Re-show introduction | ❚❚ Pause

Commit description

Entire word search rewritten: Now the search is done as usual, then before returning the result it is checked to be a whole word, if not, the next result is searched (and checked). A whole word has the following conditions:

start of word :

- beginning of the line
- starts with a symbol or space
- char before the first char is a symbol or space

end of word:

- end of line
- ends with a symbol or space
- char next to the last char is a symbol or space

Code changes

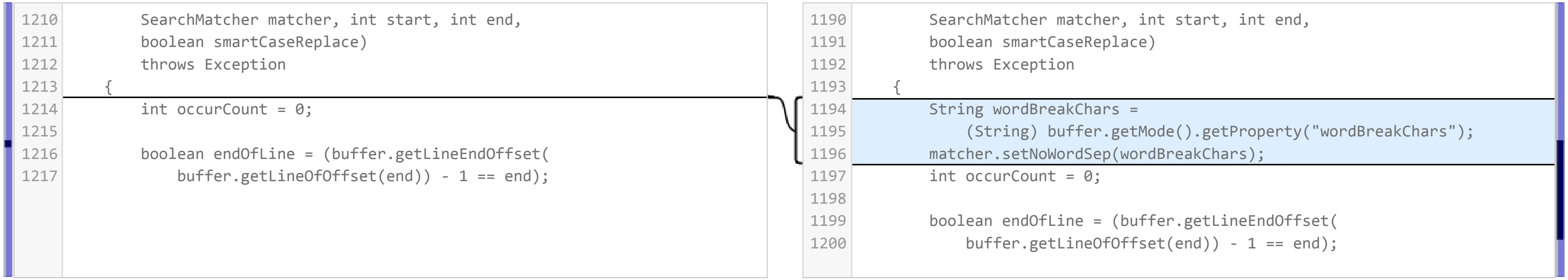
Below you find the code changes to review. The old version of the code is on the left, the new version is on the right.

To add a review remark, click on the respective line number. To delete it, click on it again and delete the remark's text. If a defect spans multiple lines, just mark one of those lines. If similar defects appear multiple times, please mark every occurrence. If you suspect something could be a defect but are not 100% sure, it's better to add a review remark.

At several of the change parts, you can show the whole changed method by clicking on "(Show more context)".

org/experiment/editor/search/SearchAndReplace.java, _replace()

org/experiment/editor/search/SearchAndReplace.java, _replace() (Show more context)

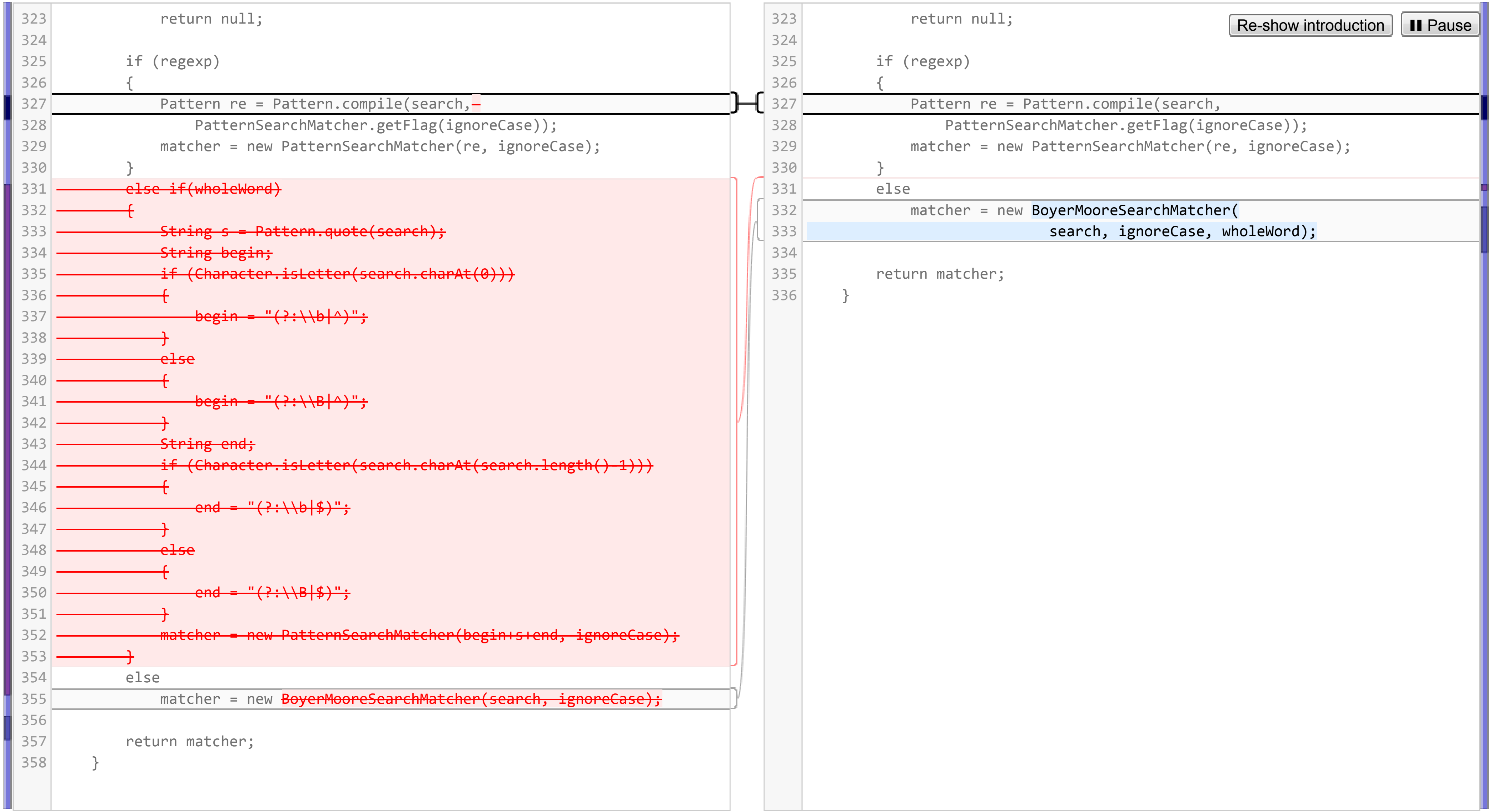


```
        SearchMatcher matcher, int start, int end,
        boolean smartCaseReplace)
        throws Exception
    {
        int occurCount = 0;

        boolean endOfLine = (buffer.getLineEndOffset(
            buffer.getLineOfOffset(end)) - 1 == end);
```

```
        SearchMatcher matcher, int start, int end,
        boolean smartCaseReplace)
        throws Exception
    {
        String wordBreakChars =
            (String) buffer.getMode().getProperty("wordBreakChars");
        matcher.setNoWordSep(wordBreakChars);
        int occurCount = 0;

        boolean endOfLine = (buffer.getLineEndOffset(
            buffer.getLineOfOffset(end)) - 1 == end);
```

