an &. The only special characters allowed in a Java identifier are underscore _ and dollar **$** .*

2 Why are the following valid Java identifiers not considered good identifiers?

1. `q`
2. `totVal`
3. `theNextValueInTheList`

Identifier is to provide a hint to what they are there for. Having a single character, q, as in item 1 is vague and problematic for maintenance.

Item 2 is slightly better, but spelling the whole thing out is more preferred. total_Value is much better.

Item 3 is the opposite of item 1, which is too long. While Java's identifiers don't have to be meaningful. As long as they conform to the Java compiler's rules they will compile fine. However, it is for those who must maintain the software to benefit a great deal working with clearly defined identifiers.

3 Categorize each of the following situations as a compile-time error, run-time error, or logical error:

1. multiplying two numbers when you meant to add them
2. dividing by zero
3. forgetting a semicolon at the end of a programming statement
4. spelling a word wrong in the output
5. producing inaccurate results
6. typing a { when you should have typed (

	Items					
	1	2	3	4	5	6
Compile-time error			x			x
Run-time error		x				
Logical error	x			x	x	

Table 1: Answers to question 3 above.

4 What output is produced by the following code fragment? Explain.

```
System.out.print ("Here we go!");
System.out.println ("12345");
System.out.print ("Test this if you are not sure.");
System.out.print ("Another.");
System.out.println ();
System.out.println ("All done.");
```

Output:

```
Here we go!12345            (followed by a new line because of )
Test this if you are not sure.Another.  (followed by nothing and a
                                         new line)
All done.                   (followed by a new line because of
                            System.out.println())
```

5 What is wrong with the following program statement? How can it be fixed?

```
System.out.println ("To be or not to be, that is the

question.");
```

Output:

The program statement will not compile because of the missing double quotes, one at the end of `To be or not to be, that is the` *, and one at the beginning of* `question.` *, in addition to a + character. The fix is as follows:*

```
System.out.println ("To be or not to be, that is the" +
"question.");
```

6 What output is produced by the following statement? Explain.

```
   System.out.println ("He thrusts his fists\n\tagainst" +
" the post\nand still insists\n\the sees the \"ghost\"");
```

Output:

The statement above produces the following output:

```
He thrusts his fists
      against the post
and still insists
      he sees the "ghost"
```

Reasons:
System.out.println() method joins the two strings together via the + operation feature. However, within each string there are escape sequences like ***\n*** *(new line),* ***\t*** *(horizontal tab), and two \" (quote or unquote) patterns that also got printed out to create the desired output effects as shown above.*

7 For each of the following expressions, indicate the order in which the operators will be evaluated by writing a number beneath each operator.

1. `a + b / c * d`
   (3) (1) (2)
2. `a % b / c * d`
   (1) (2) (3)
3. `(a – (b – c)) – d`
   (2) (1) (3)

8 Enter, compile, and run the following application:

```
public class Test
{
      public static void main (String[] args)
      {
          System.out.println ("An Emergency Broadcast");
      }
```

```
}
```

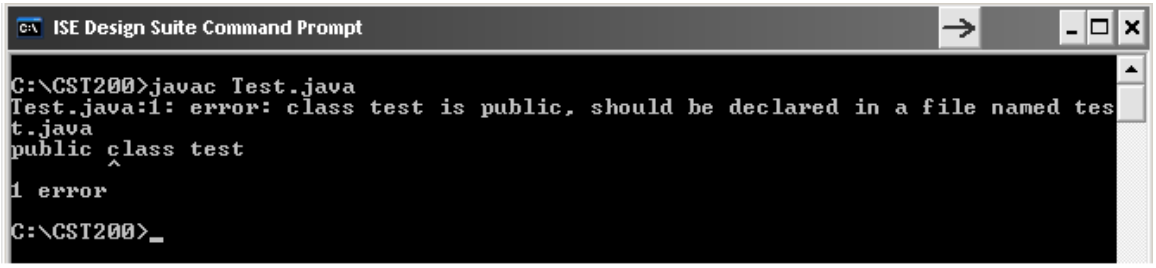
Output:

```
An Emergency Broadcast      (new line created)
```

9 (submit the output for each error) introduce the following errors, one at a time, to the program from the programming project 1.1. Record any error messages that the compiler produces. Fix the previous error each time before you introduce a new one. If no error messages are produced, explain why.

