Welcome

Participation is voluntary.

The goal of this study is to evaluate different tools for source code differencing along the following criteria

- accuracy
- understandability
- readability

When constructing software, a developer makes changes to an original version of code in order to produce a modified version. Source code differencing is the process of recovering the changes that a developer made to the original version. A differencing tool takes as input the original version and the modified version. It produces as output a view of the changes to the source code.

For this study, you will be answering questions evaluating four differencing tools on the criteria specified above.

The total study is estimated to take 1-2 hours to complete and is designed to be completed in a single sitting. There are four parts:

1. Sample problem to explain the format and questions.
2. Actual study consisting of fourteen problems. Each will provide: the original and modified source code and four tool outputs. These will be described in detail with the sample problem.
3. Post-questionnaire to gather information on the tools as a whole.
4. Exit questionnaire to gather some information relative to the study. To insure privacy, no personally identifiable information will be gathered.

Note: Each problem is timed to check for valid entry and once you have finished answering a problem and moved on to the next, you are not be permitted to go back.

If you have any questions please contact the investigator at: mdecker6@kent.edu.

If you are 18 years of age or older, understand the statements above, and freely consent to participate in the study, click on the ">>" button to begin the study.

>>

Explanatory Sample

The following provides a simple example of what you will be provided and asked in this study.

Read over all parts of this carefully before moving on to the first problem.

Each problem will provide the following information:

- **Original** and **Modified** source code
- Output of the four tools (**Tool 1**, **Tool 2**, **Tool 3**, and **Tool 4**) for the **Original/Modified** source code

The **Original/Modified** source code is provided in a table with the original and modified code in the left and right respectively. An example is provided below.

All the tools in this study output a side-by-side view of the source-code. That is, original code is annotated with changes on the left side and modified code is annotated with changes on the right side. The tools differ in what code is marked up and how they are annotated. Each of these will be explained below with an example. There is no particular ordering of the tools.

All the tools were run to ignore changes to whitespace, but some may still report them. You are NOT evaluating any of the tools with respect to whitespace changes that they may report. You are also NOT evaluating them on if they report them or not. You ARE evaluating the tools with respect to changes to code.

The basic types of changes are:

- delete - removal of code
- insert - addition of code

All of the tools identify and report these basic changes.

The following are two additional types of changes which can be inferred from deletes and inserts in **Tools 1**, **2**, and **3** and are reported directly by **Tool 4**.

- update - text such as an identifier was replaced (i.e., text deleted and new text inserted)
- move - relocation of code (i.e., deletion of code in one location and its insertion into another location)

Depending on your browser and display device, one or more of the side-by-side views or the Original/Modified source code may be too large to see on your screen all at once. To compensate, you can either scroll left/right/up/down or decrease the font size in your browser.

The output of **Tool 1** and **Tool 2** are images and may be difficult to read on small screens. If you are having difficulty, you can download the images and open them in an image viewer for easier viewing or open them in a separate browser tab and zoom in/out.

The output of **Tool 4** and some of the outputs of **Tool 3** are placed in an inline frame. You can scroll (up/down/left/right) within the inline frame if necessary. If for some reason you have trouble with them, a link is provided for each to open it in a new tab/window. These links are hosted from Kent State University and are perfectly safe. Browsers such as Chrome may incorrectly flag it as unsafe due to the certificate. Proceed to view it anyway. In Chrome, click "Advanced" (bottom left) then "Proceed to web.cs.kent.edu (unsafe)".

Original and Modified Source Code

Original	Modified

`foo bar = get_foo();`
`if(bar != null && !bar.empty())`
`    bar.first().compute();`
`bar = null;`
	`foo bar = foo();`
	`boolean is_empty = (bar != null && !bar.empty());`
	`if(is_empty)`
	`    bar.first().compute();`

Tool 1

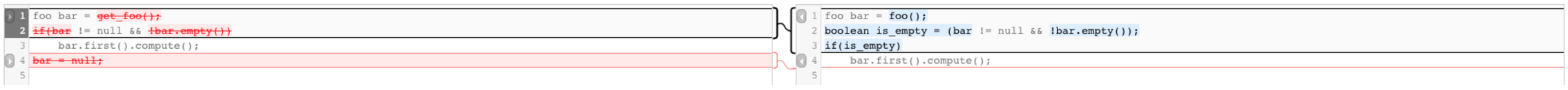
Deleted code is shown on the left side with the line(s) in a red background and inserted code is shown on the right side with the line(s) in a green background. Deleted lines are also preceded with a minus (-) and inserted lines preceded with a plus (+). When identified, within line changes are highlighed with a darker shade of red/green respectively. Line numbers are provided. Lines in the original are aligned horizontally with matching lines in the modified.

Theres are no within line matchings in this example.

```
...  @@ -1,4 +1,4 @@
1    -foo bar = get_foo();                          1  +foo bar = foo();
2    -if(bar != null && !bar.empty())               2  +boolean is_empty = (bar != null && !bar.empty());
                                                    3  +if(is_empty)
3        bar.first().compute();                     4      bar.first().compute();
4    -bar = null;
```

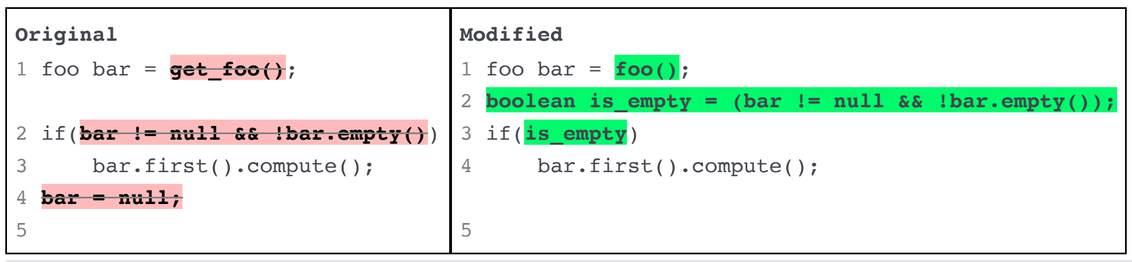
Tool 2

Deleted code is shown on the left side in ~~red~~ with a red background and a line-through. Inserted text is shown on the right side with a blue background. Line numbers are shown for the original and modified code. Lines may not be horizontally aligned.



Tool 3

Deleted code is shown on the left side with a **red** background, line-through, and in bold and inserted code is shown on the right side with a **green** background in bold. Line numbers are provided. Lines in the original are aligned horizontally with matching lines in the modified. Parts of one line in one version may match and be aligned with several in the other. In this case, the single line will be split into multiple and the same line number repeated.



Tool 4

Deleted code is shown on the left with a red background and inserted code is shown on the right with a green-yellow background. The tool displays updated code as a **gold** background in both versions with common characters in grey and deleted/inserted characters in **bold black**. **Tool 4** displays moved code in blue on both sides. For updated/moved code, click on the location in the right side to highlight the corresponding update/move in the left. Placing the mouse over a particular text pops up what syntactic element of source code changed. For a legend of the markup, click "Legend" in the top right. Pressing the key **n** will jumb to the next change, **t** to the top of the code, and **b** to the bottom of the code.

If your screen is too small, the modified code will be placed below the original, the text in the versions will word wrap, and/or you may have to scroll inside each view.

Click to view **Tool 4** in a new tab/window

Questions

The study asks questions pertaining to the following criteria for the tools:

- accuracy
- understandability
- readability

You are to answer each question pertaining only on the given code for that problem and not based on any other prior problems you have seen. You will need to provide an answer for all questions to continue to the next problem. The tools are given in no particular order and you may answer them in any order.

You will not be able to go back to a problem after continuing to the next problem. Each problem is timed to ensure valid entry.

Accuracy Questions

The first three questions pertain to accuracy. Accuracy is a measure of the tools ability to recover the changes the developer would have made to the original code inorder to create the modified version.

The first two questions are:

- **How much of the code is marked as unchanged that should be marked otherwise?**
- **How much of the code is marked as changed that should be marked otherwise?**

Unchanged code is code that is not modified between the two versions.
Changed code refers to any code that is deleted, inserted, updated, or moved.

The first question asks how much of the code did the tool report as not modified but you believe the developer must have changed it in some way.

The second question asks how much of the code the tool reported as changed but you believe the developer either did not change it or performed a different type of change.

The third question is:

- **Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.**

This question asks you to rank the tools by assigning points between the tools relative to how well each tool marks the developers changes. Give the most points to the tool that captured the changes the best and the least points to the tool that did the worst job. If any of the tools are equal in how well they perform, give them an equal amount of points. You may give each tool a number between 0-100 and different tools the same points as long as they total 100.

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0 Points
Tool 2	0 Points
Tool 3	0 Points
Tool 4	0 Points
Total	0 Points

Understandability Question

The following question measures the understandability criteria

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
This question asks you to rank the tools by assigning points between the tools relative to how easy each tool's output is to understand. Give the most points to the tool that is easiest to understand and the least points to the tool that is the hardest to understand. If any of the tools are equal in how easy their output is to understand, give them an equal amount of points. You may give each tool a number between 0-100 and different tools the same points as long as they total 100.

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0 Points

Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Readability Question

The following question measure the readability criteria.

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
This question asks you to rank the tools by assigning points between the tools relative to how readable each tool's output is. Give the most points to the tool that is the most readable and the least points to the tool that is the least readable. If any of the tools are equal in how readable their output is, give them an equal amount of points. You may give each tool a number between 0-100 and different tools the same points as long as they total 100.

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Preference Question

Answer the following about your preferences toward the **Original/Modified** and tools.

Rank the following by distributing 100 points among each relative to your preference.
This question asks you to rank the **Original/Modifed** and the tools by assigning points between them relative to how much you prefer each one. Give the most points to the one you most prefer and the least points to the one that you least prefer. If any are preferred equally, give them an equal amount of points. You may give each a number between 0-100 and different ones the same points as long as they total 100.

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments Question

Additional Comments?
This question asks for any additional comments you may have. This question is **optional**. Any comments would be appreciated.

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

@Test
public void testGeoHashValue() throws Exception {
    String mapping = XContentFactory.jsonBuilder()
            .startObject().startObject("type")
            .startObject("properties").startObject("point")
            .field("type", "geo_point").field("lat_lon", true)
            .field("geohash", true).endObject().endObject()
            .endObject().endObject().string();
 	
    DocumentMapper defaultMapper
    = MapperTestUtils.newParser().parse(mapping);
 	
    ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
            .jsonBuilder()
            .startObject()
            .field("point", GeoHashUtils.encode(1.2, 1.3))
            .endObject()
            .bytes());
 	
    assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
    assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
    assertThat(doc.rootDoc().get("point.geohash"),
        equalTo(GeoHashUtils.encode(1.2, 1.3)));
}
	@Test
	public void testGeoHashValue() throws Exception {
	    String mapping = XContentFactory.jsonBuilder()
	            .startObject().startObject("type")
	            .startObject("properties").startObject("point")
	            .field("type", "geo_point").field("lat_lon", true)
	            .field("geohash", true).endObject().endObject()
	            .endObject().endObject().string();
	
	    DocumentMapper defaultMapper
	    = createIndex("test").mapperService().documentMapperParser().parse(mapping);
	
	    ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
	            .jsonBuilder()
	            .startObject()
	            .field("point", GeoHashUtils.encode(1.2, 1.3))
	            .endObject()
	            .bytes());
	
	    assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
	    assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
	    assertThat(doc.rootDoc().get("point.geohash"),
	        equalTo(GeoHashUtils.encode(1.2, 1.3)));
	}

Tool 1

```
...  @@ -1,14 +1,14 @@
 1    @Test
 2    public void testGeoHashValue() throws Exception {
 3        String mapping = XContentFactory.jsonBuilder()
 4                .startObject().startObject("type")
 5                .startObject("properties").startObject("point")
 6                .field("type", "geo_point").field("lat_lon", true)
 7                .field("geohash", true).endObject().endObject()
 8                .endObject().endObject().string();
 9
10        DocumentMapper defaultMapper
11    -   = MapperTestUtils.newParser().parse(mapping);
12
13        ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
14                .jsonBuilder()
15                .startObject()
16                .field("point", GeoHashUtils.encode(1.2, 1.3))
17                .endObject()
18                .bytes());
19
20        assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
21        assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
22        assertThat(doc.rootDoc().get("point.geohash"),
23            equalTo(GeoHashUtils.encode(1.2, 1.3)));
24    }
```

```
 1  @Test
 2  public void testGeoHashValue() throws Exception {
 3      String mapping = XContentFactory.jsonBuilder()
 4              .startObject().startObject("type")
 5              .startObject("properties").startObject("point")
 6              .field("type", "geo_point").field("lat_lon", true)
 7              .field("geohash", true).endObject().endObject()
 8              .endObject().endObject().string();
 9
10      DocumentMapper defaultMapper
11  +   = createIndex("test").mapperService().documentMapperParser().parse(mapping);
12
13      ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
14              .jsonBuilder()
15              .startObject()
16              .field("point", GeoHashUtils.encode(1.2, 1.3))
17              .endObject()
18              .bytes());
19
20      assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
21      assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
22      assertThat(doc.rootDoc().get("point.geohash"),
23          equalTo(GeoHashUtils.encode(1.2, 1.3)));
24  }
```

