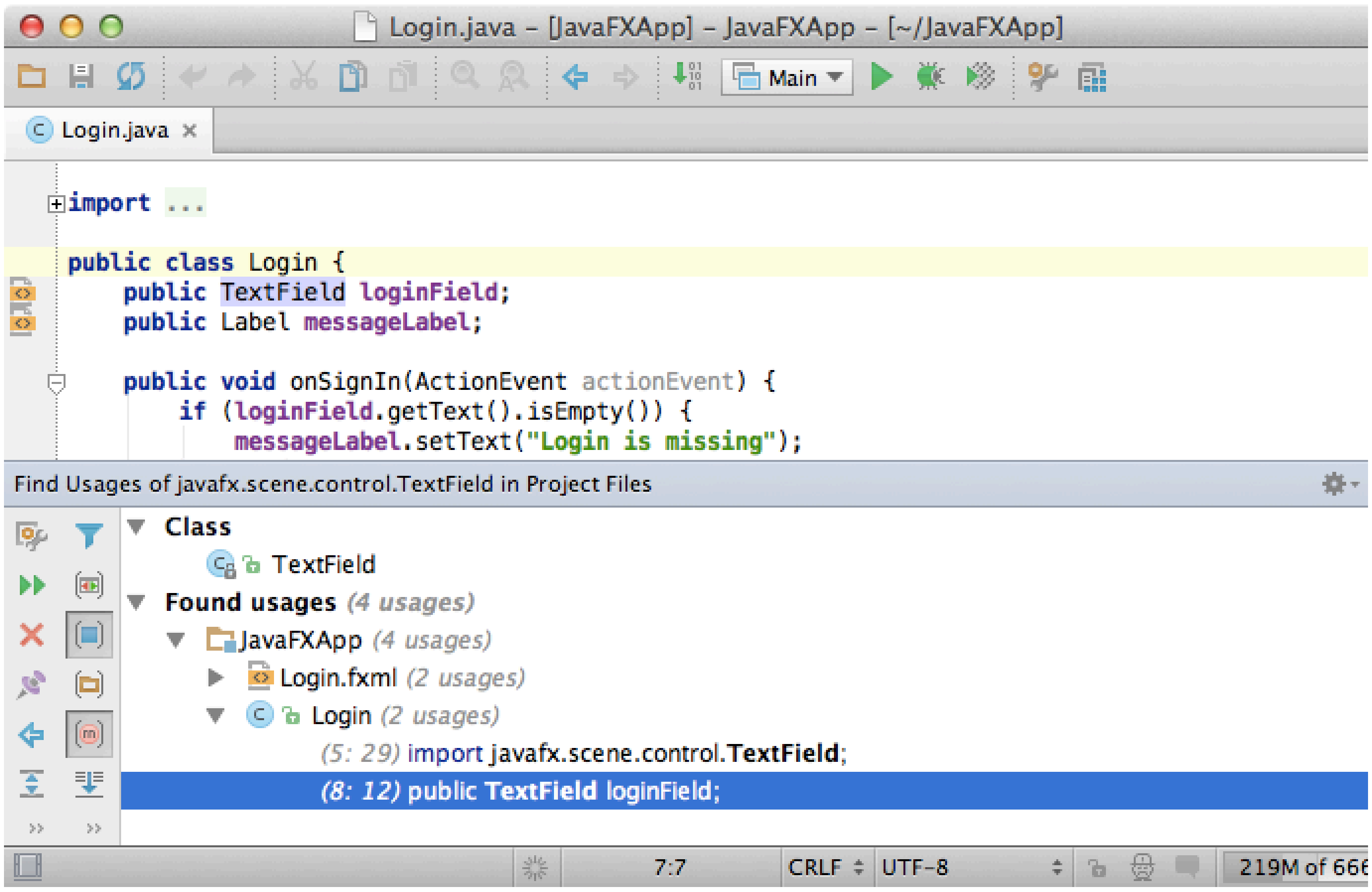
# Find Usages

Redirection Notice
This page will redirect to https://www.jetbrains.com/idea/help/searching-through-the-source-code.html.

Find usages helps you quickly find all pieces of code referencing a symbol at the caret, no matter if the symbol is a class, method, field, parameter, or another statement.

## 1. Find usages

Just press Alt + F7 and get a list of references grouped by type of usage, module and file. This feature is fast and gets you the first results almost immediately. More results appear as the IDE finds them.

