

Intentions

[Previous](#) | [Code Style and Formatting](#)

[Top](#) | [Quick Start](#)

[Next](#) | [Version Control Basics](#)

The license could not be verified: License Certificate has expired!

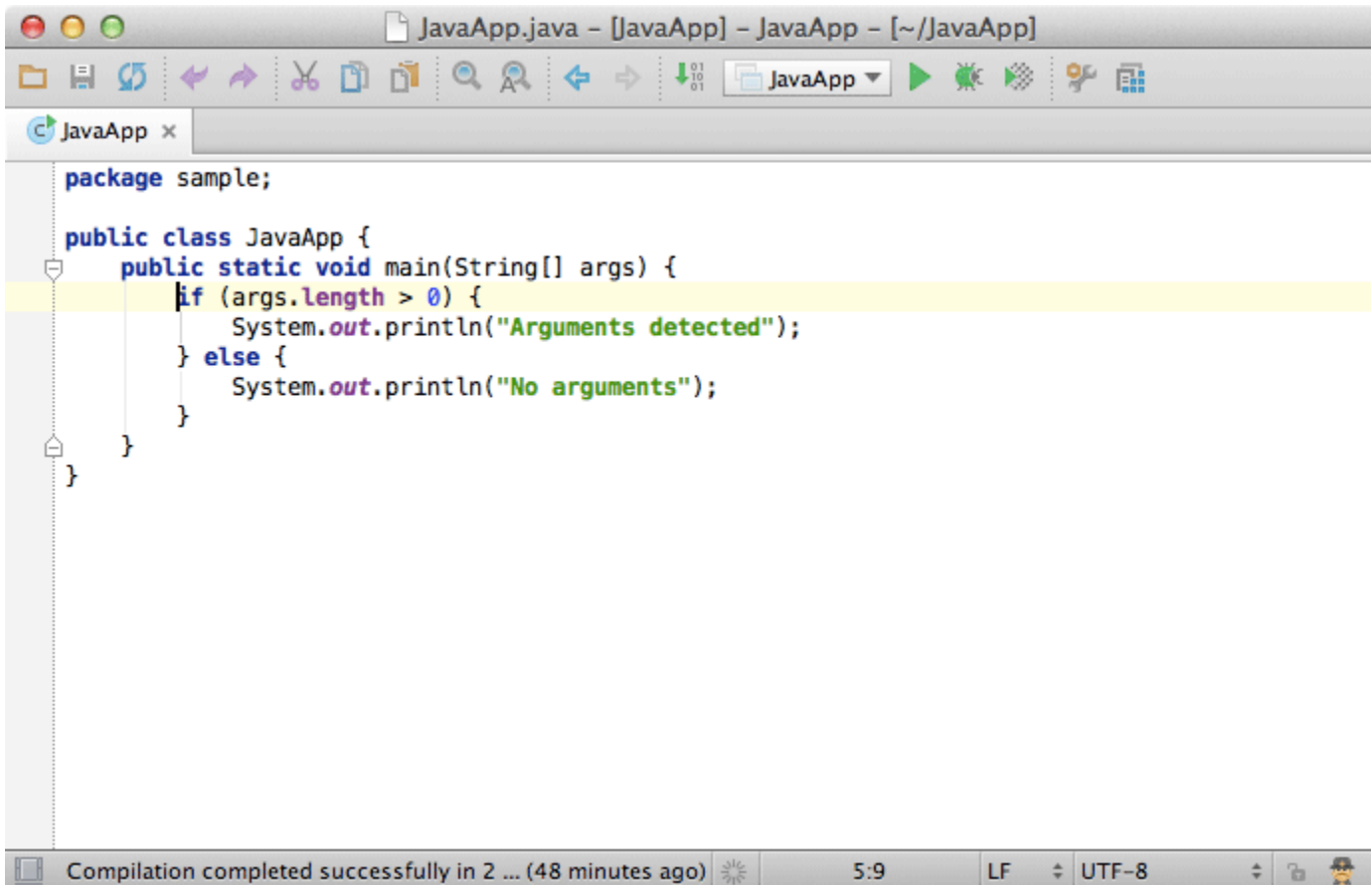
The license could not be verified: License Certificate has expired!

While inspections provide quick-fixes for code that has potential problems, intentions help you apply automatic changes to code that is correct.

To get a list of intentions applicable to the code at the caret, just press `Alt+Enter`.