Tool 2

```
 1  @Test
 2  public void testGeoHashValue() throws Exception {
 3      String mapping = XContentFactory.jsonBuilder()
 4              .startObject().startObject("type")
 5              .startObject("properties").startObject("point")
 6              .field("type", "geo_point").field("lat_lon", true)
 7              .field("geohash", true).endObject().endObject()
 8              .endObject().endObject().string();
 9
10      DocumentMapper defaultMapper
11      = MapperTestUtils.newParser().parse(mapping);
12
13      ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
14              .jsonBuilder()
15              .startObject()
16              .field("point", GeoHashUtils.encode(1.2, 1.3))
17              .endObject()
18              .bytes());
19
20      assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
21      assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
22      assertThat(doc.rootDoc().get("point.geohash"),
23          equalTo(GeoHashUtils.encode(1.2, 1.3)));
24  }
25
```

```
 1  @Test
 2  public void testGeoHashValue() throws Exception {
 3      String mapping = XContentFactory.jsonBuilder()
 4              .startObject().startObject("type")
 5              .startObject("properties").startObject("point")
 6              .field("type", "geo_point").field("lat_lon", true)
 7              .field("geohash", true).endObject().endObject()
 8              .endObject().endObject().string();
 9
10      DocumentMapper defaultMapper
11      = createIndex("test").mapperService().documentMapperParser().parse(mapping);
12
13      ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
14              .jsonBuilder()
15              .startObject()
16              .field("point", GeoHashUtils.encode(1.2, 1.3))
17              .endObject()
18              .bytes());
19
20      assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
21      assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
22      assertThat(doc.rootDoc().get("point.geohash"),
23          equalTo(GeoHashUtils.encode(1.2, 1.3)));
24  }
25
```

Tool 3

Original	Modified
1 @Test 2 public void testGeoHashValue() throws Exception {	1 @Test 2 public void testGeoHashValue() throws Exception {

```
    String mapping = XContentFactory.jsonBuilder()
            .startObject().startObject("type")
            .startObject("properties").startObject("point")
            .field("type", "geo_point").field("lat_lon", true)
            .field("geohash", true).endObject().endObject()
            .endObject().endObject().string();

    DocumentMapper defaultMapper
    = MapperTestUtils.newParser().parse(mapping);

    ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
            .jsonBuilder()
            .startObject()
            .field("point", GeoHashUtils.encode(1.2, 1.3))
            .endObject()
            .bytes());

    assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
    assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
    assertThat(doc.rootDoc().get("point.geohash"),
        equalTo(GeoHashUtils.encode(1.2, 1.3)));
}
```

```
    String mapping = XContentFactory.jsonBuilder()
            .startObject().startObject("type")
            .startObject("properties").startObject("point")
            .field("type", "geo_point").field("lat_lon", true)
            .field("geohash", true).endObject().endObject()
            .endObject().endObject().string();

    DocumentMapper defaultMapper
    = createIndex("test").mapperService().documentMapperParser().parse(mapping);

    ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
            .jsonBuilder()
            .startObject()
            .field("point", GeoHashUtils.encode(1.2, 1.3))
            .endObject()
            .bytes());

    assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
    assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
    assertThat(doc.rootDoc().get("point.geohash"),
        equalTo(GeoHashUtils.encode(1.2, 1.3)));
}
```

Tool 4

Legend | Shortcuts

original.java

```
@Test
public void testGeoHashValue() throws Exception {
    String mapping = XContentFactory.jsonBuilder()
            .startObject().startObject("type")
            .startObject("properties").startObject("point")
            .field("type", "geo_point").field("lat_lon", true)
            .field("geohash", true).endObject().endObject()
            .endObject().endObject().string();

    DocumentMapper defaultMapper
    = MapperTestUtils.newParser().parse(mapping);

    ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
            .jsonBuilder()
            .startObject()
            .field("point", GeoHashUtils.encode(1.2, 1.3))
            .endObject()
            .bytes());

    assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
    assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
    assertThat(doc.rootDoc().get("point.geohash"),
        equalTo(GeoHashUtils.encode(1.2, 1.3)));
}
```

modified.java

```
@Test
public void testGeoHashValue() throws Exception {
    String mapping = XContentFactory.jsonBuilder()
            .startObject().startObject("type")
            .startObject("properties").startObject("point")
            .field("type", "geo_point").field("lat_lon", true)
            .field("geohash", true).endObject().endObject()
            .endObject().endObject().string();

    DocumentMapper defaultMapper
    = createIndex("test").mapperService().documentMapperParser().parse(mapping);

    ParsedDocument doc = defaultMapper.parse("type", "1", XContentFactory
            .jsonBuilder()
            .startObject()
            .field("point", GeoHashUtils.encode(1.2, 1.3))
            .endObject()
            .bytes());

    assertThat(doc.rootDoc().getField("point.lat"), notNullValue());
    assertThat(doc.rootDoc().getField("point.lon"), notNullValue());
    assertThat(doc.rootDoc().get("point.geohash"),
        equalTo(GeoHashUtils.encode(1.2, 1.3)));
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

public Field getCompletionField(ContextMapping.Context ctx,
String input, BytesRef payload) {
final String originalInput = input;
if (input.length() > maxInputLength) {
final int len = correctSubStringLen(input,
Math.min(maxInputLength, input.length()));
input = input.substring(0, len);
}
for (int i = 0; i < input.length(); i++) {
if (isReservedChar(input.charAt(i))) {
throw new IllegalArgumentException("Illegal input ["
+ originalInput + "] UTF-16 codepoint [0x"
+ Integer.toHexString((int) input.charAt(i))
.toUpperCase(Locale.ROOT)
+ "] at position " + i + " is a reserved character");
}
}
return new SuggestField(fieldType.names().indexName(),
ctx, input, this.fieldType, payload,
fieldType().analyzingSuggestLookupProvider);
}
	public Field getCompletionField(ContextMapping.Context ctx,
	String input, BytesRef payload) {
	final String originalInput = input;
	if (input.length() > maxInputLength) {
	final int len = correctSubStringLen(input,
	Math.min(maxInputLength, input.length()));
	input = input.substring(0, len);
	}
	for (int i = 0; i < input.length(); i++) {
	if (isReservedChar(input.charAt(i))) {
	throw new IllegalArgumentException("Illegal input ["
	+ originalInput + "] UTF-16 codepoint [0x"
	+ Integer.toHexString((int) input.charAt(i))
	.toUpperCase(Locale.ROOT)
	+ "] at position " + i + " is a reserved character");
	}
	}
	return new SuggestField(fieldType().names().indexName(),
	ctx, input, fieldType(), payload,
	fieldType().analyzingSuggestLookupProvider);
	}

Tool 1

```
...  @@ -1,21 +1,21 @@
 1    public Field getCompletionField(ContextMapping.Context ctx,
 2                                    String input, BytesRef payload) {
 3        final String originalInput = input;
 4        if (input.length() > maxInputLength) {
 5            final int len = correctSubStringLen(input,
 6                Math.min(maxInputLength, input.length()));
 7            input = input.substring(0, len);
 8        }
 9        for (int i = 0; i < input.length(); i++) {
10            if (isReservedChar(input.charAt(i))) {
11                throw new IllegalArgumentException("Illegal input ["
12                        + originalInput + "] UTF-16 codepoint  [0x"
13                        + Integer.toHexString((int) input.charAt(i))
14                            .toUpperCase(Locale.ROOT)
15                        + "] at position " + i + " is a reserved character");
16            }
17        }
18  -     return new SuggestField(fieldType.names().indexName(),
19  -         ctx, input, this.fieldType, payload,
20            fieldType().analyzingSuggestLookupProvider);
21    }
```

```
 1    public Field getCompletionField(ContextMapping.Context ctx,
 2                                    String input, BytesRef payload) {
 3        final String originalInput = input;
 4        if (input.length() > maxInputLength) {
 5            final int len = correctSubStringLen(input,
 6                Math.min(maxInputLength, input.length()));
 7            input = input.substring(0, len);
 8        }
 9        for (int i = 0; i < input.length(); i++) {
10            if (isReservedChar(input.charAt(i))) {
11                throw new IllegalArgumentException("Illegal input ["
12                        + originalInput + "] UTF-16 codepoint  [0x"
13                        + Integer.toHexString((int) input.charAt(i))
14                            .toUpperCase(Locale.ROOT)
15                        + "] at position " + i + " is a reserved character");
16            }
17        }
18  +     return new SuggestField(fieldType().names().indexName(),
19  +         ctx, input, fieldType(), payload,
20            fieldType().analyzingSuggestLookupProvider);
21    }
```

Tool 2

```
 1  public Field getCompletionField(ContextMapping.Context ctx,
 2                                  String input, BytesRef payload) {
 3      final String originalInput = input;
 4      if (input.length() > maxInputLength) {
 5          final int len = correctSubStringLen(input,
 6              Math.min(maxInputLength, input.length()));
 7          input = input.substring(0, len);
 8      }
 9      for (int i = 0; i < input.length(); i++) {
10          if (isReservedChar(input.charAt(i))) {
11              throw new IllegalArgumentException("Illegal input ["
12                      + originalInput + "] UTF-16 codepoint  [0x"
13                      + Integer.toHexString((int) input.charAt(i))
14                          .toUpperCase(Locale.ROOT)
15                      + "] at position " + i + " is a reserved character");
16          }
17      }
18      return new SuggestField(fieldType.names().indexName(),
19          ctx, input, this.fieldType, payload,
20          fieldType().analyzingSuggestLookupProvider);
21  }
22
```

```
 1  public Field getCompletionField(ContextMapping.Context ctx,
 2                                  String input, BytesRef payload) {
 3      final String originalInput = input;
 4      if (input.length() > maxInputLength) {
 5          final int len = correctSubStringLen(input,
 6              Math.min(maxInputLength, input.length()));
 7          input = input.substring(0, len);
 8      }
 9      for (int i = 0; i < input.length(); i++) {
10          if (isReservedChar(input.charAt(i))) {
11              throw new IllegalArgumentException("Illegal input ["
12                      + originalInput + "] UTF-16 codepoint  [0x"
13                      + Integer.toHexString((int) input.charAt(i))
14                          .toUpperCase(Locale.ROOT)
15                      + "] at position " + i + " is a reserved character");
16          }
17      }
18      return new SuggestField(fieldType().names().indexName(),
19          ctx, input, fieldType(), payload,
20          fieldType().analyzingSuggestLookupProvider);
21  }
22
```

Tool 3

Original

```
 1  public Field getCompletionField(ContextMapping.Context ctx,
 2                                  String input, BytesRef payload) {
 3      final String originalInput = input;
 4      if (input.length() > maxInputLength) {
 5          final int len = correctSubStringLen(input,
 6              Math.min(maxInputLength, input.length()));
 7          input = input.substring(0, len);
 8      }
 9      for (int i = 0; i < input.length(); i++) {
```

Modified

```
 1  public Field getCompletionField(ContextMapping.Context ctx,
 2                                  String input, BytesRef payload) {
 3      final String originalInput = input;
 4      if (input.length() > maxInputLength) {
 5          final int len = correctSubStringLen(input,
 6              Math.min(maxInputLength, input.length()));
 7          input = input.substring(0, len);
 8      }
 9      for (int i = 0; i < input.length(); i++) {
```

```
10         if (isReservedChar(input.charAt(i))) {
11             throw new IllegalArgumentException("Illegal input ["
12                     + originalInput + "] UTF-16 codepoint  [0x"
13                     + Integer.toHexString((int) input.charAt(i))
14                         .toUpperCase(Locale.ROOT)
15                     + "] at position " + i + " is a reserved character");
16         }
17     }
18     return new SuggestField(fieldType.names().indexName(),
19         ctx, input, this.fieldType, payload,
20         fieldType().analyzingSuggestLookupProvider);
21 }
22
```

```
10         if (isReservedChar(input.charAt(i))) {
11             throw new IllegalArgumentException("Illegal input ["
12                     + originalInput + "] UTF-16 codepoint  [0x"
13                     + Integer.toHexString((int) input.charAt(i))
14                         .toUpperCase(Locale.ROOT)
15                     + "] at position " + i + " is a reserved character");
16         }
17     }
18     return new SuggestField(fieldType().names().indexName(),
19         ctx, input, fieldType(), payload,
20         fieldType().analyzingSuggestLookupProvider);
21 }
22
```

Tool 4

Legend | Shortcuts

original.java

```
public Field getCompletionField(ContextMapping.Context ctx,
                                String input, BytesRef payload) {
    final String originalInput = input;
    if (input.length() > maxInputLength) {
        final int len = correctSubStringLen(input,
            Math.min(maxInputLength, input.length()));
        input = input.substring(0, len);
    }
    for (int i = 0; i < input.length(); i++) {
        if (isReservedChar(input.charAt(i))) {
            throw new IllegalArgumentException("Illegal input ["
                    + originalInput + "] UTF-16 codepoint  [0x"
                    + Integer.toHexString((int) input.charAt(i))
                        .toUpperCase(Locale.ROOT)
                    + "] at position " + i + " is a reserved character");
        }
    }
    return new SuggestField(fieldType.names().indexName(),
        ctx, input, this.fieldType, payload,
        fieldType().analyzingSuggestLookupProvider);
}
```

modified.java