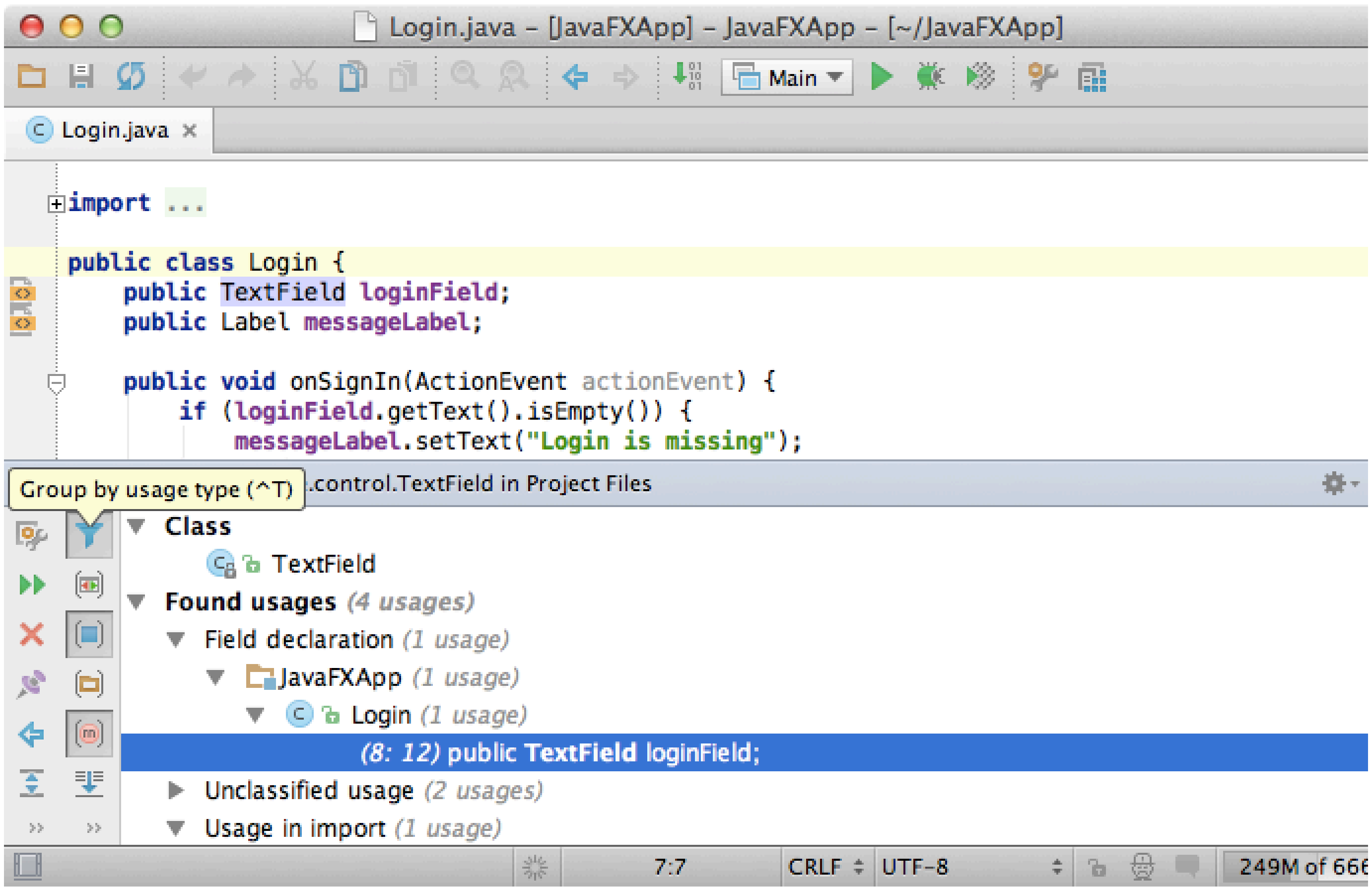


```
import ...

public class Login {
    public TextField loginField;
    public Label messageLabel;

    public void onSignIn(ActionEvent actionEvent) {
        if (loginField.getText().isEmpty()) {
            messageLabel.setText("Login is missing");
```

Find Usages of javafx.scene.control.TextField in Project Files

```
Class
    TextField
Found usages (4 usages)
    JavaFXApp (4 usages)
        Login.fxml (2 usages)
        Login (2 usages)
            (5: 29) import javafx.scene.control.TextField;
            (8: 12) public TextField loginField;
```

By default the results are not grouped by usage type, but you can enable this by pressing Ctrl + Alt + T or by clicking the corresponding button on the sidebar.