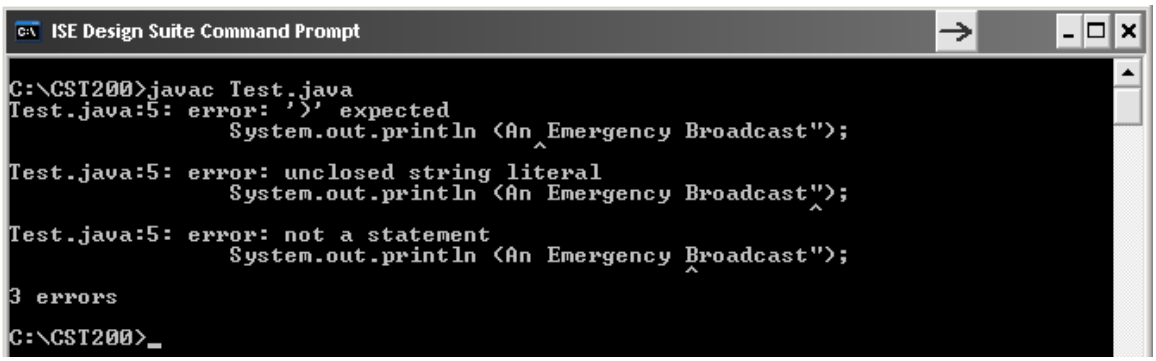
Try to predict what will happen before you make each change.

a. change Test to test (jEdit and DOS command window)

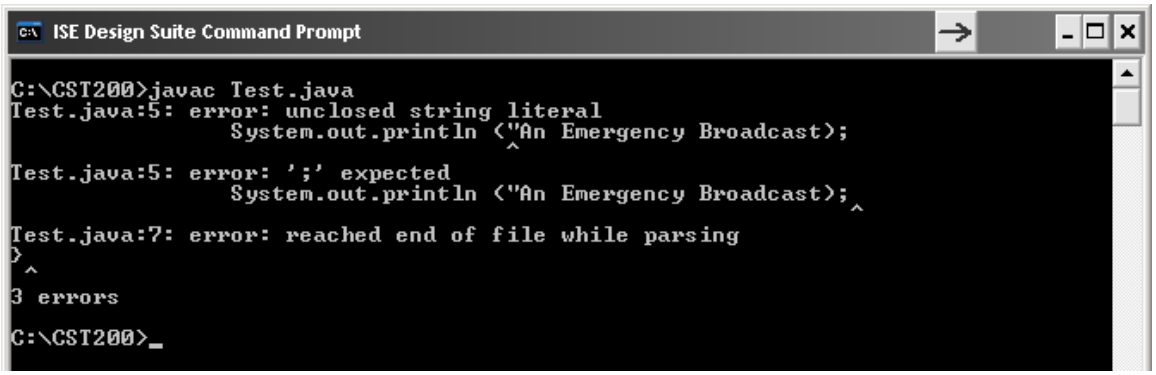


b. change Emergency to emergency (jEdit and DOS command window)

c. remove the first quotation mark in the string (jEdit and DOS command window)

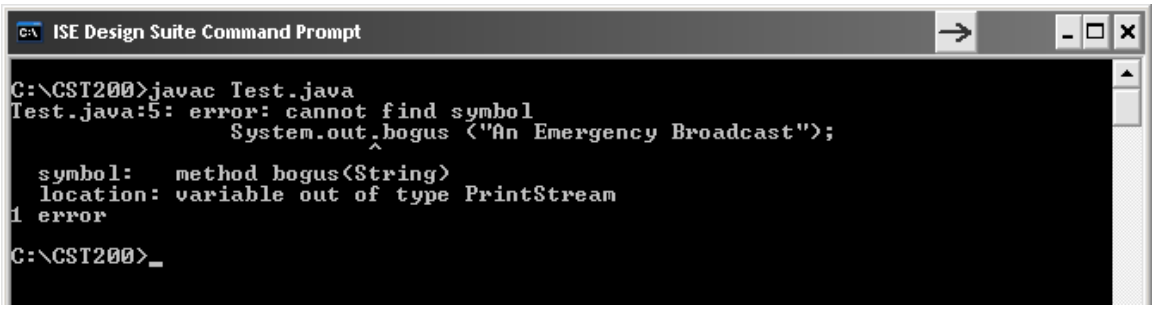


d. remove the last quotation mark in the string (jEdit and DOS command window)