```
public Field getCompletionField(ContextMapping.Context ctx,
                                String input, BytesRef payload) {
    final String originalInput = input;
    if (input.length() > maxInputLength) {
        final int len = correctSubStringLen(input,
            Math.min(maxInputLength, input.length()));
        input = input.substring(0, len);
    }
    for (int i = 0; i < input.length(); i++) {
        if (isReservedChar(input.charAt(i))) {
            throw new IllegalArgumentException("Illegal input ["
                    + originalInput + "] UTF-16 codepoint  [0x"
                    + Integer.toHexString((int) input.charAt(i))
                        .toUpperCase(Locale.ROOT)
                    + "] at position " + i + " is a reserved character");
        }
    }
    return new SuggestField(fieldType().names().indexName(),
        ctx, input, fieldType(), payload,
        fieldType().analyzingSuggestLookupProvider);
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

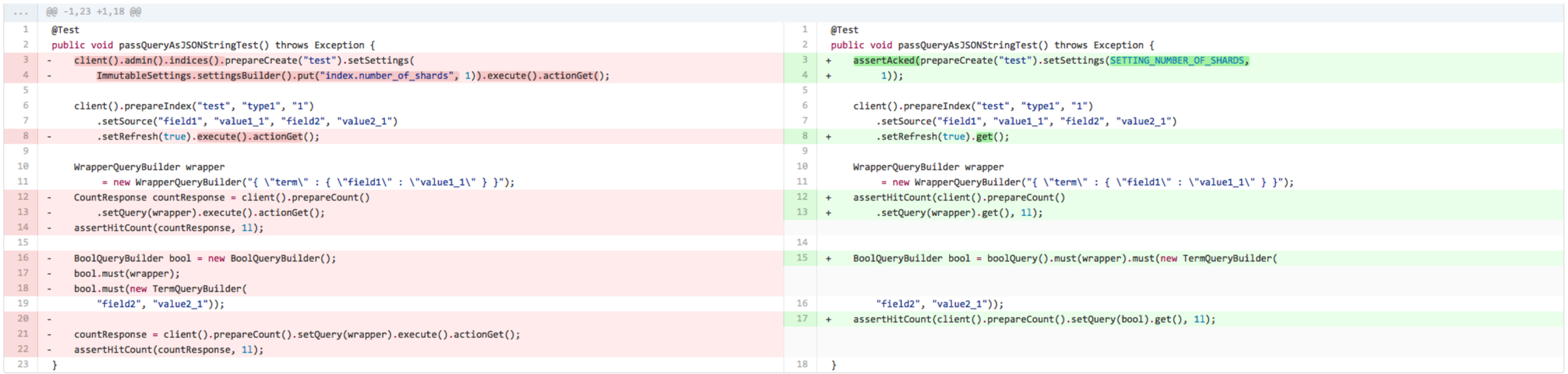
>>

Original and Modified Source Code

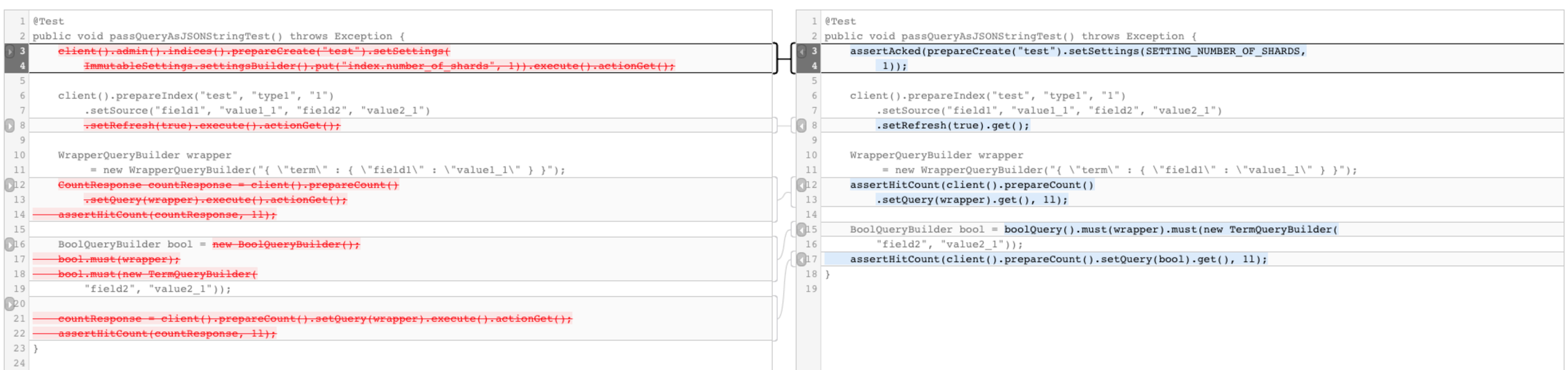
Original	Modified

`@Test`
`public void passQueryAsJSONStringTest() throws Exception {`
`  client().admin().indices().prepareCreate("test").setSettings(`
`  ImmutableSettings.settingsBuilder().put("index.number_of_shards", 1)).execute().actionGet();`
 	
`  client().prepareIndex("test", "type1", "1")`
`  .setSource("field1", "value1_1", "field2", "value2_1")`
`  .setRefresh(true).execute().actionGet();`
 	
`  WrapperQueryBuilder wrapper`
`  = new WrapperQueryBuilder("{ \"term\" : { \"field1\" : \"value1_1\" } }");`
`  CountResponse countResponse = client().prepareCount()`
`  .setQuery(wrapper).execute().actionGet();`
`  assertHitCount(countResponse, 1l);`
 	
`  BoolQueryBuilder bool = new BoolQueryBuilder();`
`  bool.must(wrapper);`
`  bool.must(new TermQueryBuilder(`
`  "field2", "value2_1"));`
 	
`  countResponse = client().prepareCount().setQuery(wrapper).execute().actionGet();`
`  assertHitCount(countResponse, 1l);`
`}`
	`@Test`
	`public void passQueryAsJSONStringTest() throws Exception {`
	`  assertAcked(prepareCreate("test").setSettings(SETTING_NUMBER_OF_SHARDS,`
	`  1));`
	
	`  client().prepareIndex("test", "type1", "1")`
	`  .setSource("field1", "value1_1", "field2", "value2_1")`
	`  .setRefresh(true).get();`
	
	`  WrapperQueryBuilder wrapper`
	`  = new WrapperQueryBuilder("{ \"term\" : { \"field1\" : \"value1_1\" } }");`
	`  assertHitCount(client().prepareCount()`
	`  .setQuery(wrapper).get(), 1l);`
	
	`  BoolQueryBuilder bool = boolQuery().must(wrapper).must(new TermQueryBuilder(`
	`  "field2", "value2_1"));`
	`  assertHitCount(client().prepareCount().setQuery(bool).get(), 1l);`
	`}`

Tool 1



Tool 2



Tool 3

Original	Modified

`@Test`
`public void passQueryAsJSONStringTest() throws Exception {`
`  client().admin().indices().prepareCreate("test").setSettings(`
`  ImmutableSettings.settingsBuilder().put("index.number_of_shards", 1)).execute().actionGet();`
 	
	`@Test`
	`public void passQueryAsJSONStringTest() throws Exception {`
	`  assertAcked(prepareCreate("test").setSettings(SETTING_NUMBER_OF_SHARDS,`
	`  1));`

```
    client().prepareIndex("test", "type1", "1")
        .setSource("field1", "value1_1", "field2", "value2_1")
        .setRefresh(true).execute().actionGet();

    WrapperQueryBuilder wrapper
         = new WrapperQueryBuilder("{ \"term\" : { \"field1\" : \"value1_1\" } }");
    CountResponse countResponse = client().prepareCount()
        .setQuery(wrapper).execute().actionGet();
    assertHitCount(countResponse, 1l);

    BoolQueryBuilder bool = new BoolQueryBuilder();
    bool.must(wrapper);
    bool.must(new TermQueryBuilder(
        "field2", "value2_1"));

    countResponse = client().prepareCount().setQuery(wrapper).execute().actionGet();
    assertHitCount(countResponse, 1l);
}
```

```
    client().prepareIndex("test", "type1", "1")
        .setSource("field1", "value1_1", "field2", "value2_1")
        .setRefresh(true).get();

    WrapperQueryBuilder wrapper
         = new WrapperQueryBuilder("{ \"term\" : { \"field1\" : \"value1_1\" } }");
    assertHitCount(client().prepareCount()
        .setQuery(wrapper).get(), 1l);

    BoolQueryBuilder bool = boolQuery().must(wrapper).must(new TermQueryBuilder(
        "field2", "value2_1"));

    assertHitCount( client().prepareCount().setQuery(bool).get(), 1l);
}
```

Tool 4

Legend | Shortcuts

original.java

```
@Test
public void passQueryAsJSONStringTest() throws Exception {
    client().admin().indices().prepareCreate("test").setSettings(
        ImmutableSettings.settingsBuilder().put("index.number_of_shards", 1)).execute().actionGet();

    client().prepareIndex("test", "type1", "1")
        .setSource("field1", "value1_1", "field2", "value2_1")
        .setRefresh(true).execute().actionGet();

    WrapperQueryBuilder wrapper
         = new WrapperQueryBuilder("{ \"term\" : { \"field1\" : \"value1_1\" } }");
    CountResponse countResponse = client().prepareCount()
        .setQuery(wrapper).execute().actionGet();
    assertHitCount(countResponse, 1l);

    BoolQueryBuilder bool = new BoolQueryBuilder();
    bool.must(wrapper);
    bool.must(new TermQueryBuilder(
        "field2", "value2_1"));

    countResponse = client().prepareCount().setQuery(wrapper).execute().actionGet();
    assertHitCount(countResponse, 1l);
}
```

modified.java

```
@Test
public void passQueryAsJSONStringTest() throws Exception {
    assertAcked(prepareCreate("test").setSettings(SETTING_NUMBER_OF_SHARDS,
        1));

    client().prepareIndex("test", "type1", "1")
        .setSource("field1", "value1_1", "field2", "value2_1")
        .setRefresh(true).get();

    WrapperQueryBuilder wrapper
         = new WrapperQueryBuilder("{ \"term\" : { \"field1\" : \"value1_1\" } }");
    assertHitCount(client().prepareCount()
        .setQuery(wrapper).get(), 1l);

    BoolQueryBuilder bool = boolQuery().must(wrapper).must(new TermQueryBuilder(
        "field2", "value2_1"));
    assertHitCount(client().prepareCount().setQuery(bool).get(), 1l);
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.

(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

`public boolean isShutdown() { return shutdown; }`	`@Override`
	`public boolean isShutdown() {`
	`  return false;`
	`}`

Tool 1

```
...  @@ -1 +1,4 @@
1    -public boolean isShutdown() { return shutdown; }
```

```
1  +@Override
2  +public boolean isShutdown() {
3  +  return false;
4  +}
```

Tool 2

```
1  public boolean isShutdown() { return shutdown; }
2
```

```
1  @Override
2  public boolean isShutdown() {
3    return false;
4  }
5
```

Tool 3

Original	Modified
1 public boolean isShutdown() {	1 @Override
1 return shutdown;	2 public boolean isShutdown() {
1 }	3 return false;
2	4 }
	5

Tool 4

Legend | Shortcuts

original.java

```
public boolean isShutdown() { return shutdown; }
```

modified.java

```
@Override
public boolean isShutdown() {
  return false;
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

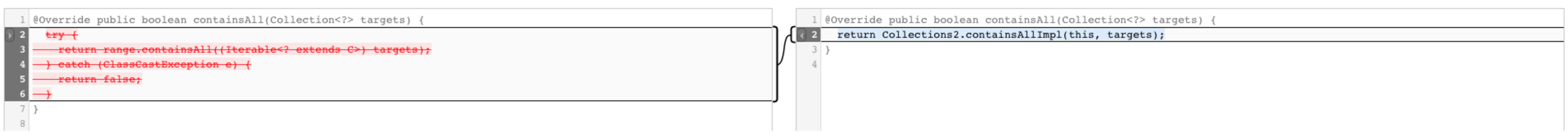
Original and Modified Source Code

Original	Modified

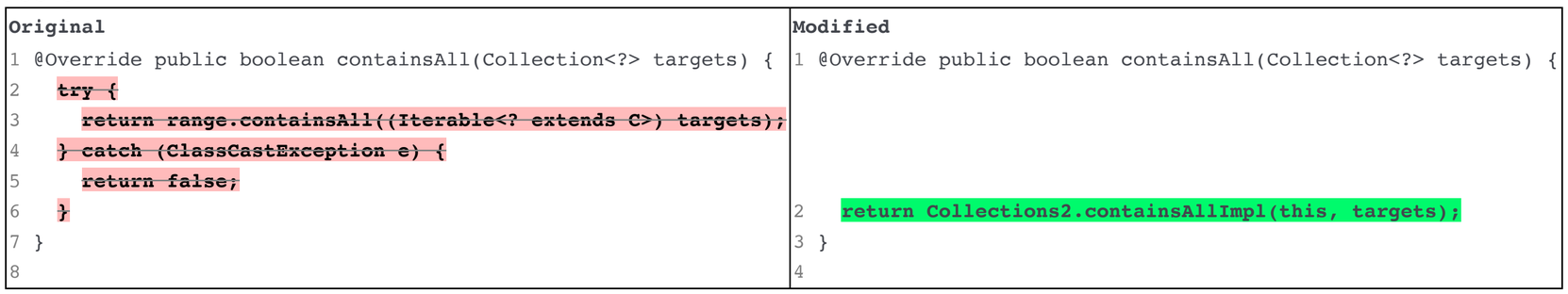
`@Override public boolean containsAll(Collection<?> targets) {`
`  try {`
`  return range.containsAll((Iterable<? extends C>) targets);`
`  } catch (ClassCastException e) {`
`  return false;`
`  }`
`}`
	`@Override public boolean containsAll(Collection<?> targets) {`
	`  return Collections2.containsAllImpl(this, targets);`
	`}`