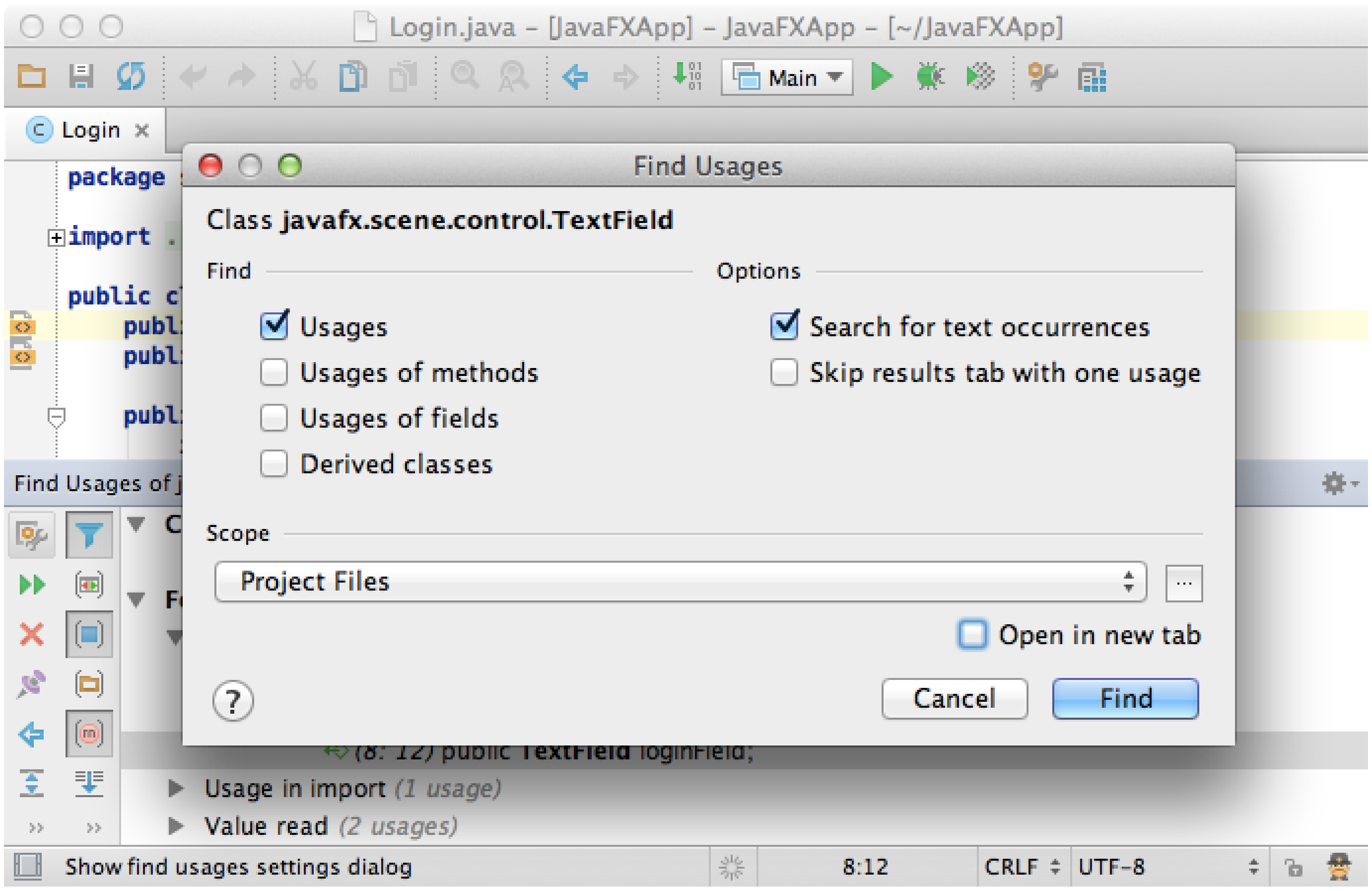


```
import ...

public class Login {
    public TextField loginField;
    public Label messageLabel;

    public void onSignIn(ActionEvent actionEvent) {
        if (loginField.getText().isEmpty()) {
            messageLabel.setText("Login is missing");
```

```
.control.TextField in Project Files
Class
    TextField
Found usages (4 usages)
    Field declaration (1 usage)
        JavaFXApp (1 usage)
            Login (1 usage)
                (8: 12) public TextField loginField;
    Unclassified usage (2 usages)
    Usage in import (1 usage)
```

## 2. Options

If you want to set custom options for the Find Usages algorithm, you can use Shift + Alt + Ctrl + F7 (Shift + Alt + Cmd + F7 fo r Mac) or click the first button on the right panel with search results.