org/experiment/editor/search/SearchAndReplace.java, getSearchMatcher()

org/experiment/editor/search/SearchAndReplace.java, getSearchMatcher() (Show more context)

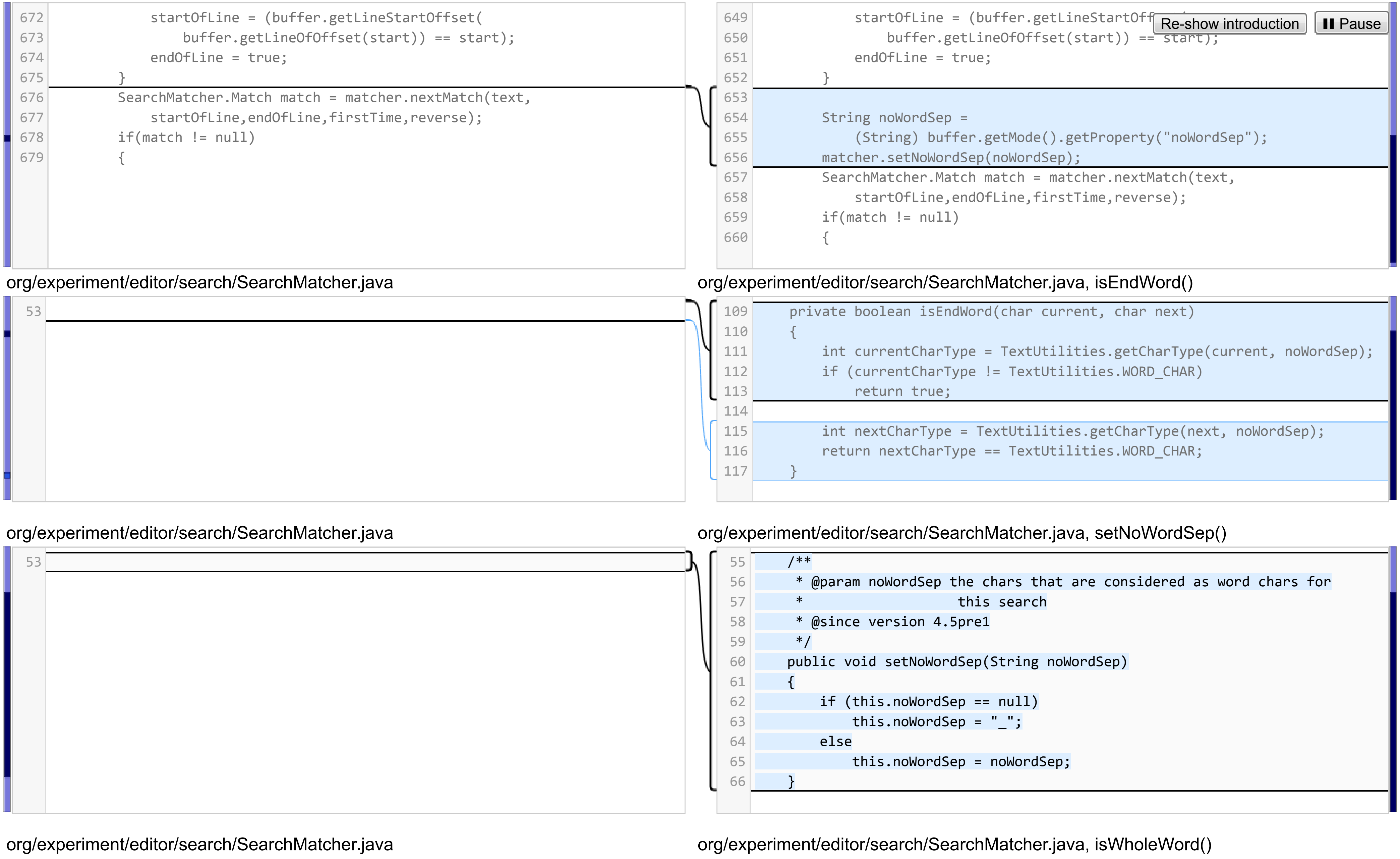


```
            return null;

        if (regexp)
        {
            Pattern re = Pattern.compile(search,
                PatternSearchMatcher.getFlag(ignoreCase));
            matcher = new PatternSearchMatcher(re, ignoreCase);
        }
        ~~else if(wholeWord)~~
        ~~{~~
            ~~String s = Pattern.quote(search);~~
            ~~String begin;~~
            ~~if (Character.isLetter(search.charAt(0)))~~
            ~~{~~
                ~~begin = "(?:\\b|^)";~~
            ~~}~~
            ~~else~~
            ~~{~~
                ~~begin = "(?:\\B|^)";~~
            ~~}~~
            ~~String end;~~
            ~~if (Character.isLetter(search.charAt(search.length()-1)))~~
            ~~{~~
                ~~end = "(?:\\b|$)";~~
            ~~}~~
            ~~else~~
            ~~{~~
                ~~end = "(?:\\B|$)";~~
            ~~}~~
            ~~matcher = new PatternSearchMatcher(begin+s+end, ignoreCase);~~
        ~~}~~
        else
            matcher = new ~~BoyerMooreSearchMatcher(search, ignoreCase);~~

        return matcher;
    }
```