Tool 1

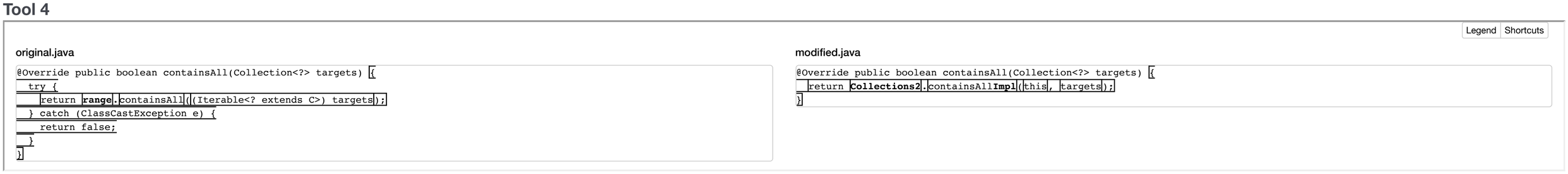
Tool 2



Tool 3



Tool 4



Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount

Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

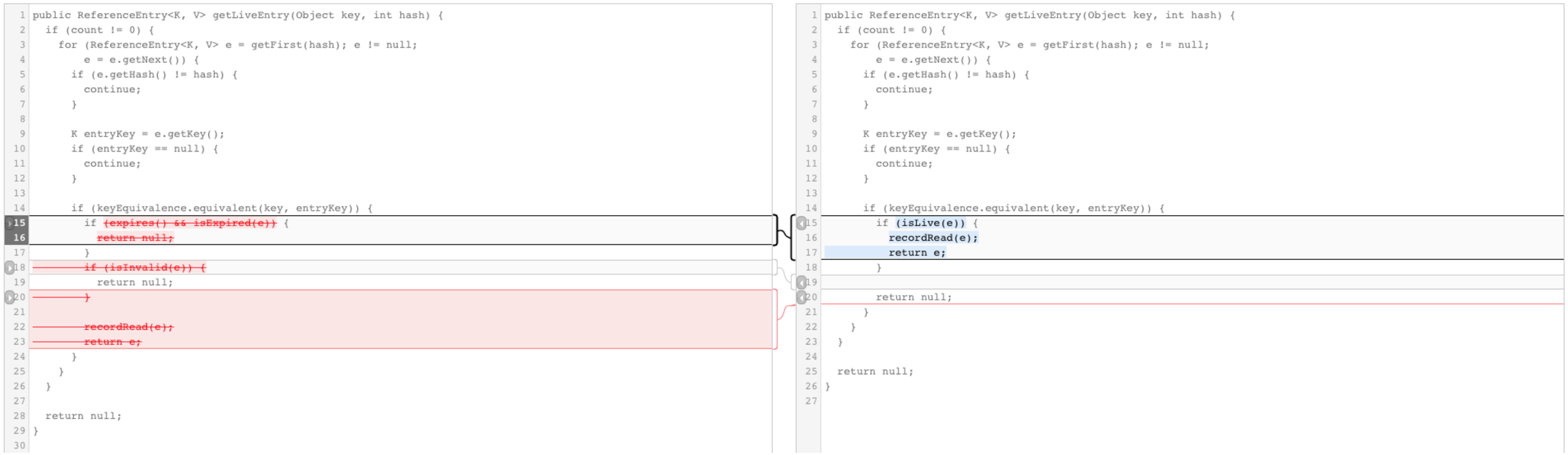
public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
  if (count != 0) {
  for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
  e = e.getNext()) {
  if (e.getHash() != hash) {
  continue;
  }
 	
  K entryKey = e.getKey();
  if (entryKey == null) {
  continue;
  }
 	
  if (keyEquivalence.equivalent(key, entryKey)) {
  if (expires() && isExpired(e)) {
  return null;
  }
  if (isInvalid(e)) {
  return null;
  }
 	
  recordRead(e);
  return e;
  }
  }
  }
 	
  return null;
}
	public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
	  if (count != 0) {
	  for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
	  e = e.getNext()) {
	  if (e.getHash() != hash) {
	  continue;
	  }
	
	  K entryKey = e.getKey();
	  if (entryKey == null) {
	  continue;
	  }
	
	  if (keyEquivalence.equivalent(key, entryKey)) {
	  if (isLive(e)) {
	  recordRead(e);
	  return e;
	  }
	
	  return null;
	  }
	  }
	  }
	
	  return null;
	}

Tool 1

```
...  @@ -1,27 +1,24 @@
1    public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
2      if (count != 0) {
3        for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
4            e = e.getNext()) {
5          if (e.getHash() != hash) {
6            continue;
7          }
8
9          K entryKey = e.getKey();
10         if (entryKey == null) {
11           continue;
12         }
13
14         if (keyEquivalence.equivalent(key, entryKey)) {
15 -         if (expires() && isExpired(e)) {
16 -           return null;
17 -         }
18 -         if (isInvalid(e)) {
19 -           return null;
20 -         }
21 -
22           recordRead(e);
23           return e;
24         }
25     }
26   }
27
28   return null;
29 }
```

```
1    public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
2      if (count != 0) {
3        for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
4            e = e.getNext()) {
5          if (e.getHash() != hash) {
6            continue;
7          }
8
9          K entryKey = e.getKey();
10         if (entryKey == null) {
11           continue;
12         }
13
14         if (keyEquivalence.equivalent(key, entryKey)) {
15 +         if (isLive(e)) {
16           recordRead(e);
17           return e;
18         }
19 +
20 +       return null;
21 +     }
22     }
23   }
24
25   return null;
26 }
```

Tool 2



```
public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
  if (count != 0) {
    for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
        e = e.getNext()) {
      if (e.getHash() != hash) {
        continue;
      }

      K entryKey = e.getKey();
      if (entryKey == null) {
        continue;
      }

      if (keyEquivalence.equivalent(key, entryKey)) {
        if (expires() && isExpired(e)) {
          return null;
        }
        if (isInvalid(e)) {
          return null;
        }

        recordRead(e);
        return e;
      }
    }
  }

  return null;
}
```

Tool 3

Original

```
public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
  if (count != 0) {
    for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
        e = e.getNext()) {
      if (e.getHash() != hash) {
        continue;
      }

      K entryKey = e.getKey();
      if (entryKey == null) {
        continue;
      }

      if (keyEquivalence.equivalent(key, entryKey)) {
        if (expires() && isExpired(e)) {
          return null;
        }
        if (isInvalid(e)) {
          return null;
        }

        recordRead(e);
        return e;
      }
    }
  }

  return null;
}
```

Modified

```
public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
  if (count != 0) {
    for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
        e = e.getNext()) {
      if (e.getHash() != hash) {
        continue;
      }

      K entryKey = e.getKey();
      if (entryKey == null) {
        continue;
      }

      if (keyEquivalence.equivalent(key, entryKey)) {

        if (isLive(e)) {
          recordRead(e);
          return e;
        }

        return null;
      }
    }
  }

  return null;
}
```

Tool 4

Legend | Shortcuts

original.java

```
public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
  if (count != 0) {
    for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
        e = e.getNext()) {
      if (e.getHash() != hash) {
        continue;
      }

      K entryKey = e.getKey();
      if (entryKey == null) {
        continue;
      }

      if (keyEquivalence.equivalent(key, entryKey)) {
        if (expires() && isExpired(e)) {
          return null;
        }
        if (isInvalid(e)) {
          return null;
        }

        recordRead(e);
        return e;
      }
    }
  }

  return null;
}
```

modified.java

```
public ReferenceEntry<K, V> getLiveEntry(Object key, int hash) {
  if (count != 0) {
    for (ReferenceEntry<K, V> e = getFirst(hash); e != null;
        e = e.getNext()) {
      if (e.getHash() != hash) {
        continue;
      }

      K entryKey = e.getKey();
      if (entryKey == null) {
        continue;
      }

      if (keyEquivalence.equivalent(key, entryKey)) {
        if (isLive(e)) {
          recordRead(e);
          return e;
        }

        return null;
      }
    }
  }

  return null;
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4		

Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

private void convertFile(File file) {
  try {
  InplaceFileConverter fc = new InplaceFileConverter(lineConverter);
  byte[] ba = fc.readFile(file);
  fc.convert(file, ba);
  } catch (IOException exc) {
  addException(new ConversionException(exc.toString()));
  }
}
	private void convertFile(File file) {
	  try {
	  InplaceFileConverter fc = new InplaceFileConverter(ruleSet);
	  fc.convert(file);
	  } catch (IOException exc) {
	  addException(new ConversionException(exc.toString()));
	  }
	}

Tool 1

```
...  @@ -1,8 +1,7 @@
1    private void convertFile(File file) {
2      try {
3  -     InplaceFileConverter fc = new InplaceFileConverter(lineConverter);
4  -     byte[] ba = fc.readFile(file);
5  -     fc.convert(file, ba);
6      } catch (IOException exc) {
7        addException(new ConversionException(exc.toString()));
8      }
9    }
```

```
1    private void convertFile(File file) {
2      try {
3  +     InplaceFileConverter fc = new InplaceFileConverter(ruleSet);
4  +     fc.convert(file);
5      } catch (IOException exc) {
6        addException(new ConversionException(exc.toString()));
7      }
8    }
```

Tool 2

```
1  private void convertFile(File file) {
2    try {
3      InplaceFileConverter fc = new InplaceFileConverter(lineConverter);
4      byte[] ba = fc.readFile(file);
5      fc.convert(file, ba);
6    } catch (IOException exc) {
7      addException(new ConversionException(exc.toString()));
8    }
9  }
10
```

```
1  private void convertFile(File file) {
2    try {
3      InplaceFileConverter fc = new InplaceFileConverter(ruleSet);
4      fc.convert(file);
5    } catch (IOException exc) {
6      addException(new ConversionException(exc.toString()));
7    }
8  }
9
```

Tool 3

Original	Modified

1 private void convertFile(File file) {
2 try {
3 InplaceFileConverter fc = new InplaceFileConverter(lineConverter);
4 byte[] ba = fc.readFile(file);
5 fc.convert(file, ba);
6 } catch (IOException exc) {
7 addException(new ConversionException(exc.toString()));
8 }
9 }
10
	1 private void convertFile(File file) {
	2 try {
	3 InplaceFileConverter fc = new InplaceFileConverter(ruleSet);
	4 fc.convert(file);
	5 } catch (IOException exc) {
	6 addException(new ConversionException(exc.toString()));
	7 }
	8 }
	9

Tool 4

Legend | Shortcuts

original.java

```
private void convertFile(File file) {
  try {
    InplaceFileConverter fc = new InplaceFileConverter(lineConverter);
    byte[] ba = fc.readFile(file);
    fc.convert(file, ba);
  } catch (IOException exc) {
    addException(new ConversionException(exc.toString()));
  }
}
```

modified.java

```
private void convertFile(File file) {
  try {
    InplaceFileConverter fc = new InplaceFileConverter(ruleSet);
    fc.convert(file);
  } catch (IOException exc) {
    addException(new ConversionException(exc.toString()));
  }
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

public static void assertDuration(double currentDuration,
long referenceDuraion, double referenceBIPS)
throws AssertionFailedError {
double ajustedDuration = adjustExpectedDuration(referenceDuraion,
referenceBIPS);
if (currentDuration > ajustedDuration * SLACK_FACTOR) {
throw new AssertionFailedError(currentDuration
+ " exceeded expected "
+ ajustedDuration + " (adjusted), "
+ referenceDuraion + " (raw)");
}
}
	public static void assertDuration(double currentDuration,
	long referenceDuration, double referenceBIPS)
	throws AssertionFailedError {
	double ajustedDuration = adjustExpectedDuration(referenceDuration,
	referenceBIPS);
	if (currentDuration > ajustedDuration * SLACK_FACTOR) {
	throw new AssertionFailedError("current duration "+ currentDuration
	+ " exceeded expected "
	+ ajustedDuration + " (adjusted reference), "
	+ referenceDuration + " (raw reference)");
	}
	}

Tool 1

```
...  @@ -1,12 +1,12 @@
1    public static void assertDuration(double currentDuration,
2  -     long referenceDuraion, double referenceBIPS)
2  +     long referenceDuration, double referenceBIPS)
3        throws AssertionFailedError {
4  -   double ajustedDuration = adjustExpectedDuration(referenceDuraion,
4  +   double ajustedDuration = adjustExpectedDuration(referenceDuration,
5          referenceBIPS);
6      if (currentDuration > ajustedDuration * SLACK_FACTOR) {
7  -     throw new AssertionFailedError(currentDuration
7  +     throw new AssertionFailedError("current duration "+ currentDuration
8            + " exceeded expected "
9  -         + ajustedDuration + " (adjusted), "
9  +         + ajustedDuration + " (adjusted reference), "
10 -         + referenceDuraion + " (raw)");
10 +         + referenceDuration + " (raw reference)");
11     }
12   }
```