1. Optimize statements and manage braces

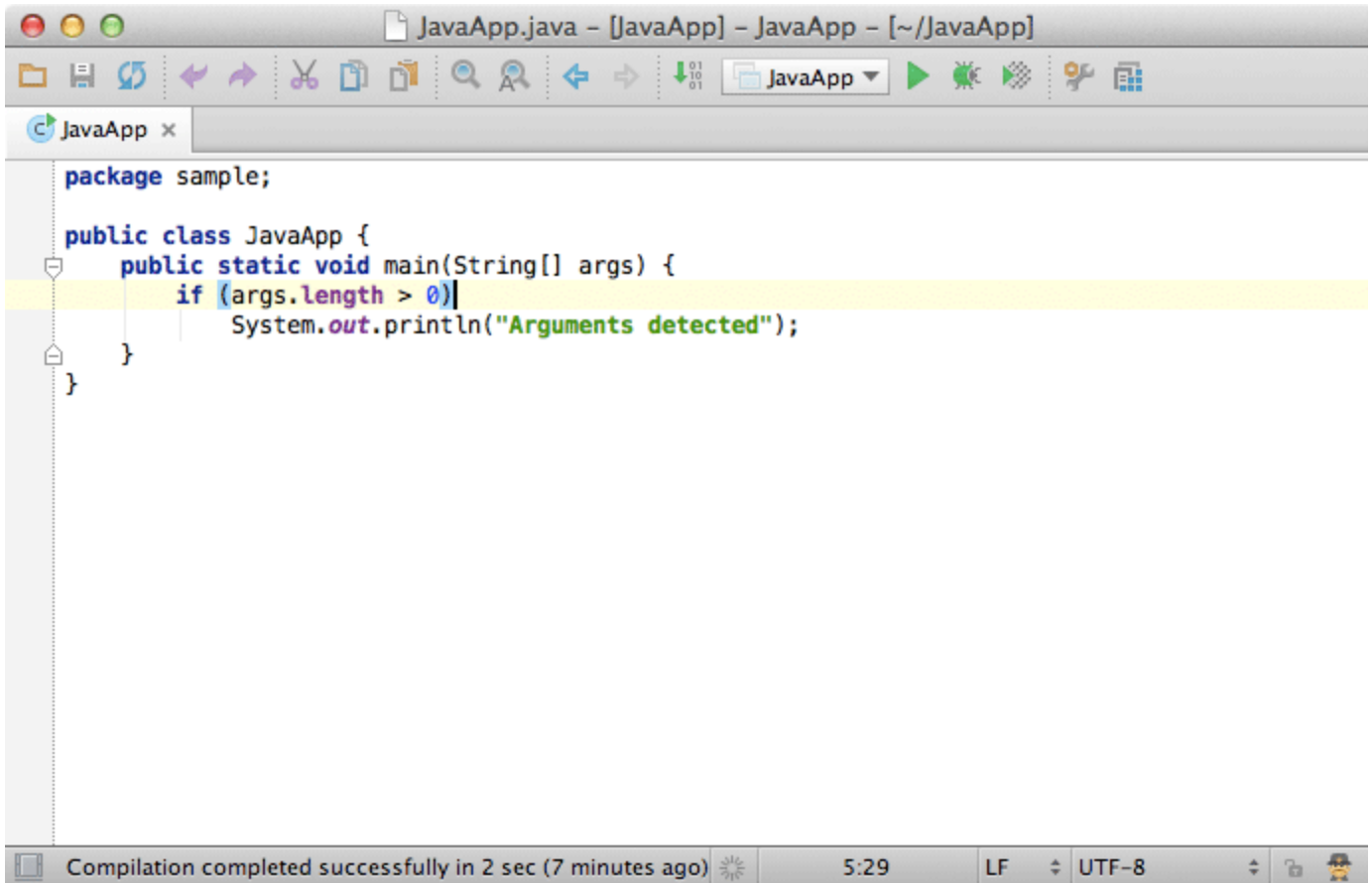
Since the IDE is aware of the data flow in your code, it can help you optimize Boolean conditions and transform if/else statements:



```
package sample;

public class JavaApp {
    public static void main(String[] args) {
        if (args.length > 0) {
            System.out.println("Arguments detected");
        } else {
            System.out.println("No arguments");
        }
    }
}
```

With intentions you can, for example, automatically add and remove braces for your statements:



```
package sample;

public class JavaApp {
    public static void main(String[] args) {
        if (args.length > 0)
            System.out.println("Arguments detected");
    }
}
```

Compilation completed successfully in 2 sec (7 minutes ago) 5:29 LF UTF-8

2. String concatenation

If you have String concatenation, the IDE can help you replace it with a `String.format` expression or a `StringBuilder` pattern.