Login.java – [JavaFXApp] – JavaFXApp – [~/JavaFXApp]
Main
Login
Find Usages of j
Find Usages
Usage in import (1 usage)
Value read (2 usages)
Show find usages settings dialog
8:12
CRLF
UTF-8

Class **javafx.scene.control.TextField**

Find
- [x] Usages
- [ ] Usages of methods
- [ ] Usages of fields
- [ ] Derived classes

Options
- [x] Search for text occurrences
- [ ] Skip results tab with one usage

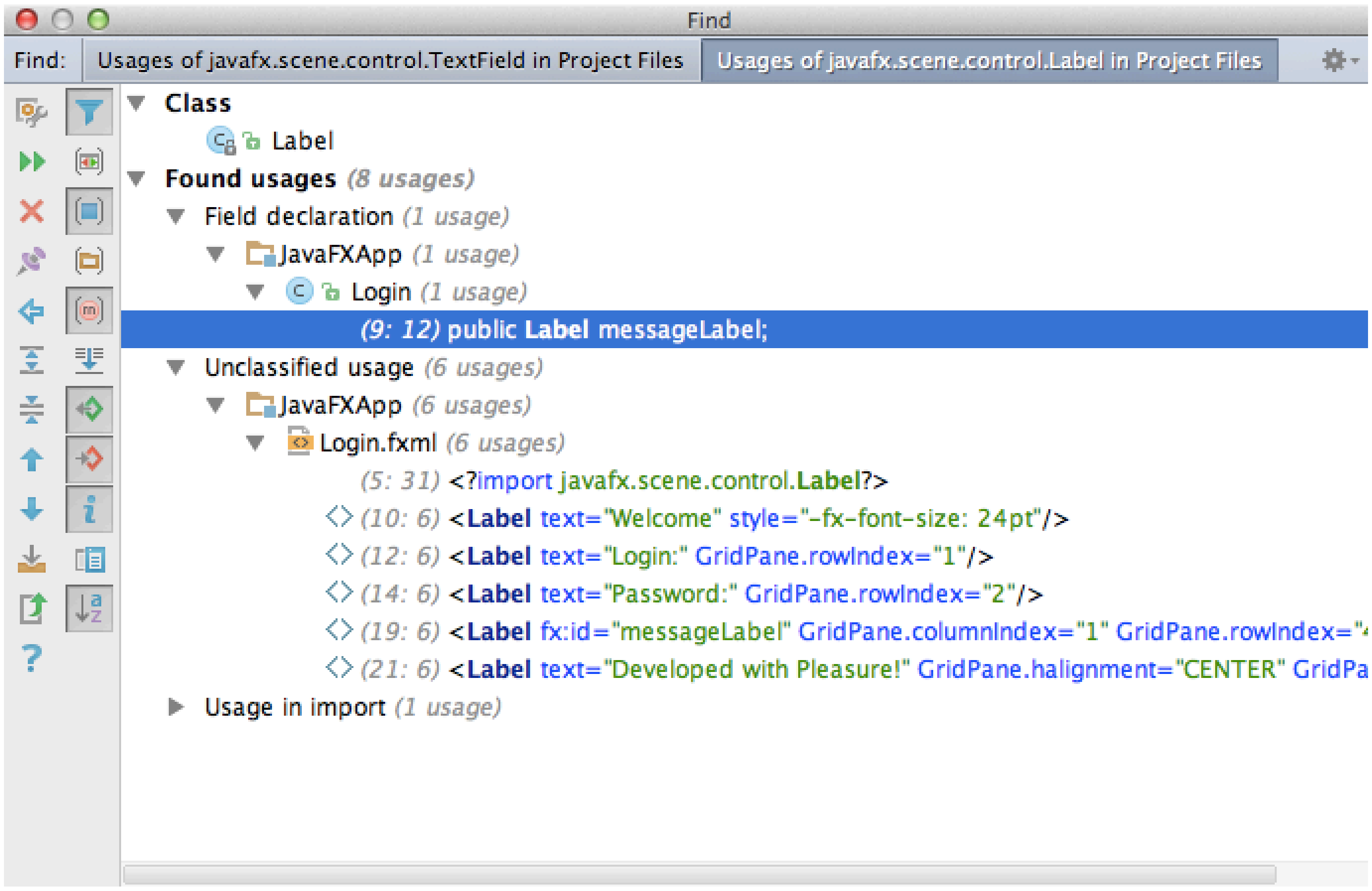
Scope: Project Files

- [ ] Open in new tab

Cancel | Find

## 3. Open in a new tab

If you want to keep the results of previous searches, tell the IDE to open new results in a new tab by checking the Open in new tab option.