Tool 2

```
1  public static void assertDuration(double currentDuration,
2      long referenceDuraion, double referenceBIPS)
3      throws AssertionFailedError {
4    double ajustedDuration = adjustExpectedDuration(referenceDuraion,
5        referenceBIPS);
6    if (currentDuration > ajustedDuration * SLACK_FACTOR) {
7      throw new AssertionFailedError(currentDuration
8          + " exceeded expected "
9          + ajustedDuration + " (adjusted), "
10         + referenceDuraion + " (raw)");
11   }
12 }
13
```

```
1  public static void assertDuration(double currentDuration,
2      long referenceDuration, double referenceBIPS)
3      throws AssertionFailedError {
4    double ajustedDuration = adjustExpectedDuration(referenceDuration,
5        referenceBIPS);
6    if (currentDuration > ajustedDuration * SLACK_FACTOR) {
7      throw new AssertionFailedError("current duration "+ currentDuration
8          + " exceeded expected "
9          + ajustedDuration + " (adjusted reference), "
10         + referenceDuration + " (raw reference)");
11   }
12 }
13
```

Tool 3

Original

```
1  public static void assertDuration(double currentDuration,
2      long referenceDuraion, double referenceBIPS)
3      throws AssertionFailedError {
4    double ajustedDuration = adjustExpectedDuration(referenceDuraion,
5        referenceBIPS);
6    if (currentDuration > ajustedDuration * SLACK_FACTOR) {
7      throw new AssertionFailedError( currentDuration
8          + " exceeded expected "
9          + ajustedDuration + " (adjusted), "
10         + referenceDuraion + " (raw)");
11   }
12 }
13
```

Modified

```
1  public static void assertDuration(double currentDuration,
2      long referenceDuration, double referenceBIPS)
3      throws AssertionFailedError {
4    double ajustedDuration = adjustExpectedDuration(referenceDuration,
5        referenceBIPS);
6    if (currentDuration > ajustedDuration * SLACK_FACTOR) {
7      throw new AssertionFailedError("current duration "+ currentDuration
8          + " exceeded expected "
9          + ajustedDuration + " (adjusted reference), "
10         + referenceDuration + " (raw reference)");
11   }
12 }
13
```

Tool 4

Legend | Shortcuts

original.java

```
public static void assertDuration(double currentDuration,
    long referenceDuraion, double referenceBIPS)
    throws AssertionFailedError {
  double ajustedDuration = adjustExpectedDuration(referenceDuraion,
      referenceBIPS);
  if (currentDuration > ajustedDuration * SLACK_FACTOR) {
    throw new AssertionFailedError(currentDuration
        + " exceeded expected "
        + ajustedDuration + " (adjusted), "
        + referenceDuraion + " (raw)");
  }
}
```

modified.java

```
public static void assertDuration(double currentDuration,
    long referenceDuration, double referenceBIPS)
    throws AssertionFailedError {
  double ajustedDuration = adjustExpectedDuration(referenceDuration,
      referenceBIPS);
  if (currentDuration > ajustedDuration * SLACK_FACTOR) {
    throw new AssertionFailedError("current duration "+ currentDuration
        + " exceeded expected "
        + ajustedDuration + " (adjusted reference), "
        + referenceDuration + " (raw reference)");
  }
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

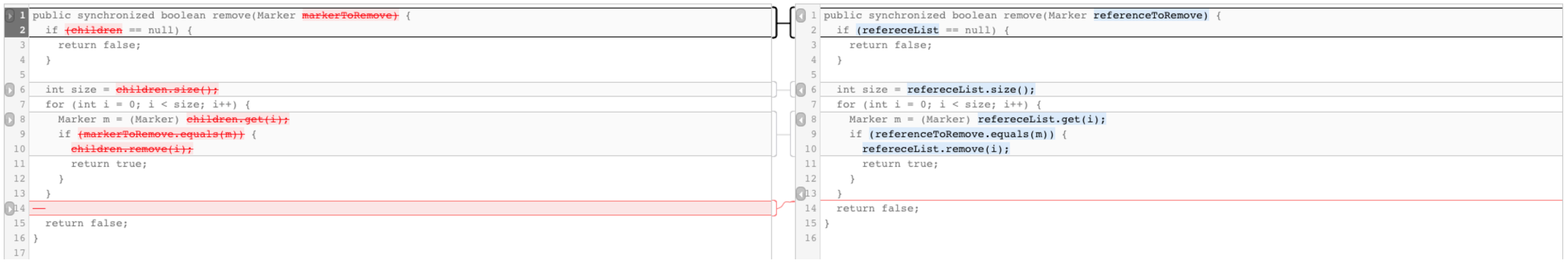
public synchronized boolean remove(Marker markerToRemove) {
  if (children == null) {
  return false;
  }
 	
  int size = children.size();
  for (int i = 0; i < size; i++) {
  Marker m = (Marker) children.get(i);
  if (markerToRemove.equals(m)) {
  children.remove(i);
  return true;
  }
  }
 	
  return false;
}
	public synchronized boolean remove(Marker referenceToRemove) {
	  if (refereceList == null) {
	  return false;
	  }
	
	  int size = refereceList.size();
	  for (int i = 0; i < size; i++) {
	  Marker m = (Marker) refereceList.get(i);
	  if (referenceToRemove.equals(m)) {
	  refereceList.remove(i);
	  return true;
	  }
	  }
	  return false;
	}

Tool 1

```
...  @@ -1,16 +1,15 @@
1   -public synchronized boolean remove(Marker markerToRemove) {
2   -  if (children == null) {
3          return false;
4        }
5
6   -  int size = children.size();
7        for (int i = 0; i < size; i++) {
8   -      Marker m = (Marker) children.get(i);
9   -      if (markerToRemove.equals(m)) {
10  -        children.remove(i);
11             return true;
12       }
13     }
14  -
15     return false;
16   }
```

```
1   +public synchronized boolean remove(Marker referenceToRemove) {
2   +  if (refereceList == null) {
3          return false;
4        }
5
6   +  int size = refereceList.size();
7        for (int i = 0; i < size; i++) {
8   +      Marker m = (Marker) refereceList.get(i);
9   +      if (referenceToRemove.equals(m)) {
10  +        refereceList.remove(i);
11             return true;
12       }
13     }
14
15     return false;
16   }
```

Tool 2



Tool 3

Original

```
 1 public synchronized boolean remove(Marker markerToRemove) {
 2   if (children == null) {
 3     return false;
 4   }
 5
 6   int size = children.size();
 7   for (int i = 0; i < size; i++) {
 8     Marker m = (Marker) children.get(i);
 9     if (markerToRemove.equals(m)) {
10       children.remove(i);
11       return true;
12     }
13   }
14
15   return false;
16 }
17
```

Modified

```
 1 public synchronized boolean remove(Marker referenceToRemove) {
 2   if (refereceList == null) {
 3     return false;
 4   }
 5
 6   int size = refereceList.size();
 7   for (int i = 0; i < size; i++) {
 8     Marker m = (Marker) refereceList.get(i);
 9     if (referenceToRemove.equals(m)) {
10       refereceList.remove(i);
11       return true;
12     }
13   }
14 return false;
15 }
16
```

Tool 4

Legend | Shortcuts

original.java

```
public synchronized boolean remove(Marker markerToRemove) {
  if (children == null) {
    return false;
  }

  int size = children.size();
  for (int i = 0; i < size; i++) {
    Marker m = (Marker) children.get(i);
    if (markerToRemove.equals(m)) {
      children.remove(i);
      return true;
    }
  }

  return false;
}
```

modified.java

```
public synchronized boolean remove(Marker referenceToRemove) {
  if (refereceList == null) {
    return false;
  }

  int size = refereceList.size();
  for (int i = 0; i < size; i++) {
    Marker m = (Marker) refereceList.get(i);
    if (referenceToRemove.equals(m)) {
      refereceList.remove(i);
      return true;
    }
  }
  return false;
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0 Points
Tool 2	0 Points
Tool 3	0 Points
Tool 4	0 Points
Total	0 Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0 Points
Tool 2	0 Points
Tool 3	0 Points
Tool 4	0 Points
Total	0 Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0 Points
Tool 2	0 Points
Tool 3	0 Points

Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original

```
static class ExpirationSpec {
  @Nullable
  private final Long expireAfterAccessMillis;
  @Nullable
  private final Long expireAfterWriteMillis;

  private ExpirationSpec(Long expireAfterAccessMillis, Long expireAfterWriteMillis) {
    Preconditions.checkArgument(
        expireAfterAccessMillis == null || expireAfterWriteMillis == null);
    this.expireAfterAccessMillis = expireAfterAccessMillis;
    this.expireAfterWriteMillis = expireAfterWriteMillis;
  }

  public static ExpirationSpec afterAccess(long afterAccess, TimeUnit unit) {
    return new ExpirationSpec(unit.toMillis(afterAccess), null);
  }

  public static ExpirationSpec afterWrite(long afterWrite, TimeUnit unit) {
    return new ExpirationSpec(null, unit.toMillis(afterWrite));
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(expireAfterAccessMillis, expireAfterWriteMillis);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof ExpirationSpec) {
      ExpirationSpec that = (ExpirationSpec) o;
      return Objects.equal(this.expireAfterAccessMillis, that.expireAfterAccessMillis)
          && Objects.equal(this.expireAfterWriteMillis, that.expireAfterWriteMillis);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("expireAfterAccessMillis", expireAfterAccessMillis)
        .add("expireAfterWriteMillis", expireAfterWriteMillis)
        .toString();
  }
}
```

Modified

```
static class DurationSpec {
  private final long duration;
  private final TimeUnit unit;

  private DurationSpec(long duration, TimeUnit unit) {
    this.duration = duration;
    this.unit = unit;
  }

  public static DurationSpec of(long duration, TimeUnit unit) {
    return new DurationSpec(duration, unit);
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(duration, unit);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof DurationSpec) {
      DurationSpec that = (DurationSpec) o;
      return unit.toNanos(duration) == that.unit.toNanos(that.duration);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("duration", duration)
        .add("unit", unit)
        .toString();
  }
}
```

Tool 1

```
@@ -1,35 +1,26 @@
-static class ExpirationSpec {
-  @Nullable
-  private final Long expireAfterAccessMillis;
-  @Nullable
-  private final Long expireAfterWriteMillis;

-  private ExpirationSpec(Long expireAfterAccessMillis, Long expireAfterWriteMillis) {
-    Preconditions.checkArgument(
-        expireAfterAccessMillis == null || expireAfterWriteMillis == null);
-    this.expireAfterAccessMillis = expireAfterAccessMillis;
-    this.expireAfterWriteMillis = expireAfterWriteMillis;
   }

-  public static ExpirationSpec afterAccess(long afterAccess, TimeUnit unit) {
-    return new ExpirationSpec(unit.toMillis(afterAccess), null);
-  }
-
-  public static ExpirationSpec afterWrite(long afterWrite, TimeUnit unit) {
-    return new ExpirationSpec(null, unit.toMillis(afterWrite));
   }

   @Override
   public int hashCode() {
-    return Objects.hashCode(expireAfterAccessMillis, expireAfterWriteMillis);
   }

   @Override
   public boolean equals(Object o) {
-    if (o instanceof ExpirationSpec) {
-      ExpirationSpec that = (ExpirationSpec) o;
-      return Objects.equal(this.expireAfterAccessMillis, that.expireAfterAccessMillis)
-          && Objects.equal(this.expireAfterWriteMillis, that.expireAfterWriteMillis);
     }
     return false;
   }

   @Override
   public String toString() {
     return Objects.toStringHelper(this)
-        .add("expireAfterAccessMillis", expireAfterAccessMillis)
-        .add("expireAfterWriteMillis", expireAfterWriteMillis)
         .toString();
   }
 }

+static class DurationSpec {
+  private final long duration;
+  private final TimeUnit unit;

+  private DurationSpec(long duration, TimeUnit unit) {
+    this.duration = duration;
+    this.unit = unit;
   }

+  public static DurationSpec of(long duration, TimeUnit unit) {
+    return new DurationSpec(duration, unit);

   }

   @Override
   public int hashCode() {
+    return Objects.hashCode(duration, unit);
   }

   @Override
   public boolean equals(Object o) {
+    if (o instanceof DurationSpec) {
+      DurationSpec that = (DurationSpec) o;
+      return unit.toNanos(duration) == that.unit.toNanos(that.duration);

     }
     return false;
   }

   @Override
   public String toString() {
     return Objects.toStringHelper(this)
+        .add("duration", duration)
+        .add("unit", unit)
         .toString();
   }
 }
```

Tool 2

```
static class ExpirationSpec {
  @Nullable
  private final Long expireAfterAccessMillis;
  @Nullable
  private final Long expireAfterWriteMillis;

  private ExpirationSpec(Long expireAfterAccessMillis, Long expireAfterWriteMillis) {
    Preconditions.checkArgument(
        expireAfterAccessMillis == null || expireAfterWriteMillis == null);
    this.expireAfterAccessMillis = expireAfterAccessMillis;
    this.expireAfterWriteMillis = expireAfterWriteMillis;
  }

  public static ExpirationSpec afterAccess(long afterAccess, TimeUnit unit) {
    return new ExpirationSpec(unit.toMillis(afterAccess), null);
  }

  public static ExpirationSpec afterWrite(long afterWrite, TimeUnit unit) {
    return new ExpirationSpec(null, unit.toMillis(afterWrite));
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(expireAfterAccessMillis, expireAfterWriteMillis);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof ExpirationSpec) {
      ExpirationSpec that = (ExpirationSpec) o;
      return Objects.equal(this.expireAfterAccessMillis, that.expireAfterAccessMillis)
          && Objects.equal(this.expireAfterWriteMillis, that.expireAfterWriteMillis);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("expireAfterAccessMillis", expireAfterAccessMillis)
        .add("expireAfterWriteMillis", expireAfterWriteMillis)
        .toString();
  }
}
```

```
static class DurationSpec {
  private final long duration;
  private final TimeUnit unit;

  private DurationSpec(long duration, TimeUnit unit) {
    this.duration = duration;
    this.unit = unit;
  }

  public static DurationSpec of(long duration, TimeUnit unit) {
    return new DurationSpec(duration, unit);
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(duration, unit);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof DurationSpec) {
      DurationSpec that = (DurationSpec) o;
      return unit.toNanos(duration) == that.unit.toNanos(that.duration);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("duration", duration)
        .add("unit", unit)
        .toString();
  }
}
```