org/experiment/editor/search/SearchAndReplace.java, find()

```
            return null;

        if (regexp)
        {
            Pattern re = Pattern.compile(search,
                PatternSearchMatcher.getFlag(ignoreCase));
            matcher = new PatternSearchMatcher(re, ignoreCase);
        }
        else
            matcher = new BoyerMooreSearchMatcher(
                                search, ignoreCase, wholeWord);

        return matcher;
    }
```

org/experiment/editor/search/SearchAndReplace.java, find() (Show more context)



```
            startOfLine = (buffer.getLineStartOffset(
                buffer.getLineOfOffset(start)) == start);
            endOfLine = true;
        }
        SearchMatcher.Match match = matcher.nextMatch(text,
            startOfLine,endOfLine,firstTime,reverse);
        if(match != null)
        {
```

```
            startOfLine = (buffer.getLineStartOff
                buffer.getLineOfOffset(start)) == start);
            endOfLine = true;
        }

        String noWordSep =
            (String) buffer.getMode().getProperty("noWordSep");
        matcher.setNoWordSep(noWordSep);
        SearchMatcher.Match match = matcher.nextMatch(text,
            startOfLine,endOfLine,firstTime,reverse);
        if(match != null)
        {
```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java, isEndWord()

```
    private boolean isEndWord(char current, char next)
    {
        int currentCharType = TextUtilities.getCharType(current, noWordSep);
        if (currentCharType != TextUtilities.WORD_CHAR)
            return true;

        int nextCharType = TextUtilities.getCharType(next, noWordSep);
        return nextCharType == TextUtilities.WORD_CHAR;
    }
```

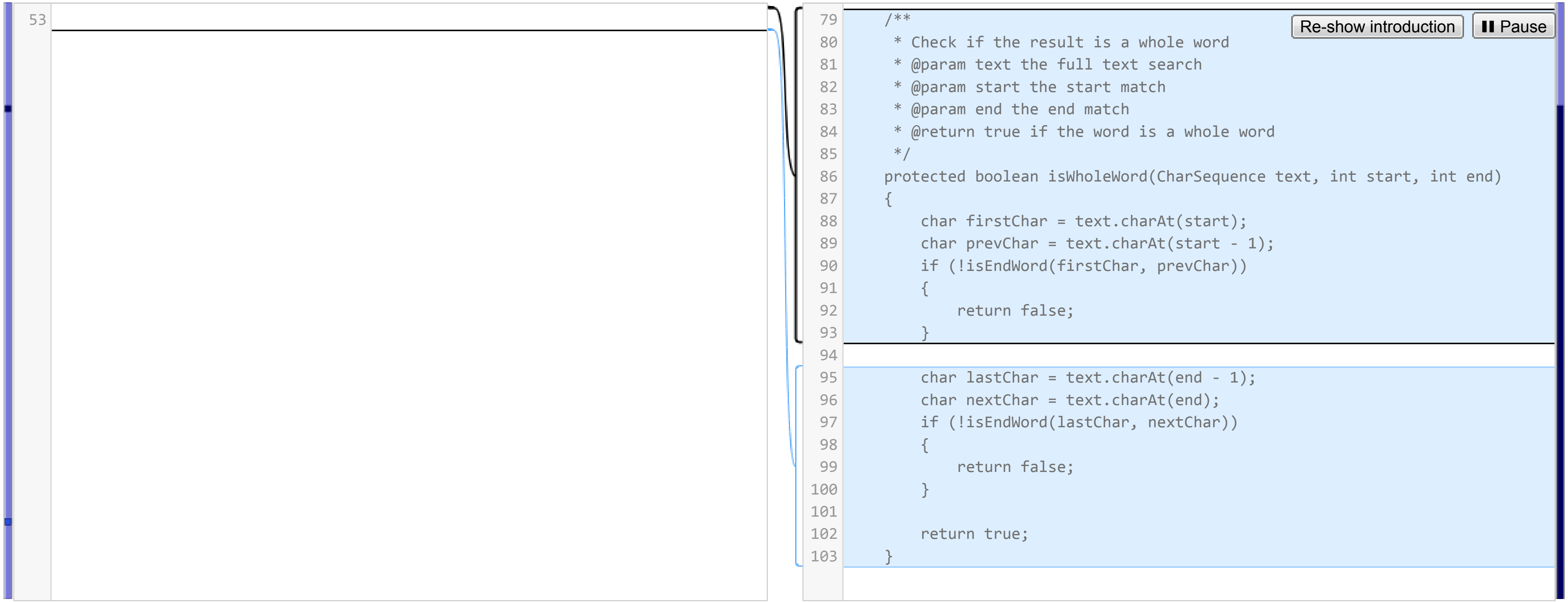
org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java, setNoWordSep()

```
    /**
     * @param noWordSep the chars that are considered as word chars for
     *                    this search
     * @since version 4.5pre1
     */
    public void setNoWordSep(String noWordSep)
    {
        if (this.noWordSep == null)
            this.noWordSep = "_";
        else
            this.noWordSep = noWordSep;
    }
```