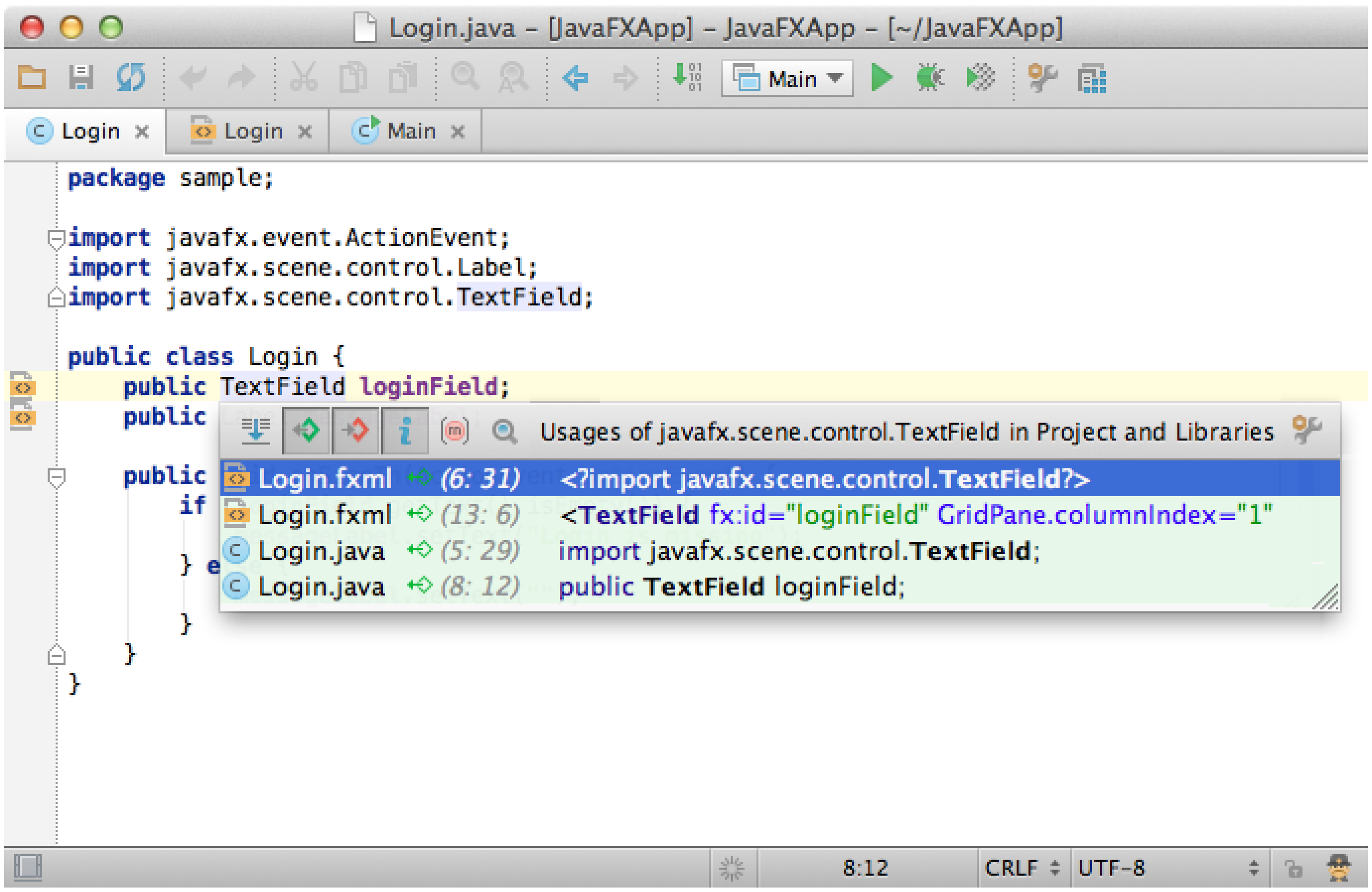


Find
Find:
Usages of javafx.scene.control.TextField in Project Files
Usages of javafx.scene.control.Label in Project Files

```
Class
    Label
Found usages (8 usages)
    Field declaration (1 usage)
        JavaFXApp (1 usage)
            Login (1 usage)
                (9: 12) public Label messageLabel;
    Unclassified usage (6 usages)
        JavaFXApp (6 usages)
            Login.fxml (6 usages)
                (5: 31) <?import javafx.scene.control.Label?>
                (10: 6) <Label text="Welcome" style="-fx-font-size: 24pt"/>
                (12: 6) <Label text="Login:" GridPane.rowIndex="1"/>
                (14: 6) <Label text="Password:" GridPane.rowIndex="2"/>
                (19: 6) <Label fx:id="messageLabel" GridPane.columnIndex="1" GridPane.rowIndex="
                (21: 6) <Label text="Developed with Pleasure!" GridPane.halignment="CENTER" GridPa
    Usage in import (1 usage)
```

## 4. Quick popup

If you want to see the results quickly without leaving the editor, simply press Alt + Ctrl + F7 (Alt + Cmd + F7 for Mac).