Tool 3

Original

```
static class ExpirationSpec {
  @Nullable
  private final Long expireAfterAccessMillis;
  @Nullable
  private final Long expireAfterWriteMillis;

  private ExpirationSpec(Long expireAfterAccessMillis, Long expireAfterWriteMillis) {
    Preconditions.checkArgument(
        expireAfterAccessMillis == null || expireAfterWriteMillis == null);
    this.expireAfterAccessMillis = expireAfterAccessMillis;
    this.expireAfterWriteMillis = expireAfterWriteMillis;
  }

  public static ExpirationSpec afterAccess(long afterAccess, TimeUnit unit) {
    return new ExpirationSpec(unit.toMillis(afterAccess), null);
  }

  public static ExpirationSpec afterWrite(long afterWrite, TimeUnit unit) {
    return new ExpirationSpec(null, unit.toMillis(afterWrite));
  }









  @Override
  public int hashCode() {
    return Objects.hashCode(expireAfterAccessMillis, expireAfterWriteMillis);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof ExpirationSpec) {
      ExpirationSpec that = (ExpirationSpec) o;
      return Objects.equal(this.expireAfterAccessMillis, that.expireAfterAccessMillis)
          && Objects.equal(this.expireAfterWriteMillis, that.expireAfterWriteMillis);
    }


    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("expireAfterAccessMillis", expireAfterAccessMillis)
        .add("expireAfterWriteMillis", expireAfterWriteMillis)
        .toString();
  }
}

```

Modified

```
static class DurationSpec {

  private final long duration;
  private final TimeUnit unit;

  private DurationSpec(long duration, TimeUnit unit) {
    this.duration = duration;
    this.unit = unit;
  }

  public static DurationSpec of(long duration, TimeUnit unit) {
    return new DurationSpec(duration, unit);
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(duration, unit);
  }

  @Override
  public boolean equals(Object o) {




    if (o instanceof DurationSpec) {
      DurationSpec that = (DurationSpec) o;
      return unit.toNanos(duration) == that.unit.toNanos(that.duration);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("duration", duration)
        .add("unit", unit)
        .toString();
  }
}

```

Click to view **Tool 3** in a new tab/window

Tool 4

Legend | Shortcuts

original.java

```
static class ExpirationSpec {
  @Nullable
  private final Long expireAfterAccessMillis;
  @Nullable
  private final Long expireAfterWriteMillis;

  private ExpirationSpec(Long expireAfterAccessMillis, Long expireAfterWriteMillis) {
    Preconditions.checkArgument(
        expireAfterAccessMillis == null || expireAfterWriteMillis == null);
    this.expireAfterAccessMillis = expireAfterAccessMillis;
    this.expireAfterWriteMillis = expireAfterWriteMillis;
  }

  public static ExpirationSpec afterAccess(long afterAccess, TimeUnit unit) {
    return new ExpirationSpec(unit.toMillis(afterAccess), null);
  }

  public static ExpirationSpec afterWrite(long afterWrite, TimeUnit unit) {
    return new ExpirationSpec(null, unit.toMillis(afterWrite));
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(expireAfterAccessMillis, expireAfterWriteMillis);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof ExpirationSpec) {
      ExpirationSpec that = (ExpirationSpec) o;
      return Objects.equal(this.expireAfterAccessMillis, that.expireAfterAccessMillis)
          && Objects.equal(this.expireAfterWriteMillis, that.expireAfterWriteMillis);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("expireAfterAccessMillis", expireAfterAccessMillis)
        .add("expireAfterWriteMillis", expireAfterWriteMillis)
        .toString();
  }
}
```

modified.java

```
static class DurationSpec {
  private final long duration;
  private final TimeUnit unit;

  private DurationSpec(long duration, TimeUnit unit) {
    this.duration = duration;
    this.unit = unit;
  }

  public static DurationSpec of(long duration, TimeUnit unit) {
    return new DurationSpec(duration, unit);
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(duration, unit);
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof DurationSpec) {
      DurationSpec that = (DurationSpec) o;
      return unit.toNanos(duration) == that.unit.toNanos(that.duration);
    }
    return false;
  }

  @Override
  public String toString() {
    return Objects.toStringHelper(this)
        .add("duration", duration)
        .add("unit", unit)
        .toString();
  }
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1 0 Points

Tool 2 0 Points

Tool 3 0 Points

Tool 4 0 Points

Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  showLineRange(0,buffer.getLineCount() - 1);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < buffer.getLineCount(); i++)
  {
    if (leastFolded == -1 || buffer.getFoldLevel(i) < leastFolded)
    {
      leastFolded = buffer.getFoldLevel(i);
    }

    if (buffer.getFoldLevel(i) < foldLevel ||
        buffer.getFoldLevel(i) == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != buffer.getLineCount())
    hideLineRange(firstInvisible,buffer.getLineCount() - 1);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

Modified

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  int lineCount = buffer.getLineCount();
  int end = lineCount - 1;
  showLineRange(0,end);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < lineCount; i++)
  {
    int level = buffer.getFoldLevel(i);
    if (leastFolded == -1 || level < leastFolded)
    {
      leastFolded = level;
    }

    if (level < foldLevel || level == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != lineCount)
    hideLineRange(firstInvisible,end);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

Tool 1

```
...  @@ -1,22 +1,24 @@
 1   public void expandFolds(int foldLevel, boolean update)
 2   {
 3     if(buffer.getFoldHandler() instanceof IndentFoldHandler)
 4       foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;
 5
 6   -   showLineRange(0,buffer.getLineCount() - 1);
 6   +   int lineCount = buffer.getLineCount();
 7   +   int end = lineCount - 1;
 8   +   showLineRange(0,end);
 7
 8     int leastFolded = -1;
 9     int firstInvisible = 0;
10
11   -   for(int i = 0; i < buffer.getLineCount(); i++)
13   +   for(int i = 0; i < lineCount; i++)
12     {
13   -     if (leastFolded == -1 || buffer.getFoldLevel(i) < leastFolded)
15   +     int level = buffer.getFoldLevel(i);
16   +     if (leastFolded == -1 || level < leastFolded)
14       {
15   -       leastFolded = buffer.getFoldLevel(i);
18   +       leastFolded = level;
16       }
17
18   -     if (buffer.getFoldLevel(i) < foldLevel ||
19   -         buffer.getFoldLevel(i) == leastFolded)
21   +     if (level < foldLevel || level == leastFolded)
20       {
21         if(firstInvisible != i)
22         {
23           hideLineRange(firstInvisible,
24             i - 1);
25         }
26         firstInvisible = i + 1;
27       }
28     }
29
30   - if(firstInvisible != buffer.getLineCount())
31   -   hideLineRange(firstInvisible,buffer.getLineCount() - 1);
32   + if(firstInvisible != lineCount)
33   +   hideLineRange(firstInvisible,end);
32
33     notifyScreenLineChanges();
34     if(update && textArea.getDisplayManager() == this)
35     {
36       textArea.foldStructureChanged();
37     }
38   }
```

Tool 2

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  showLineRange(0,buffer.getLineCount() - 1);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < buffer.getLineCount(); i++)
  {
    if (leastFolded == -1 || buffer.getFoldLevel(i) < leastFolded)
    {
      leastFolded = buffer.getFoldLevel(i);
    }

    if (buffer.getFoldLevel(i) < foldLevel ||
        buffer.getFoldLevel(i) == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != buffer.getLineCount())
    hideLineRange(firstInvisible,buffer.getLineCount() - 1);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  int lineCount = buffer.getLineCount();
  int end = lineCount - 1;
  showLineRange(0,end);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < lineCount; i++)
  {
    int level = buffer.getFoldLevel(i);
    if (leastFolded == -1 || level < leastFolded)
    {
      leastFolded = level;
    }

    if (level < foldLevel || level == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != lineCount)
    hideLineRange(firstInvisible,end);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

Tool 3

Original

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;


  showLineRange(0,buffer.getLineCount() - 1);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < buffer.getLineCount(); i++)
  {

    if (leastFolded == -1 || buffer.getFoldLevel(i) < leastFolded)
    {
      leastFolded = buffer.getFoldLevel(i);
    }

    if (buffer.getFoldLevel(i) < foldLevel ||
        buffer.getFoldLevel(i) == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != buffer.getLineCount())
    hideLineRange(firstInvisible,buffer.getLineCount() - 1);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

Modified

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  int lineCount = buffer.getLineCount();
  int end = lineCount - 1;
  showLineRange(0,end);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < lineCount; i++)
  {
    int level = buffer.getFoldLevel(i);
    if (leastFolded == -1 || level < leastFolded)
    {
      leastFolded = level;
    }

    if (level < foldLevel ||
level == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != lineCount)
    hideLineRange(firstInvisible,end);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

Click to view **Tool 3** in a new tab/window

Tool 4

Legend | Shortcuts

original.java

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  showLineRange(0,buffer.getLineCount() - 1);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < buffer.getLineCount(); i++)
  {
    if (leastFolded == -1 || buffer.getFoldLevel(i) < leastFolded)
    {
      leastFolded = buffer.getFoldLevel(i);
    }

    if (buffer.getFoldLevel(i) < foldLevel ||
        buffer.getFoldLevel(i) == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != buffer.getLineCount())
    hideLineRange(firstInvisible,buffer.getLineCount() - 1);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

modified.java

```
public void expandFolds(int foldLevel, boolean update)
{
  if(buffer.getFoldHandler() instanceof IndentFoldHandler)
    foldLevel = (foldLevel - 1) * buffer.getIndentSize() + 1;

  int lineCount = buffer.getLineCount();
  int end = lineCount - 1;
  showLineRange(0,end);

  int leastFolded = -1;
  int firstInvisible = 0;

  for(int i = 0; i < lineCount; i++)
  {
    int level = buffer.getFoldLevel(i);
    if (leastFolded == -1 || level < leastFolded)
    {
      leastFolded = level;
    }

    if (level < foldLevel || level == leastFolded)
    {
      if(firstInvisible != i)
      {
        hideLineRange(firstInvisible,
          i - 1);
      }
      firstInvisible = i + 1;
    }
  }

  if(firstInvisible != lineCount)
    hideLineRange(firstInvisible,end);

  notifyScreenLineChanges();
  if(update && textArea.getDisplayManager() == this)
  {
    textArea.foldStructureChanged();
  }
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

void physDown(int amount, int screenAmount)
{
  skew = 0;
 	
  if(!isLineVisible(physicalLine))
  {
  int lastVisibleLine = getLastVisibleLine();
  if(physicalLine > lastVisibleLine)
  physicalLine = lastVisibleLine;
  else
  {
  int nextPhysicalLine = getNextVisibleLine(physicalLine);
  amount -= (nextPhysicalLine - physicalLine);
  scrollLine += getScreenLineCount(physicalLine);
  physicalLine = nextPhysicalLine;
  }
  }
 	
  for(;;)
  {
  int nextPhysicalLine = getNextVisibleLine(
  physicalLine);
  if(nextPhysicalLine == -1)
  break;
  else if(nextPhysicalLine > physicalLine + amount)
  break;
  else
  {
  scrollLine += getScreenLineCount(physicalLine);
  amount -= (nextPhysicalLine - physicalLine);
  physicalLine = nextPhysicalLine;
  }
  }
 	
  if(screenAmount != 0)
  {
  if(screenAmount < 0)
  scrollUp(-screenAmount);
  else
  scrollDown(screenAmount);
  }
}
	void physDown(int amount, int screenAmount)
	{
	  skew = 0;
	
	  if(!isLineVisible(physicalLine))
	  {
	  int lastVisibleLine = getLastVisibleLine();
	  if(physicalLine > lastVisibleLine)
	  physicalLine = lastVisibleLine;
	  else
	  {
	  int nextPhysicalLine = getNextVisibleLine(physicalLine);
	  amount -= (nextPhysicalLine - physicalLine);
	  scrollLine += getScreenLineCount(physicalLine);
	  physicalLine = nextPhysicalLine;
	  }
	  }
	
	  for(;;)
	  {
	  int nextPhysicalLine = getNextVisibleLine(
	  physicalLine);
	  if(nextPhysicalLine == -1)
	  break;
	  else if(nextPhysicalLine > physicalLine + amount)
	  break;
	  else
	  {
	  scrollLine += getScreenLineCount(physicalLine);
	  amount -= (nextPhysicalLine - physicalLine);
	  physicalLine = nextPhysicalLine;
	  }
	  }
	
	  if(screenAmount < 0)
	  scrollUp(-screenAmount);
	  else if(screenAmount > 0)
	  scrollDown(screenAmount);
	}