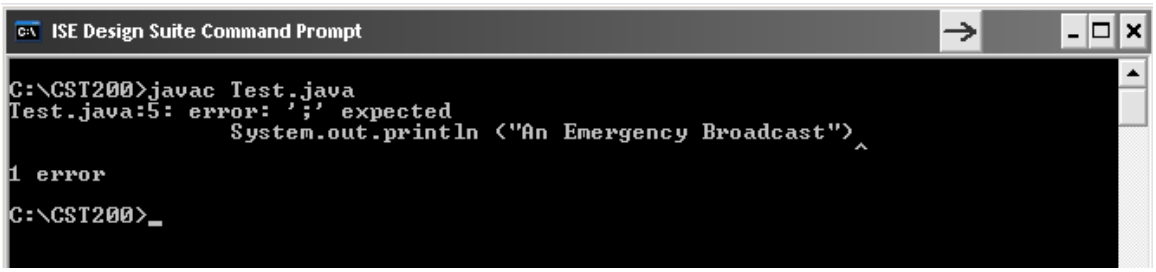


e. change main to man (jEdit and DOS command window)

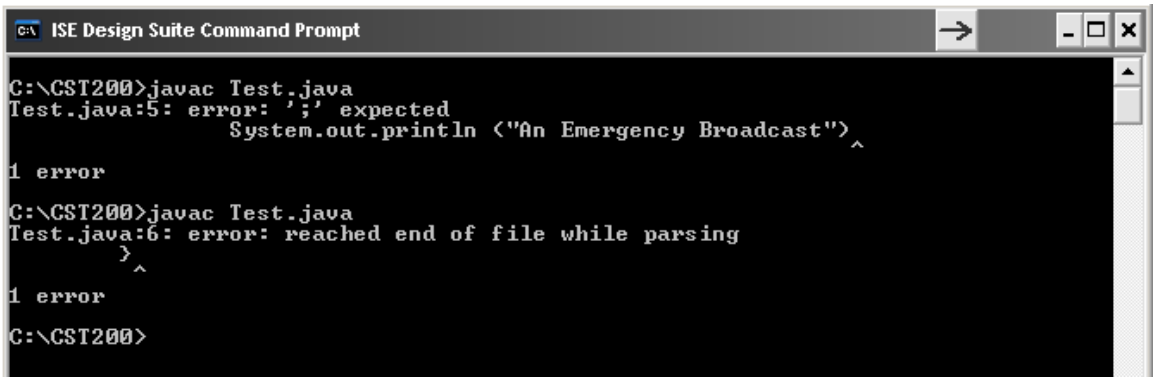
f. change println to bogus (jEdit and DOS command window)



g. remove the semicolon at the end of the println statement (jEdit and DOS command window)



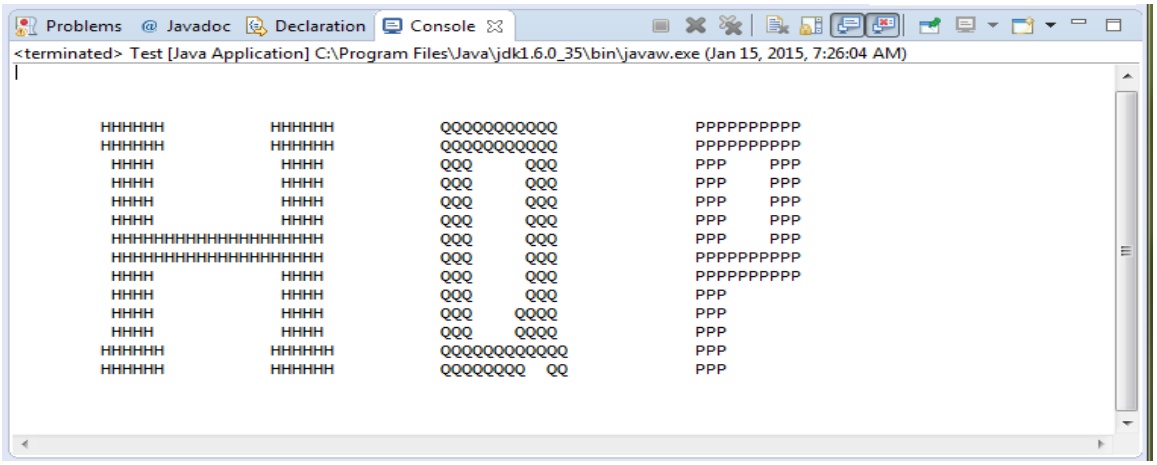
h. remove the last brace in the program (jEdit and DOS command window)