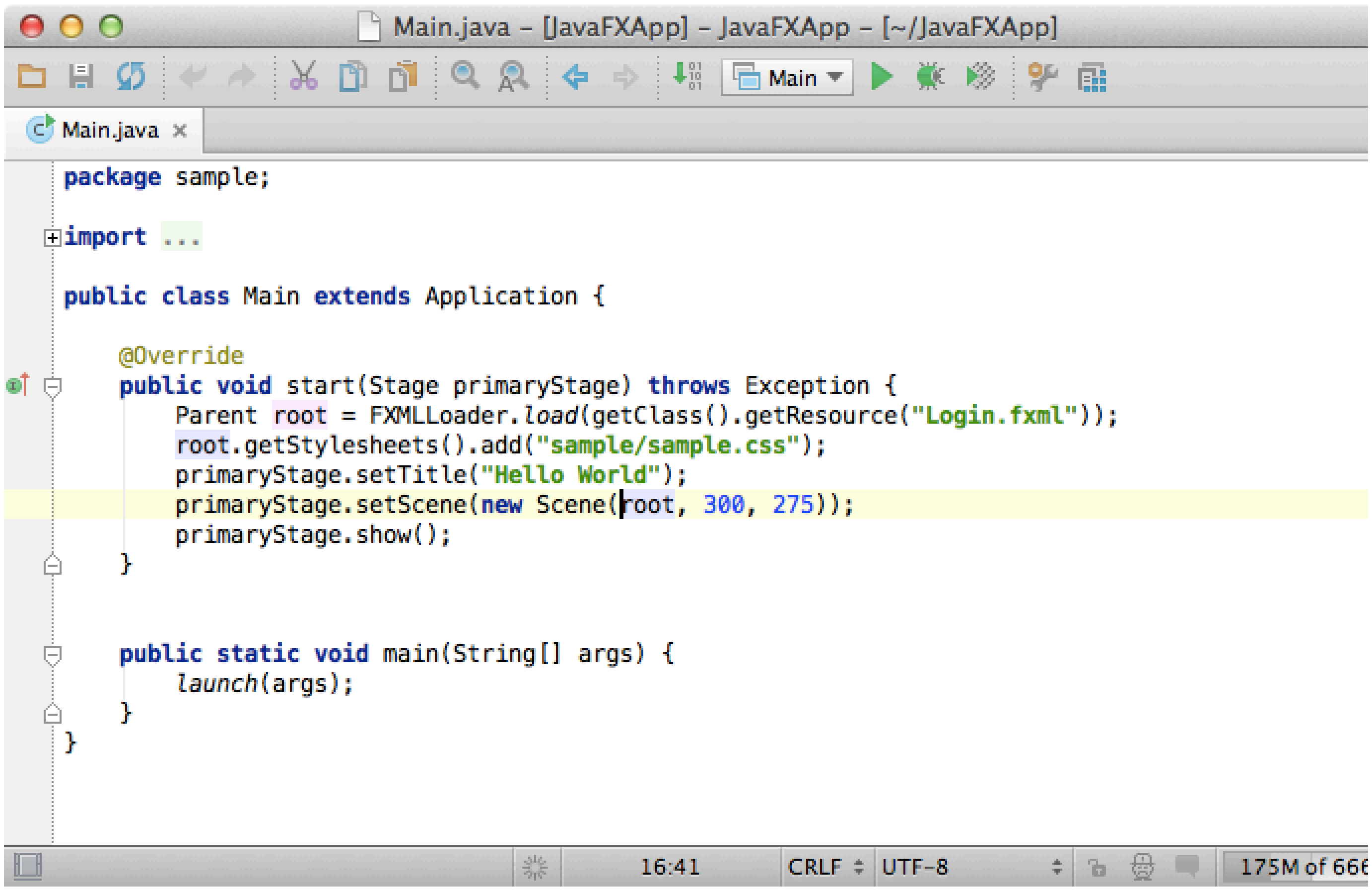
```
package sample;

import javafx.event.ActionEvent;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;

public class Login {
    public TextField loginField;
    public

    public
        if

        } e

        }
    }
}
```

Usages of javafx.scene.control.TextField in Project and Libraries

```
Login.fxml   (6: 31)   <?import javafx.scene.control.TextField?>
Login.fxml   (13: 6)   <TextField fx:id="loginField" GridPane.columnIndex="1"
Login.java   (5: 29)   import javafx.scene.control.TextField;
Login.java   (8: 12)   public TextField loginField;
```