org/experiment/editor/search/SearchMatcher.java

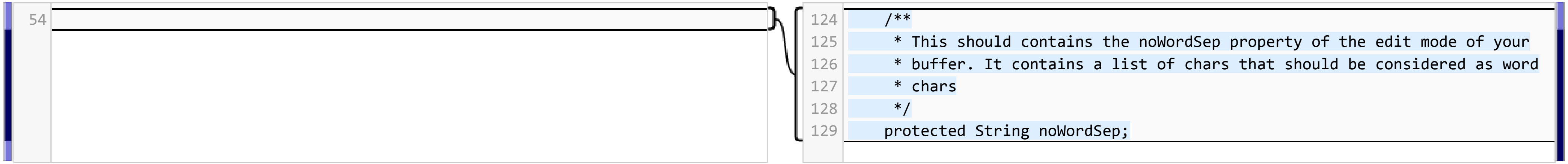
org/experiment/editor/search/SearchMatcher.java, isWholeWord()



```java
    /**
     * Check if the result is a whole word
     * @param text the full text search
     * @param start the start match
     * @param end the end match
     * @return true if the word is a whole word
     */
    protected boolean isWholeWord(CharSequence text, int start, int end)
    {
        char firstChar = text.charAt(start);
        char prevChar = text.charAt(start - 1);
        if (!isEndWord(firstChar, prevChar))
        {
            return false;
        }

        char lastChar = text.charAt(end - 1);
        char nextChar = text.charAt(end);
        if (!isEndWord(lastChar, nextChar))
        {
            return false;
        }

        return true;
    }
```

org/experiment/editor/search/SearchMatcher.java

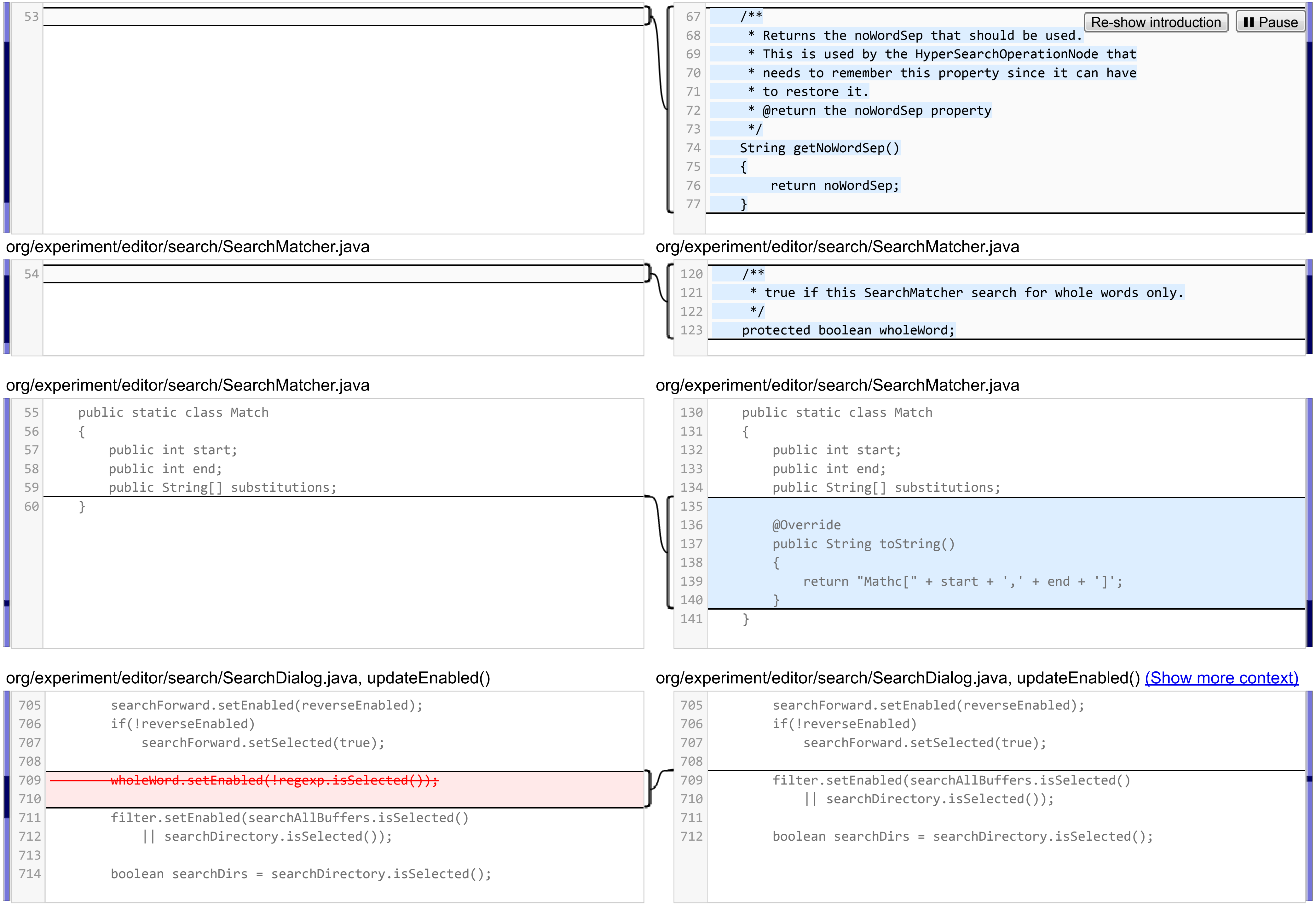
org/experiment/editor/search/SearchMatcher.java



```java
    /**
     * This should contains the noWordSep property of the edit mode of your
     * buffer. It contains a list of chars that should be considered as word
     * chars
     */
    protected String noWordSep;
```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java, getNoWordSep()



```
    /**
     * Returns the noWordSep that should be used.
     * This is used by the HyperSearchOperationNode that
     * needs to remember this property since it can have
     * to restore it.
     * @return the noWordSep property
     */
    String getNoWordSep()
    {
        return noWordSep;
    }
```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java

```
    /**
     * true if this SearchMatcher search for whole words only.
     */
    protected boolean wholeWord;
```