10 (submit the code) Write an application that displays your initials in large block letters.

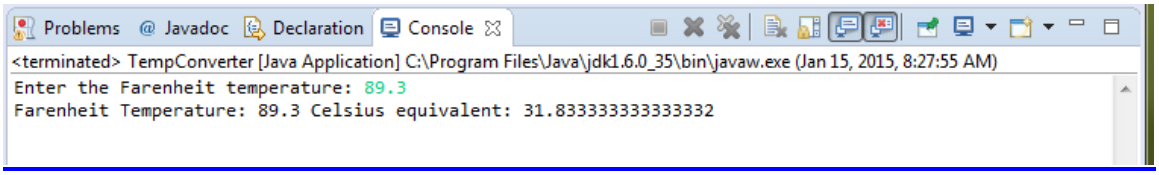
Make each large letter out of the corresponding regular character.

Output (Eclipse IDE):



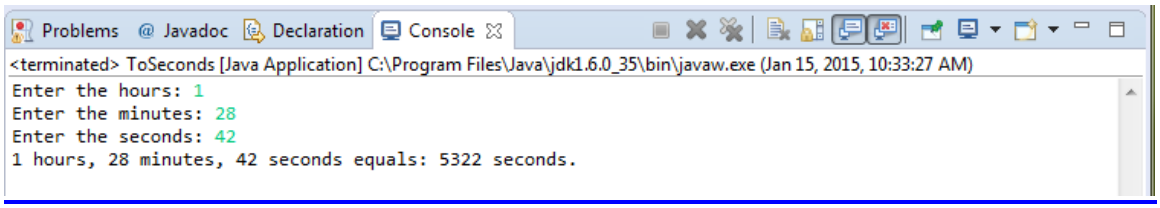
11 Create a version of the `TempConverter` application to convert from Fahrenheit to Celsius. Read the Fahrenheit temperature from the user.

Output (Eclipse IDE):



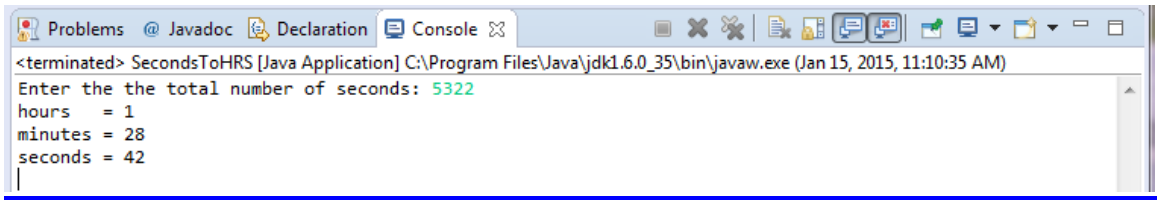
12 Write an application that reads values representing a time duration in hours, minutes, and seconds, and then prints the equivalent total number of seconds. (For example, 1 hour, 28 minutes, and 42 seconds is equivalent to 5322 seconds.)

Output (Eclipse IDE):



13 Create a version of the previous project that reverses the computation. That is, read a value representing a number of seconds, then print the equivalent amount of time as a combination of hours, minutes, and seconds. (For example, 9999 seconds is equivalent to 2 hours, 46 minutes, and 39 seconds.)

Output (Eclipse IDE):



14 Source Code Files

For section 10, the file is: ***LargePrint.java***
For section 11, the file is: ***TempConverter.java***
For section 12, the file is: ***ToSeconds.java***
For section 13, the file is: ***SecondsToHRS.java***