Tool 1

```
...  @@ -1,42 +1,39 @@
 1   void physDown(int amount, int screenAmount)
 2   {
 3     skew = 0;
 4
 5     if(!isLineVisible(physicalLine))
 6     {
 7       int lastVisibleLine = getLastVisibleLine();
 8       if(physicalLine > lastVisibleLine)
 9         physicalLine = lastVisibleLine;
10       else
11       {
12         int nextPhysicalLine = getNextVisibleLine(physicalLine);
13         amount -= (nextPhysicalLine - physicalLine);
14         scrollLine += getScreenLineCount(physicalLine);
15         physicalLine = nextPhysicalLine;
16       }
17     }
18
19     for(;;)
20     {
21       int nextPhysicalLine = getNextVisibleLine(
22         physicalLine);
23       if(nextPhysicalLine == -1)
24         break;
25       else if(nextPhysicalLine > physicalLine + amount)
26         break;
27       else
28       {
29         scrollLine += getScreenLineCount(physicalLine);
30         amount -= (nextPhysicalLine - physicalLine);
31         physicalLine = nextPhysicalLine;
32       }
33     }
34
35 -   if(screenAmount != 0)
36 -   {
37     if(screenAmount < 0)
38       scrollUp(-screenAmount);
39 -     else
40       scrollDown(screenAmount);
41   }
42 -}
```

```
 1   void physDown(int amount, int screenAmount)
 2   {
 3     skew = 0;
 4
 5     if(!isLineVisible(physicalLine))
 6     {
 7       int lastVisibleLine = getLastVisibleLine();
 8       if(physicalLine > lastVisibleLine)
 9         physicalLine = lastVisibleLine;
10       else
11       {
12         int nextPhysicalLine = getNextVisibleLine(physicalLine);
13         amount -= (nextPhysicalLine - physicalLine);
14         scrollLine += getScreenLineCount(physicalLine);
15         physicalLine = nextPhysicalLine;
16       }
17     }
18
19     for(;;)
20     {
21       int nextPhysicalLine = getNextVisibleLine(
22         physicalLine);
23       if(nextPhysicalLine == -1)
24         break;
25       else if(nextPhysicalLine > physicalLine + amount)
26         break;
27       else
28       {
29         scrollLine += getScreenLineCount(physicalLine);
30         amount -= (nextPhysicalLine - physicalLine);
31         physicalLine = nextPhysicalLine;
32       }
33     }
34
35     if(screenAmount < 0)
36       scrollUp(-screenAmount);
37 +   else if(screenAmount > 0)
38       scrollDown(screenAmount);
39   }
```

Tool 2

```
 1  void physDown(int amount, int screenAmount)
 2  {
 3    skew = 0;
 4
 5    if(!isLineVisible(physicalLine))
 6    {
 7      int lastVisibleLine = getLastVisibleLine();
 8      if(physicalLine > lastVisibleLine)
 9        physicalLine = lastVisibleLine;
10      else
11      {
12        int nextPhysicalLine = getNextVisibleLine(physicalLine);
13        amount -= (nextPhysicalLine - physicalLine);
14        scrollLine += getScreenLineCount(physicalLine);
15        physicalLine = nextPhysicalLine;
16      }
17    }
18
19    for(;;)
20    {
21      int nextPhysicalLine = getNextVisibleLine(
22        physicalLine);
23      if(nextPhysicalLine == -1)
24        break;
25      else if(nextPhysicalLine > physicalLine + amount)
26        break;
27      else
28      {
29        scrollLine += getScreenLineCount(physicalLine);
30        amount -= (nextPhysicalLine - physicalLine);
31        physicalLine = nextPhysicalLine;
32      }
33    }
34
35    if(screenAmount < 0)
36      scrollUp(-screenAmount);
37    else if(screenAmount > 0)
38      scrollDown(screenAmount);
39  }
40
```

Tool 3

Original

```
 1  void physDown(int amount, int screenAmount)
 2  {
 3    skew = 0;
 4
 5    if(!isLineVisible(physicalLine))
 6    {
 7      int lastVisibleLine = getLastVisibleLine();
 8      if(physicalLine > lastVisibleLine)
 9        physicalLine = lastVisibleLine;
10      else
11      {
```

Modified

```
 1  void physDown(int amount, int screenAmount)
 2  {
 3    skew = 0;
 4
 5    if(!isLineVisible(physicalLine))
 6    {
 7      int lastVisibleLine = getLastVisibleLine();
 8      if(physicalLine > lastVisibleLine)
 9        physicalLine = lastVisibleLine;
10      else
11      {
```

```
12        int nextPhysicalLine = getNextVisibleLine(physicalLine);
13        amount -= (nextPhysicalLine - physicalLine);
14        scrollLine += getScreenLineCount(physicalLine);
15        physicalLine = nextPhysicalLine;
16      }
17    }
18
19    for(;;)
20    {
21      int nextPhysicalLine = getNextVisibleLine(
22        physicalLine);
23      if(nextPhysicalLine == -1)
24        break;
25      else if(nextPhysicalLine > physicalLine + amount)
26        break;
27      else
28      {
29        scrollLine += getScreenLineCount(physicalLine);
30        amount -= (nextPhysicalLine - physicalLine);
31        physicalLine = nextPhysicalLine;
32      }
33    }
34
35    if(screenAmount != 0)
36    {
37      if(screenAmount < 0)
38        scrollUp(-screenAmount);
39      else
40        scrollDown(screenAmount);
41    }
42  }
43
```

```
12        int nextPhysicalLine = getNextVisibleLine(physicalLine);
13        amount -= (nextPhysicalLine - physicalLine);
14        scrollLine += getScreenLineCount(physicalLine);
15        physicalLine = nextPhysicalLine;
16      }
17    }
18
19    for(;;)
20    {
21      int nextPhysicalLine = getNextVisibleLine(
22        physicalLine);
23      if(nextPhysicalLine == -1)
24        break;
25      else if(nextPhysicalLine > physicalLine + amount)
26        break;
27      else
28      {
29        scrollLine += getScreenLineCount(physicalLine);
30        amount -= (nextPhysicalLine - physicalLine);
31        physicalLine = nextPhysicalLine;
32      }
33    }
34
35    if(screenAmount < 0)
36      scrollUp(-screenAmount);
37    else if(screenAmount > 0)
38      scrollDown(screenAmount);
39  }
40
```

Tool 4

Legend | Shortcuts

original.java

```
void physDown(int amount, int screenAmount)
{
  skew = 0;

  if(!isLineVisible(physicalLine))
  {
    int lastVisibleLine = getLastVisibleLine();
    if(physicalLine > lastVisibleLine)
      physicalLine = lastVisibleLine;
    else
    {
      int nextPhysicalLine = getNextVisibleLine(physicalLine);
      amount -= (nextPhysicalLine - physicalLine);
      scrollLine += getScreenLineCount(physicalLine);
      physicalLine = nextPhysicalLine;
    }
  }

  for(;;)
  {
    int nextPhysicalLine = getNextVisibleLine(
      physicalLine);
    if(nextPhysicalLine == -1)
      break;
    else if(nextPhysicalLine > physicalLine + amount)
      break;
    else
    {
      scrollLine += getScreenLineCount(physicalLine);
      amount -= (nextPhysicalLine - physicalLine);
      physicalLine = nextPhysicalLine;
    }
  }

  if(screenAmount != 0)
  {
    if(screenAmount < 0)
      scrollUp(-screenAmount);
    else
      scrollDown(screenAmount);
  }
}
```

modified.java

```
void physDown(int amount, int screenAmount)
{
  skew = 0;

  if(!isLineVisible(physicalLine))
  {
    int lastVisibleLine = getLastVisibleLine();
    if(physicalLine > lastVisibleLine)
      physicalLine = lastVisibleLine;
    else
    {
      int nextPhysicalLine = getNextVisibleLine(physicalLine);
      amount -= (nextPhysicalLine - physicalLine);
      scrollLine += getScreenLineCount(physicalLine);
      physicalLine = nextPhysicalLine;
    }
  }

  for(;;)
  {
    int nextPhysicalLine = getNextVisibleLine(
      physicalLine);
    if(nextPhysicalLine == -1)
      break;
    else if(nextPhysicalLine > physicalLine + amount)
      break;
    else
    {
      scrollLine += getScreenLineCount(physicalLine);
      amount -= (nextPhysicalLine - physicalLine);
      physicalLine = nextPhysicalLine;
    }
  }

  if(screenAmount < 0)
    scrollUp(-screenAmount);
  else if(screenAmount > 0)
    scrollDown(screenAmount);
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

`@Beta`
`@Nullable`
`@CheckForNull`
`public static Long tryParse(String string) {`
`  if (checkNotNull(string).isEmpty()) {`
`  return null;`
`  }`
`  boolean negative = string.charAt(0) == '-';`
`  int index = negative ? 1 : 0;`
`  if (index == string.length()) {`
`  return null;`
`  }`
`  int digit = string.charAt(index++) - '0';`
`  if (digit < 0 || digit > 9) {`
`  return null;`
`  }`
`  long accum = -digit;`
`  while (index < string.length()) {`
`  digit = string.charAt(index++) - '0';`
`  if (digit < 0 || digit > 9 || accum < Long.MIN_VALUE / 10) {`
`  return null;`
`  }`
`  accum *= 10;`
`  if (accum < Long.MIN_VALUE + digit) {`
`  return null;`
`  }`
`  accum -= digit;`
`  }`
 	
`  if (negative) {`
`  return accum;`
`  } else if (accum == Long.MIN_VALUE) {`
`  return null;`
`  } else {`
`  return -accum;`
`  }`
`}`
	`@Beta`
	`@Nullable`
	`@CheckForNull`
	`public static Long tryParse(String string, int radix) {`
	`  if (checkNotNull(string).isEmpty()) {`
	`  return null;`
	`  }`
	`  if (radix < Character.MIN_RADIX || radix > Character.MAX_RADIX) {`
	`  throw new IllegalArgumentException(`
	`  "radix must be between MIN_RADIX and MAX_RADIX but was " + radix);`
	`  }`
	`  boolean negative = string.charAt(0) == '-';`
	`  int index = negative ? 1 : 0;`
	`  if (index == string.length()) {`
	`  return null;`
	`  }`
	`  int digit = digit(string.charAt(index++));`
	`  if (digit < 0 || digit >= radix) {`
	`  return null;`
	`  }`
	`  long accum = -digit;`
	
	`  long cap = Long.MIN_VALUE / radix;`
	
	`  while (index < string.length()) {`
	`  digit = digit(string.charAt(index++));`
	`  if (digit < 0 || digit >= radix || accum < cap) {`
	`  return null;`
	`  }`
	`  accum *= radix;`
	`  if (accum < Long.MIN_VALUE + digit) {`
	`  return null;`
	`  }`
	`  accum -= digit;`
	`  }`
	
	`  if (negative) {`
	`  return accum;`
	`  } else if (accum == Long.MIN_VALUE) {`
	`  return null;`
	`  } else {`
	`  return -accum;`
	`  }`
	`}`

Tool 1

```
...  @@ -1,26 +1,33 @@
 1   @Beta                                                  1   @Beta
 2   @Nullable                                              2   @Nullable
 3   @CheckForNull                                          3   @CheckForNull
 4  -public static Long tryParse(String string) {          4  +public static Long tryParse(String string, int radix) {
 5     if (checkNotNull(string).isEmpty()) {               5     if (checkNotNull(string).isEmpty()) {
 6       return null;                                      6       return null;
 7     }                                                   7     }
                                                           8  +  if (radix < Character.MIN_RADIX || radix > Character.MAX_RADIX) {
                                                           9  +    throw new IllegalArgumentException(
                                                          10  +        "radix must be between MIN_RADIX and MAX_RADIX but was " + radix);
                                                          11  +  }
 8     boolean negative = string.charAt(0) == '-';        12    boolean negative = string.charAt(0) == '-';
 9     int index = negative ? 1 : 0;                      13    int index = negative ? 1 : 0;
10     if (index == string.length()) {                    14    if (index == string.length()) {
11       return null;                                     15      return null;
12     }                                                   16    }
13  -  int digit = string.charAt(index++) - '0';         17  +  int digit = digit(string.charAt(index++));
14  -  if (digit < 0 || digit > 9) {                      18  +  if (digit < 0 || digit >= radix) {
15       return null;                                     19      return null;
16     }                                                   20    }
17     long accum = -digit;                               21    long accum = -digit;
                                                          22  +
                                                          23  +  long cap = Long.MIN_VALUE / radix;
                                                          24  +
18     while (index < string.length()) {                  25    while (index < string.length()) {
19  -    digit = string.charAt(index++) - '0';           26  +    digit = digit(string.charAt(index++));
20  -    if (digit < 0 || digit > 9 || accum < Long.MIN_VALUE / 10) {  27  +    if (digit < 0 || digit >= radix || accum < cap) {
21         return null;                                   28        return null;
22       }                                                29      }
23  -    accum *= 10;                                    30  +    accum *= radix;
24       if (accum < Long.MIN_VALUE + digit) {          31      if (accum < Long.MIN_VALUE + digit) {
25         return null;                                   32        return null;
26       }                                                33      }
27       accum -= digit;                                34      accum -= digit;
28     }                                                  35    }
29                                                        36
30     if (negative) {                                    37    if (negative) {
31       return accum;                                    38      return accum;
32     } else if (accum == Long.MIN_VALUE) {              39    } else if (accum == Long.MIN_VALUE) {
33       return null;                                     40      return null;
34     } else {                                           41    } else {
35       return -accum;                                   42      return -accum;
36     }                                                  43    }
37   }                                                    44  }
```

Tool 2