org/experiment/editor/search/SearchMatcher.java

```
    public static class Match
    {
        public int start;
        public int end;
        public String[] substitutions;
    }
```

org/experiment/editor/search/SearchMatcher.java

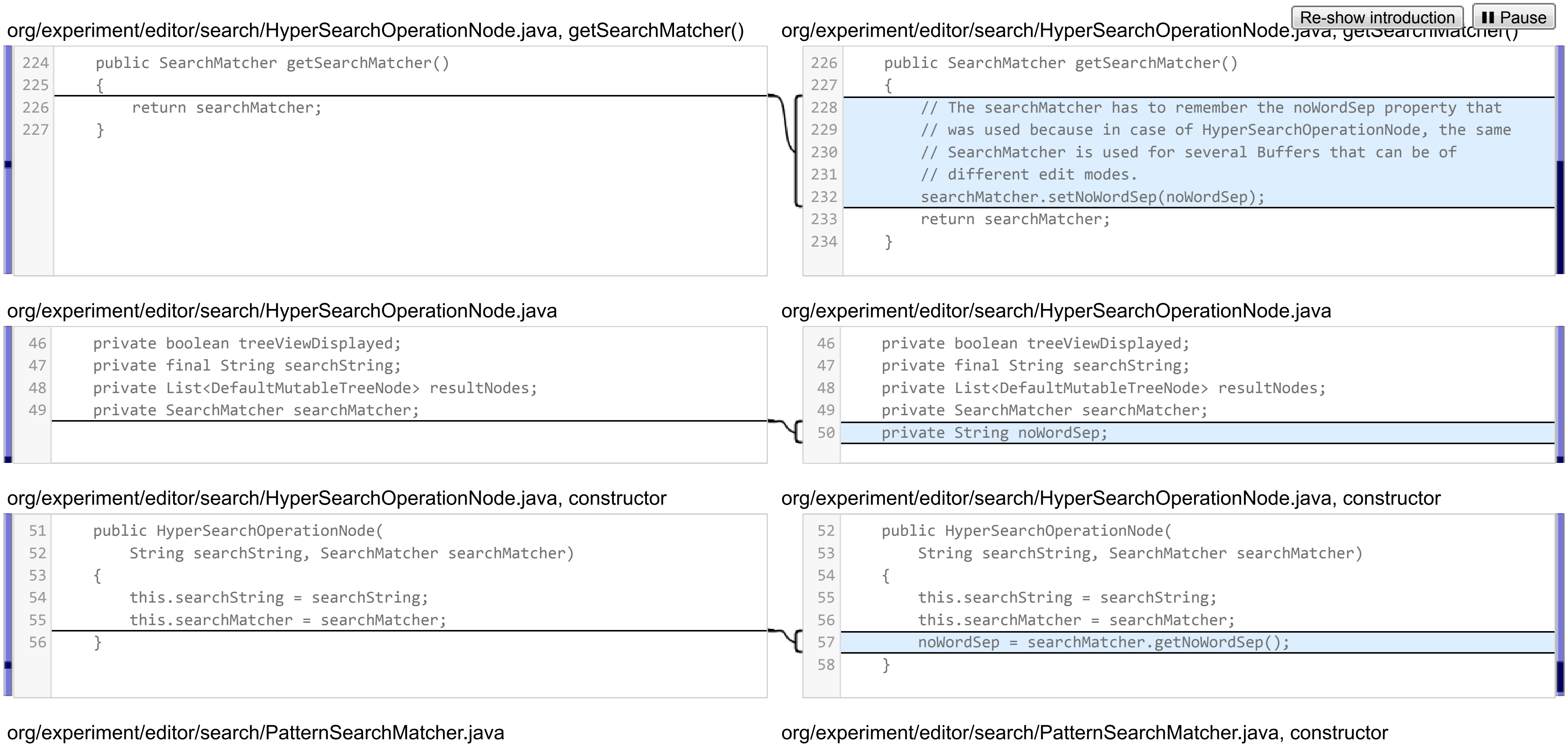
```
    public static class Match
    {
        public int start;
        public int end;
        public String[] substitutions;

        @Override
        public String toString()
        {
            return "Mathc[" + start + ',' + end + ']';
        }
    }
```

