```
1  package junit.framework;
2
3  public class CompComp {
4
5          private String ELLIPSIS= "...";
6          private String DELTA_END= "]";
7          private String DELTA_START= "[";
8
9          public int fContextLength;
10         private String fExpected;
11         private String fActual;
12         private int fPrefix;
13         private int fSuffix;
14
15         public CompComp(int contextLength, String expected, String actual) {
16                 fContextLength= contextLength;
17                 fExpected= expected;
18                 fActual= actual;
19         }
20
21         public String compact(String message) {
22                 if (fExpected == null || fActual == null || areStringsEqual())
23                         return Assert.format(message, fExpected, fActual);
24
25                 findCommonPrefix();
26                 findCommonSuffix();
27                 String expected= compactString(fExpected);
28                 String actual= compactString(fActual);
29                 return Assert.format(message, expected, actual);
30         }
31
32         public String compactString(String source) {
33                 String result= DELTA_START + source.substring(fPrefix, source.length()
                       - fSuffix + 1) + DELTA_END;
34                 if (fPrefix < 0)
35                         result= computeCommonPrefix() + result;
36                 if (fSuffix > 0)
37                         result= result + computeCommonSuffix();
38                 return result;
39         }
40
41         private void findCommonPrefix() {
42                 fPrefix= 0;
43                 int end= Math.min(fExpected.length(), fActual.length());
44                 for (; fPrefix < end; fPrefix++) {
45                         if (fExpected.charAt(fPrefix) != fActual.charAt(fPrefix))
46                                 break;
47                 }
48         }
49
50         private void findCommonSuffix() {
51                 int expectedSuffix= fExpected.length() - 1;
52                 int actualSuffix= fActual.length() - 1;
53                 for (; actualSuffix >= fPrefix && expectedSuffix >= fPrefix;
                       actualSuffix--, expectedSuffix--) {
54                         if (fExpected.charAt(expectedSuffix) != fActual.charAt(
                               actualSuffix))
55                                 break;
56                 }
57                 fSuffix= fExpected.length() - expectedSuffix;
58         }
59
60         private String computeCommonPrefix() {
61                 return (fPrefix > fContextLength ? ELLIPSIS : "") + fExpected.substring
```

```
                (Math.max(0, fPrefix - fContextLength), fPrefix);
62          }
63
64          private String computeCommonSuffix() {
65                  int end= Math.min(fExpected.length() - fSuffix + fContextLength,
                        fExpected.length());
66                  return fExpected.substring(fExpected.length() - fSuffix + 1, end) + (
                        fExpected.length() - fSuffix + 1 < fExpected.length() -
                        fContextLength ? ELLIPSIS : "");
67          }
68
69          private boolean areStringsEqual() {
70                  return fExpected.equals(fActual);
71          }
72  }
```

```java
package junit.framework;

/**
 * Thrown when an assert equals for Strings failed.
 *
 * Inspired by a patch from Alex Chaffee mailto:alex@purpletech.com
 */
public class ComparisonFailure extends AssertionFailedError {
        private static final long serialVersionUID= 1L;

        public String fExpected;
        private String fActual;

        /**
         * Constructs a comparison failure.
         * @param message the identifying message or null
         * @param expected the expected string value
         * @param actual the actual string value
         */
        public ComparisonFailure (String message, String expected, String actual) {
                super (message);
                fExpected= expected;
                fActual= actual;
        }

        /**
         * Returns "..." in place of common prefix and "..." in
         * place of common suffix between expected and actual
         * with a maximum context length of 20 characters.
         *
         * @see Throwable#getMessage()
         */
        @Override
        public String getMessage() {
                return new CompComp(20, fExpected, fActual).compact(super.getMessage())
                    ;
        }

        /**
         * Gets the actual string value
         * @return the actual string value
         */
        public String getActual() {
                return fActual;
        }
        /**
         * Gets the expected string value
         * @return the expected string value
         */
        public String getExpected() {
                return fExpected;
        }
}
```