```
 1  @Beta                                                  1  @Beta
 2  @Nullable                                              2  @Nullable
 3  @CheckForNull                                          3  @CheckForNull
 4  public static Long tryParse(String ~~string~~) {      4  public static Long tryParse(String string, int radix) {
 5    if (checkNotNull(string).isEmpty()) {                5    if (checkNotNull(string).isEmpty()) {
 6      return null;                                       6      return null;
 7    }                                                    7    }
 8    boolean negative = string.charAt(0) == '-';          8    if (radix < Character.MIN_RADIX || radix > Character.MAX_RADIX) {
 9    int index = negative ? 1 : 0;                        9      throw new IllegalArgumentException(
10    if (index == string.length()) {                     10          "radix must be between MIN_RADIX and MAX_RADIX but was " + radix);
11      return null;                                      11    }
12    }                                                   12    boolean negative = string.charAt(0) == '-';
13    int digit = ~~string.charAt(index++) - '0';~~       13    int index = negative ? 1 : 0;
14    if (digit < 0 || digit ~~> 9~~) {                   14    if (index == string.length()) {
15      return null;                                      15      return null;
16    }                                                   16    }
17    long accum = -digit;                                17    int digit = digit(string.charAt(index++));
18    while (index < string.length()) {                   18    if (digit < 0 || digit >= radix) {
19      digit = ~~string.charAt(index++) - '0';~~         19      return null;
20      if (digit < 0 || digit ~~> 9~~ || accum < ~~Long.MIN_VALUE / 10~~) {  20    }
21        return null;                                    21    long accum = -digit;
22      }                                                 22
23      accum *= ~~10;~~                                  23    long cap = Long.MIN_VALUE / radix;
24      if (accum < Long.MIN_VALUE + digit) {             24
25        return null;                                    25    while (index < string.length()) {
26      }                                                 26      digit = digit(string.charAt(index++));
27      accum -= digit;                                   27      if (digit < 0 || digit >= radix || accum < cap) {
28    }                                                   28        return null;
29                                                        29      }
30    if (negative) {                                     30      accum *= radix;
31      return accum;                                     31      if (accum < Long.MIN_VALUE + digit) {
32    } else if (accum == Long.MIN_VALUE) {               32        return null;
33      return null;                                      33      }
34    } else {                                            34      accum -= digit;
35      return -accum;                                    35    }
36    }                                                   36
37  }                                                     37    if (negative) {
38                                                        38      return accum;
                                                          39    } else if (accum == Long.MIN_VALUE) {
                                                          40      return null;
                                                          41    } else {
                                                          42      return -accum;
                                                          43    }
                                                          44  }
                                                          45
```

Tool 3

Original

```
@Beta
@Nullable
@CheckForNull
public static Long tryParse(String string) {
  if (checkNotNull(string).isEmpty()) {
    return null;
  }
  boolean negative = string.charAt(0) == '-';
  int index = negative ? 1 : 0;
  if (index == string.length()) {
    return null;
  }
  int digit = string.charAt(index++) - '0';
  if (digit < 0 || digit > 9) {
    return null;
  }
  long accum = -digit;
  while (index < string.length()) {
    digit = string.charAt(index++) - '0';
    if (digit < 0 || digit > 9 || accum < Long.MIN_VALUE / 10) {
      return null;
    }
    accum *= 10;
    if (accum < Long.MIN_VALUE + digit) {
      return null;
    }
    accum -= digit;
  }

  if (negative) {
    return accum;
  } else if (accum == Long.MIN_VALUE) {
    return null;
  } else {
    return -accum;
  }
}

```

Modified

```
@Beta
@Nullable
@CheckForNull
public static Long tryParse(String string, int radix) {
  if (checkNotNull(string).isEmpty()) {
    return null;
  }
  if (radix < Character.MIN_RADIX || radix > Character.MAX_RADIX) {
    throw new IllegalArgumentException(
        "radix must be between MIN_RADIX and MAX_RADIX but was " + radix);
  }
  boolean negative = string.charAt(0) == '-';
  int index = negative ? 1 : 0;
  if (index == string.length()) {
    return null;
  }
  int digit = digit(string.charAt(index++)) ;
  if (digit < 0 || digit >= radix) {
    return null;
  }
  long accum = -digit;

  long cap = Long.MIN_VALUE / radix;

  while (index < string.length()) {
    digit = digit(string.charAt(index++));
    if (digit < 0 || digit >= radix || accum < cap) {
      return null;
    }
    accum *= radix;
    if (accum < Long.MIN_VALUE + digit) {
      return null;
    }
    accum -= digit;
  }

  if (negative) {
    return accum;
  } else if (accum == Long.MIN_VALUE) {
    return null;
  } else {
    return -accum;
  }
}

```

Click to view **Tool 3** in a new tab/window

Tool 4

Legend | Shortcuts

original.java

```
@Beta
@Nullable
@CheckForNull
public static Long tryParse(String string) {
  if (checkNotNull(string).isEmpty()) {
    return null;
  }
  boolean negative = string.charAt(0) == '-';
  int index = negative ? 1 : 0;
  if (index == string.length()) {
    return null;
  }
  int digit = string.charAt(index++) - '0';
  if (digit < 0 || digit > 9) {
    return null;
  }
  long accum = -digit;
  while (index < string.length()) {
    digit = string.charAt(index++) - '0';
    if (digit < 0 || digit > 9 || accum < Long.MIN_VALUE / 10) {
      return null;
    }
    accum *= 10;
    if (accum < Long.MIN_VALUE + digit) {
      return null;
    }
    accum -= digit;
  }

  if (negative) {
    return accum;
  } else if (accum == Long.MIN_VALUE) {
    return null;
  } else {
    return -accum;
  }
}
```

modified.java

```
@Beta
@Nullable
@CheckForNull
public static Long tryParse(String string, int radix) {
  if (checkNotNull(string).isEmpty()) {
    return null;
  }
  if (radix < Character.MIN_RADIX || radix > Character.MAX_RADIX) {
    throw new IllegalArgumentException(
        "radix must be between MIN_RADIX and MAX_RADIX but was " + radix);
  }
  boolean negative = string.charAt(0) == '-';
  int index = negative ? 1 : 0;
  if (index == string.length()) {
    return null;
  }
  int digit = digit(string.charAt(index++));
  if (digit < 0 || digit >= radix) {
    return null;
  }
  long accum = -digit;

  long cap = Long.MIN_VALUE / radix;

  while (index < string.length()) {
    digit = digit(string.charAt(index++));
    if (digit < 0 || digit >= radix || accum < cap) {
      return null;
    }
    accum *= radix;
    if (accum < Long.MIN_VALUE + digit) {
      return null;
    }
    accum -= digit;
  }

  if (negative) {
    return accum;
  } else if (accum == Long.MIN_VALUE) {
    return null;
  } else {
    return -accum;
  }
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

A moderate A very large

	None at all	A slight amount	amount	A large amount	amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Original and Modified Source Code

Original	Modified

public ImmutableSet<ClassInfo> getClasses(Package pkg) {
  return getClasses(pkg.getName());
}
	public ImmutableSet<ClassInfo> getClassesRecursive(String packageName) {
	  checkNotNull(packageName);
	  String packagePrefix = packageName + '.';
	  ImmutableSet.Builder<ClassInfo> builder = ImmutableSet.builder();
	  for (ClassInfo classInfo : classes) {
	  if (classInfo.getName().startsWith(packagePrefix)) {
	  builder.add(classInfo);
	  }
	  }
	  return builder.build();
	}

Tool 1

```
...  @@ -1,3 +1,11 @@
1    -public ImmutableSet<ClassInfo> getClasses(Package pkg) {
2    -  return getClasses(pkg.getName());
3     }
```

```
1    +public ImmutableSet<ClassInfo> getClassesRecursive(String packageName) {
2    +  checkNotNull(packageName);
3    +  String packagePrefix = packageName + '.';
4    +  ImmutableSet.Builder<ClassInfo> builder = ImmutableSet.builder();
5    +  for (ClassInfo classInfo : classes) {
6    +    if (classInfo.getName().startsWith(packagePrefix)) {
7    +      builder.add(classInfo);
8    +    }
9    +  }
10   +  return builder.build();
11    }
```

Tool 2

```
1  public ImmutableSet<ClassInfo> getClasses(Package pkg) {
2    return getClasses(pkg.getName());
3  }
4
```

```
1  public ImmutableSet<ClassInfo> getClassesRecursive(String packageName) {
2    checkNotNull(packageName);
3    String packagePrefix = packageName + '.';
4    ImmutableSet.Builder<ClassInfo> builder = ImmutableSet.builder();
5    for (ClassInfo classInfo : classes) {
6      if (classInfo.getName().startsWith(packagePrefix)) {
7        builder.add(classInfo);
8      }
9    }
10   return builder.build();
11 }
12
```

Tool 3

Original

```
1  public ImmutableSet<ClassInfo> getClasses(Package pkg) {
2    return getClasses(pkg.getName());
3  }
4
```

Modified

```
1  public ImmutableSet<ClassInfo> getClassesRecursive(String packageName) {
2    checkNotNull(packageName);
3    String packagePrefix = packageName + '.';
4    ImmutableSet.Builder<ClassInfo> builder = ImmutableSet.builder();
5    for (ClassInfo classInfo : classes) {
6      if (classInfo.getName().startsWith(packagePrefix)) {
7        builder.add(classInfo);
8      }
9    }
10   return builder.build();
11 }
12
```

Tool 4

Legend | Shortcuts

original.java

```
public ImmutableSet<ClassInfo> getClasses(Package pkg) {
  return getClasses(pkg.getName());
}
```

modified.java

```
public ImmutableSet<ClassInfo> getClassesRecursive(String packageName) {
  checkNotNull(packageName);
  String packagePrefix = packageName + '.';
  ImmutableSet.Builder<ClassInfo> builder = ImmutableSet.builder();
  for (ClassInfo classInfo : classes) {
    if (classInfo.getName().startsWith(packagePrefix)) {
      builder.add(classInfo);
    }
  }
  return builder.build();
}
```

Click to view **Tool 4** in a new tab/window

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

How much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing the differences between the original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Post Questionnaire

Answer these questions considering all of the problems as a whole.

Answer the following for each of the tools. You may answer in any order.

Overall, how much of the code is marked as unchanged that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Answer the following for each of the tools. You may answer in any order.

Overall, how much of the code is marked as changed that should be marked otherwise?

	None at all	A slight amount	A moderate amount	A large amount	A very large amount
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Tool 3	○	○	○	○	○
Tool 4	○	○	○	○	○

Rank the tools by distributing 100 points among the tools relative to how well each tool does at capturing differences between original and modified code.
(Most points = Best, Least points = Worst, Equal points = Equally well)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how easy each tool's output is to understand.
(Most points = Easiest, Least points = Hardest, Equal points = Equally easy/hard)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the tools by distributing 100 points among the tools relative to how readable each tool's output is.
(Most points = Most readable, Least points = Least readable, Equal points = Equally readable)

Tool 1	0	Points
Tool 2	0	Points
Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Rank the following by distributing 100 points among each relative to your preference.
(Most points = Most preferred, Least points = Least preferred, Equal points = Equally preferred)

Original/Modified	0	Points
Tool 1	0	Points
Tool 2	0	Points

Tool 3	0	Points
Tool 4	0	Points
Total	0	Points

Additional Comments?

>>

Exit Questionnaire

The following questions ask a few background questions pertaining to the study. The data will remain confidential.

Age?

- < 18 years
- 18 - 25 years
- 25 - 40 years
- > 40 years

Gender?

- Male
- Female
- Other

Select the status that most fits your description.

- Undergraduate Student
- Graduate Student
- Faculty
- Industry Practitioner

Years of programming experience in Java.

- < 6 months
- 6 months - 1 year
- 1 - 2 years
- 2 - 5 years
- > 5 years

Years of programming experience in any language.

- < 6 months
- 6 months - 1 year
- 1 - 2 years
- 2 - 5 years
- > 5 years

How would you rate your coding skills at this point?

- Excellent
- Above average/Good
- Average
- Below Average
- Poor

Which programming language did you first learn?

List programming languages you have experience with.

Write none if you cannot program in any language.

Which OS are you proficient in using?

- Mac
- Windows
- Linux
- Other

How often do you use differencing tools?

For example: GNU diff

- Regularly
- Frequently
- Occasionally
- Rarely
- Never

How would you rate your skills using differencing tools at this point?

For example: GNU diff

- Excellent
- Above average/Good
- Average
- Below Average
- Poor

How often do you use version control systems?

For example: CVS, Subversion/svn, Git, Perforce, Team Foundation (MS)

- Regularly
- Frequently
- Occasionally
- Rarely
- Never

How would you rate your skills using a version control system at this point?

For example: CVS, Subversion/svn, Git, Perforce, Team Foundation (MS)

- Excellent
- Above average/Good
- Average
- Below Average
- Poor

Are you Red, Green, or Blue color bind?

Select all types of color blindness that apply

- Yes
- Red
- Green
- Blue
- No
- Do not know

What browser are you using?

- Firefox
- Google Chrome
- Internet Explorer
- Safari
- Other

>>

Finished

Thank you for completing the survey.

If you are completing this for a class, record this number: 5349

Please click ">>" to finish.

>>