org/experiment/editor/search/SearchDialog.java, updateEnabled()

```
        searchForward.setEnabled(reverseEnabled);
        if(!reverseEnabled)
            searchForward.setSelected(true);

        ~~wholeWord.setEnabled(!regexp.isSelected());~~

        filter.setEnabled(searchAllBuffers.isSelected()
            || searchDirectory.isSelected());

        boolean searchDirs = searchDirectory.isSelected();
```

org/experiment/editor/search/SearchDialog.java, updateEnabled() (Show more context)

```
        searchForward.setEnabled(reverseEnabled);
        if(!reverseEnabled)
            searchForward.setSelected(true);

        filter.setEnabled(searchAllBuffers.isSelected()
            || searchDirectory.isSelected());

        boolean searchDirs = searchDirectory.isSelected();
```



org/experiment/editor/search/HyperSearchOperationNode.java, getSearchMatcher()

```
    public SearchMatcher getSearchMatcher()
    {
        return searchMatcher;
    }
```

org/experiment/editor/search/HyperSearchOperationNode.java, getSearchMatcher()

```
    public SearchMatcher getSearchMatcher()
    {
        // The searchMatcher has to remember the noWordSep property that
        // was used because in case of HyperSearchOperationNode, the same
        // SearchMatcher is used for several Buffers that can be of
        // different edit modes.
        searchMatcher.setNoWordSep(noWordSep);
        return searchMatcher;
    }
```

org/experiment/editor/search/HyperSearchOperationNode.java

```
    private boolean treeViewDisplayed;
    private final String searchString;
    private List<DefaultMutableTreeNode> resultNodes;
    private SearchMatcher searchMatcher;
```

org/experiment/editor/search/HyperSearchOperationNode.java

```
    private boolean treeViewDisplayed;
    private final String searchString;
    private List<DefaultMutableTreeNode> resultNodes;
    private SearchMatcher searchMatcher;
    private String noWordSep;
```

org/experiment/editor/search/HyperSearchOperationNode.java, constructor

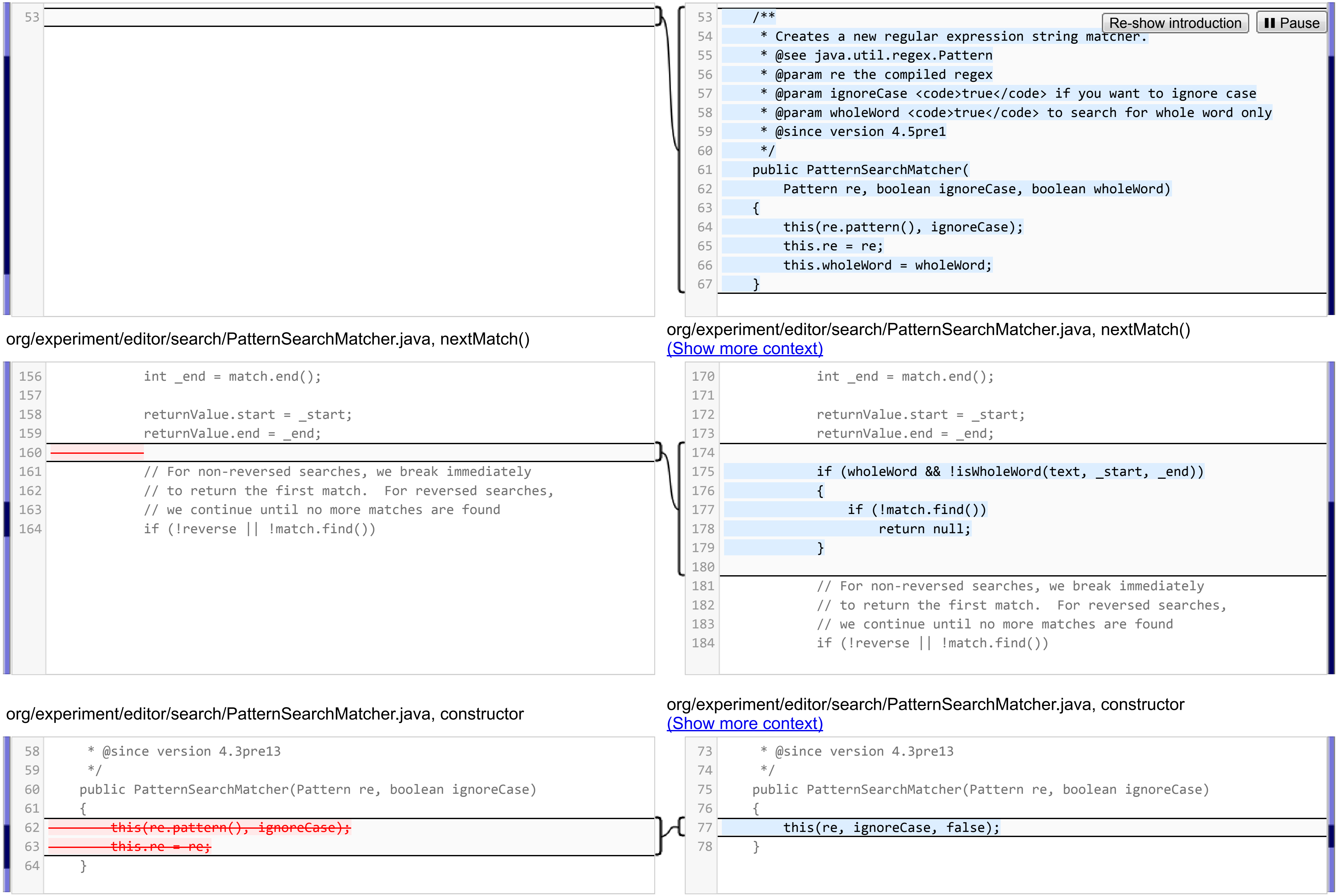
```
    public HyperSearchOperationNode(
        String searchString, SearchMatcher searchMatcher)
    {
        this.searchString = searchString;
        this.searchMatcher = searchMatcher;
    }
```