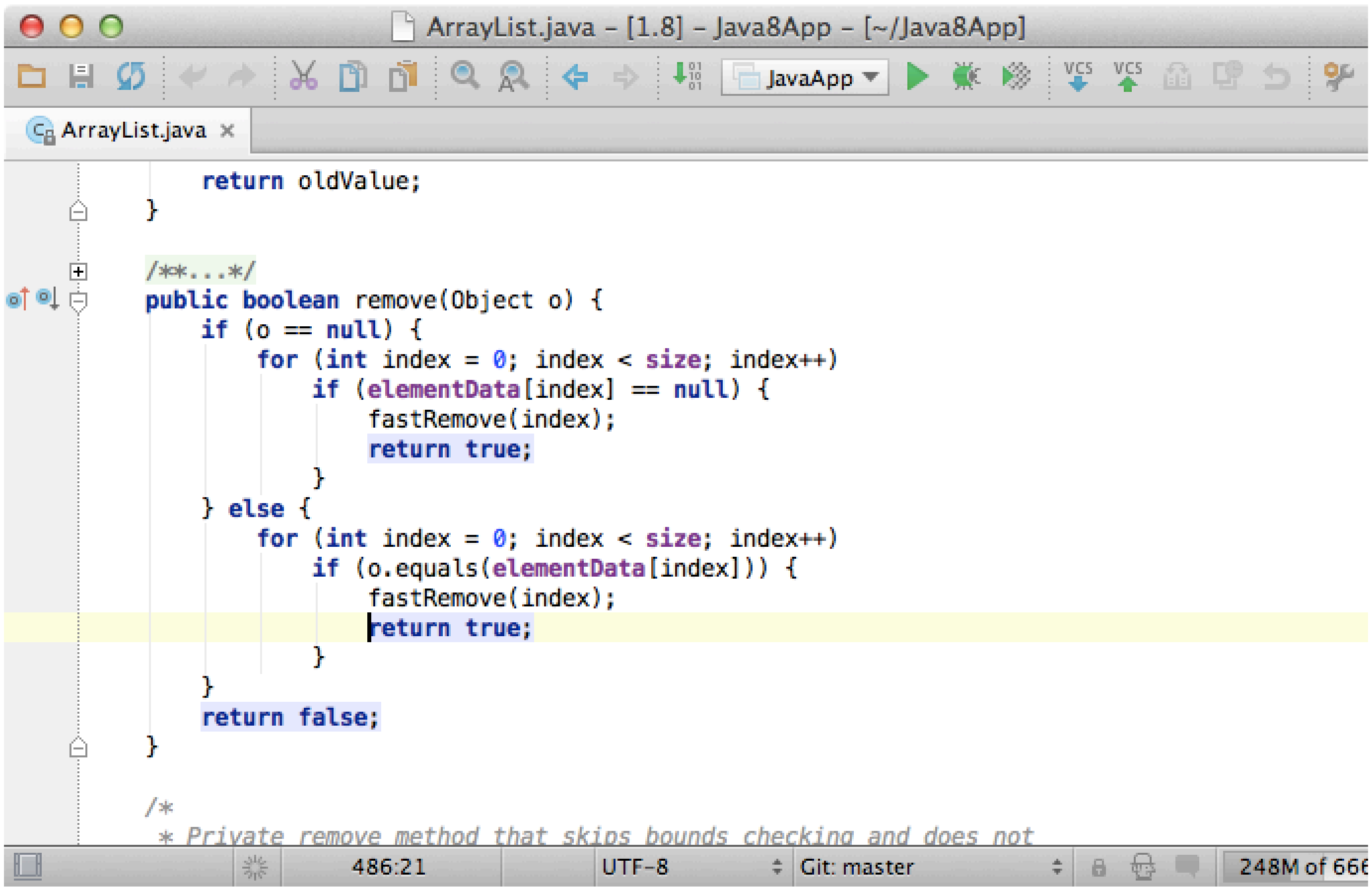
## 5. Highlight usages, return and throw statements

