really throw the Java compiler for a loop. For example, consider this program, which contains the single error of capitalizing the word `For`:

```
public class CaseApp
{
    public static void main(String[] args)
    {
        For (int i = 0; i<5; i++)
            System.out.println("Hi");
    }
}
```

When you try to compile this program with Java 1.6, the compiler generates a total of four error messages for this one mistake:

```
C:\Java AIO\CaseApp.java:5: '.class' expected
        For (int i = 0; i<5; i++)
                 ^
C:\Java AIO\CaseApp.java:5: illegal start of type
        For (int i = 0; i<5; i++)
                          ^
C:\Java AIO\CaseApp.java:5: not a statement
        For (int i = 0; i<5; i++)
                           ^
C:\Java AIO\CaseApp.java:5: ';' expected
        For (int i = 0; i<5; i++)
                                ^
4 errors      For (int i = 0; i<5; i++)
```

Even though this single mistake generates four error messages, not one of the messages actually points to the problem. The little arrow beneath the source line indicates what part of the line is in error, and none of these error messages have the arrow pointing anywhere near the word `For`! The compiler isn't smart enough to realize that you meant `for` instead of `For`. So it treats `For` as a legitimate identifier, and then complains about everything else on the line that follows it. It would be much more helpful if it generated an error message like this:

```
C:\Java AIO\CaseApp.java:5: 'For' is not a keyword
        For (int i = 0; i<5; i++)
                        ^
```

The moral of the story is that keywords are case-sensitive, and if your program won't compile and the error messages don't make any sense, check for keywords that you've mistakenly capitalized.