org/experiment/editor/search/HyperSearchOperationNode.java, constructor

```
    public HyperSearchOperationNode(
        String searchString, SearchMatcher searchMatcher)
    {
        this.searchString = searchString;
        this.searchMatcher = searchMatcher;
        noWordSep = searchMatcher.getNoWordSep();
    }
```

org/experiment/editor/search/PatternSearchMatcher.java

org/experiment/editor/search/PatternSearchMatcher.java, constructor



```
    /**
     * Creates a new regular expression string matcher.
     * @see java.util.regex.Pattern
     * @param re the compiled regex
     * @param ignoreCase <code>true</code> if you want to ignore case
     * @param wholeWord <code>true</code> to search for whole word only
     * @since version 4.5pre1
     */
    public PatternSearchMatcher(
        Pattern re, boolean ignoreCase, boolean wholeWord)
    {
        this(re.pattern(), ignoreCase);
        this.re = re;
        this.wholeWord = wholeWord;
    }
```

org/experiment/editor/search/PatternSearchMatcher.java, nextMatch()

```
            int _end = match.end();

            returnValue.start = _start;
            returnValue.end = _end;

            // For non-reversed searches, we break immediately
            // to return the first match.  For reversed searches,
            // we continue until no more matches are found
            if (!reverse || !match.find())
```

org/experiment/editor/search/PatternSearchMatcher.java, nextMatch()
[(Show more context)]()

```
            int _end = match.end();

            returnValue.start = _start;
            returnValue.end = _end;

            if (wholeWord && !isWholeWord(text, _start, _end))
            {
                if (!match.find())
                    return null;
            }

            // For non-reversed searches, we break immediately
            // to return the first match.  For reversed searches,
            // we continue until no more matches are found
            if (!reverse || !match.find())
```

org/experiment/editor/search/PatternSearchMatcher.java, constructor

```
     * @since version 4.3pre13
     */
    public PatternSearchMatcher(Pattern re, boolean ignoreCase)
    {
```
~~this(re.pattern(), ignoreCase);~~
~~this.re = re;~~
```
    }
```

org/experiment/editor/search/PatternSearchMatcher.java, constructor
[(Show more context)]()

```
     * @since version 4.3pre13
     */
    public PatternSearchMatcher(Pattern re, boolean ignoreCase)
    {
        this(re, ignoreCase, false);
    }
```

Re-show introduction ❚❚ Pause

org/experiment/editor/search/HyperSearchRequest.java, doHyperSearch()

```
    private int doHyperSearch(Buffer buffer, int start, int end,
        DefaultMutableTreeNode bufferNode)
    {
        int resultCount = 0;
        EditorTextArea textArea = Editor.getActiveView().getTextArea();
        int caretLine = textArea.getBuffer() == buffer ?
                textArea.getCaretLine() : -1;
```

org/experiment/editor/search/HyperSearchRequest.java, doHyperSearch()
(Show more context)

```
    private int doHyperSearch(Buffer buffer, int start, int end,
        DefaultMutableTreeNode bufferNode)
    {
        String noWordSep =
            (String) buffer.getMode().getProperty("noWordSep");
        matcher.setNoWordSep(noWordSep);
        int resultCount = 0;
        EditorTextArea textArea = Editor.getActiveView().getTextArea();
        int caretLine = textArea.getBuffer() == buffer ?
                textArea.getCaretLine() : -1;
```