By default, IntelliJ IDEA highlights the usages of a symbol, return or throw statement at the caret within the opened file in the editor.



```java
package sample;

import ...

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("Login.fxml"));
        root.getStylesheets().add("sample/sample.css");
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```

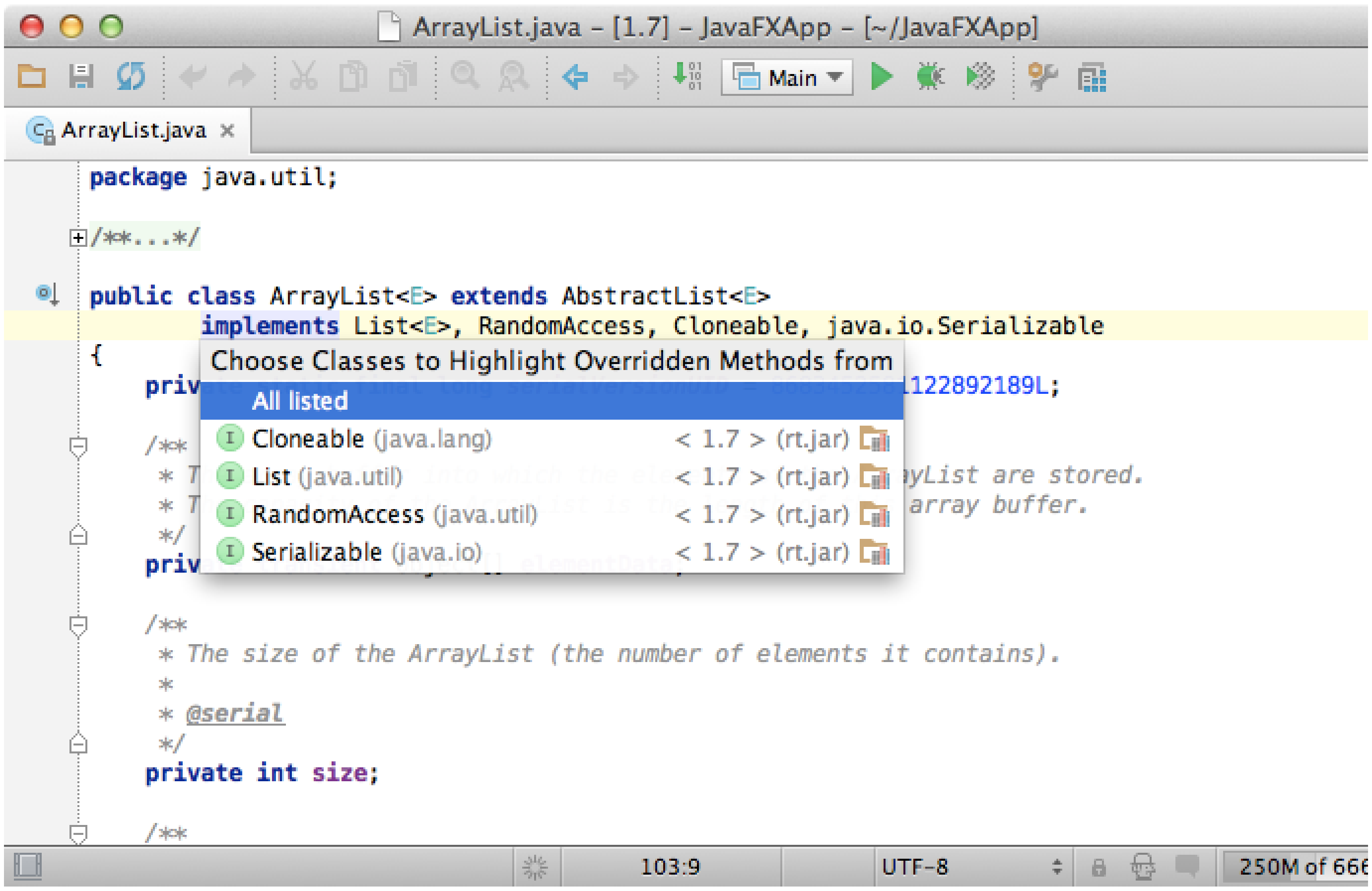
However, many developers prefer disabling this feature by deselecting Settings  Editor  Highlight usages of element at the caret and calling it via Shift + Ctrl + F7 (Shift + Cmd + F7 for Mac) instead, only when you need it. This way you can highlight more than one symbol if you want, and to remove the highlighting by simply pressing Esc.



```
        return oldValue;
    }

    /**...*/
    public boolean remove(Object o) {
        if (o == null) {
            for (int index = 0; index < size; index++)
                if (elementData[index] == null) {
                    fastRemove(index);
                    return true;
                }
        } else {
            for (int index = 0; index < size; index++)
                if (o.equals(elementData[index])) {
                    fastRemove(index);
                    return true;
                }
        }
        return false;
    }

    /*
     * Private remove method that skips bounds checking and does not
```

## 6. Highlight implemented and overridden methods

Another useful aspect of highlighting usages in IntelliJ IDEA is that you can easily find the methods that are overridden or implemented for a particular class or interface. Just put the caret at the implements or extends statement and press Shift + Ctrl + F7 (Shift + Cmd + F7 for Mac). If there are multiple classes or interfaces, you will be asked whose methods to highlight.


ArrayList.java - [1.7] - JavaFXApp - [~/JavaFXApp]

```
package java.util;

/**...*/

public class ArrayList<E> extends AbstractList<E>
        implements List<E>, RandomAccess, Cloneable, java.io.Serializable
{
    priv                                                    122892189L;

    /**
     * T                                                 yList are stored.
     * T                                                 array buffer.
     */
    priv

    /**
     * The size of the ArrayList (the number of elements it contains).
     *
     * @serial
     */
    private int size;

    /**
```

Choose Classes to Highlight Overridden Methods from

```
All listed
Cloneable (java.lang)          < 1.7 > (rt.jar)
List (java.util)               < 1.7 > (rt.jar)
RandomAccess (java.util)       < 1.7 > (rt.jar)
Serializable (java.io)         < 1.7 > (rt.jar)
```