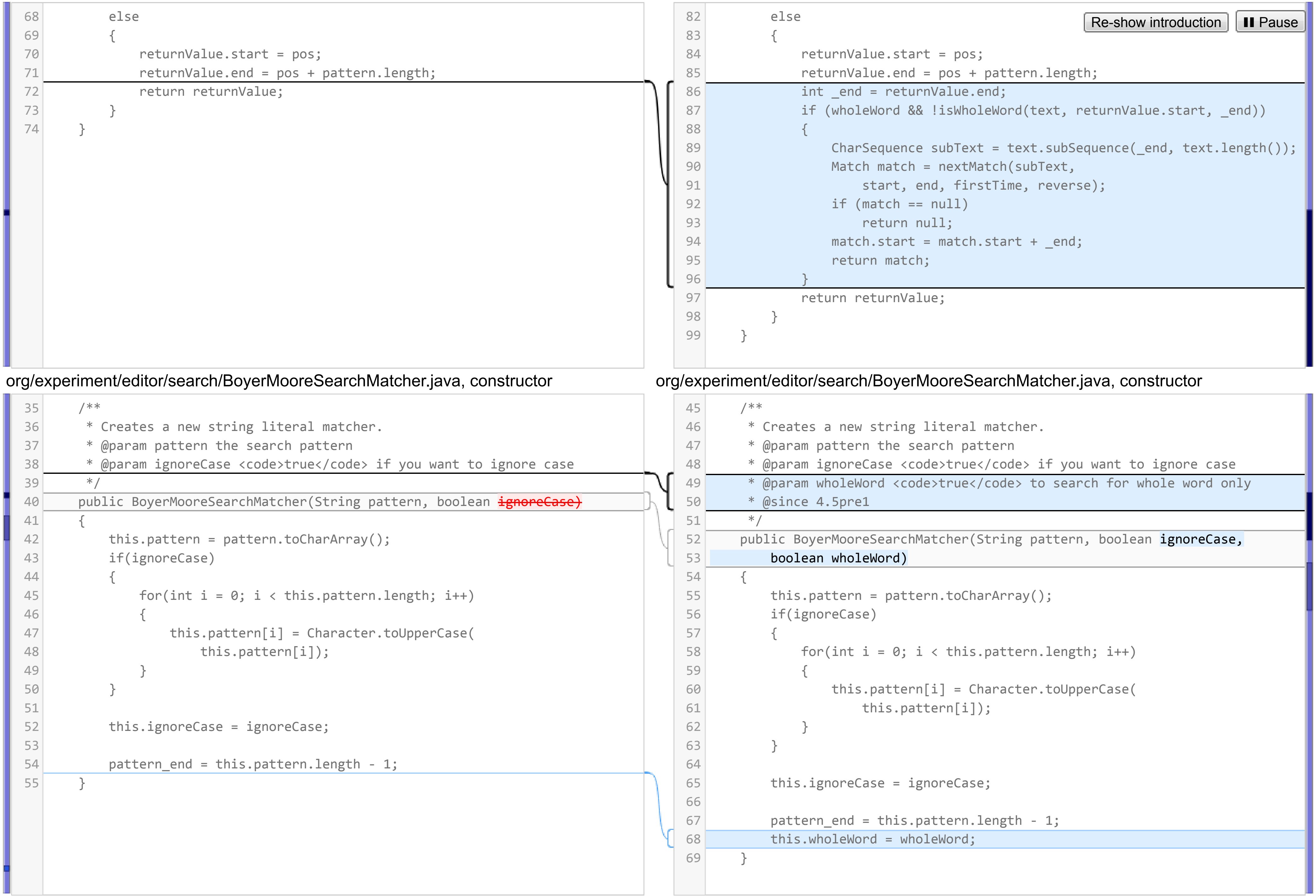
org/experiment/editor/search/BoyerMooreSearchMatcher.java

org/experiment/editor/search/BoyerMooreSearchMatcher.java, constructor

```
    /**
     * Creates a new string literal matcher.
     * @param pattern the search pattern
     * @param ignoreCase <code>true</code> if you want to ignore case
     */
    public BoyerMooreSearchMatcher(String pattern, boolean ignoreCase)
    {
        this(pattern, ignoreCase, false);
    }
```

org/experiment/editor/search/BoyerMooreSearchMatcher.java, nextMatch()

org/experiment/editor/search/BoyerMooreSearchMatcher.java, nextMatch()
(Show more context)



```
        else
        {
            returnValue.start = pos;
            returnValue.end = pos + pattern.length;
            return returnValue;
        }
    }
```

```
        else
        {
            returnValue.start = pos;
            returnValue.end = pos + pattern.length;
            int _end = returnValue.end;
            if (wholeWord && !isWholeWord(text, returnValue.start, _end))
            {
                CharSequence subText = text.subSequence(_end, text.length());
                Match match = nextMatch(subText,
                    start, end, firstTime, reverse);
                if (match == null)
                    return null;
                match.start = match.start + _end;
                return match;
            }
            return returnValue;
        }
    }
```

org/experiment/editor/search/BoyerMooreSearchMatcher.java, constructor

```
    /**
     * Creates a new string literal matcher.
     * @param pattern the search pattern
     * @param ignoreCase <code>true</code> if you want to ignore case
     */
    public BoyerMooreSearchMatcher(String pattern, boolean ~~ignoreCase)~~
    {
        this.pattern = pattern.toCharArray();
        if(ignoreCase)
        {
            for(int i = 0; i < this.pattern.length; i++)
            {
                this.pattern[i] = Character.toUpperCase(
                    this.pattern[i]);
            }
        }

        this.ignoreCase = ignoreCase;

        pattern_end = this.pattern.length - 1;
    }
```

org/experiment/editor/search/BoyerMooreSearchMatcher.java, constructor

```
    /**
     * Creates a new string literal matcher.
     * @param pattern the search pattern
     * @param ignoreCase <code>true</code> if you want to ignore case
     * @param wholeWord <code>true</code> to search for whole word only
     * @since 4.5pre1
     */
    public BoyerMooreSearchMatcher(String pattern, boolean ignoreCase,
        boolean wholeWord)
    {
        this.pattern = pattern.toCharArray();
        if(ignoreCase)
        {
            for(int i = 0; i < this.pattern.length; i++)
            {
                this.pattern[i] = Character.toUpperCase(
                    this.pattern[i]);
            }
        }

        this.ignoreCase = ignoreCase;

        pattern_end = this.pattern.length - 1;
        this.wholeWord = wholeWord;
    }
```

End review ▶

Re-show introduction ❚❚ Pause