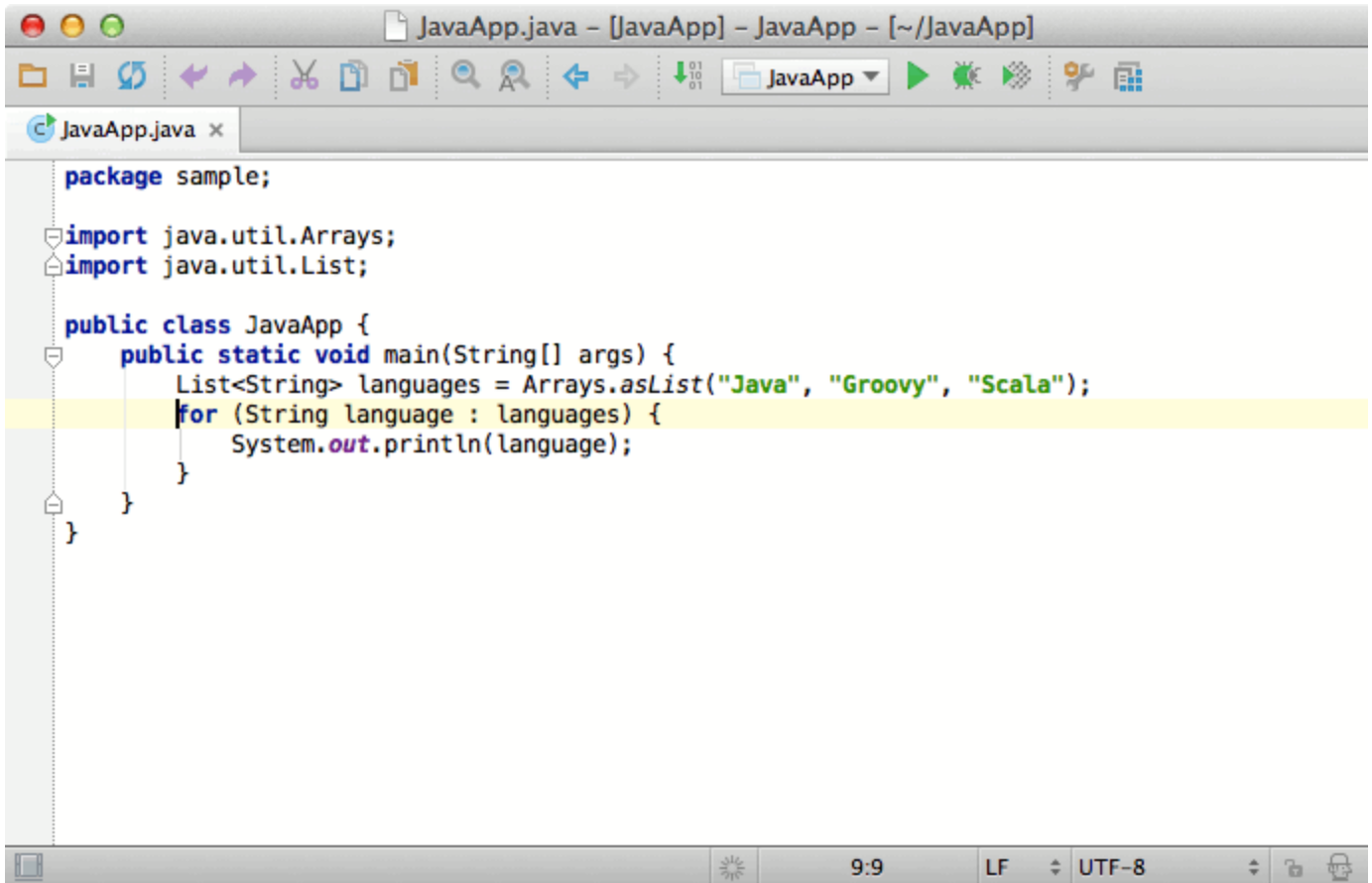
```
package sample;

public class JavaApp {
    public static void main(String[] args) {
        if (args.length == 1) {
            System.out.println("Hello " + args[0] + "!");
        }
    }
}
```

Compilation completed successfully in 2 ... (42 minutes ago) 6:56 LF UTF-8

3. Switch between loop declaration styles

With intentions you can also switch between loop declaration styles or even reverse its direction.



The image shows a screenshot of an IDE window titled "JavaApp.java - [JavaApp] - JavaApp - [~/JavaApp]". The code is as follows:

```
package sample;

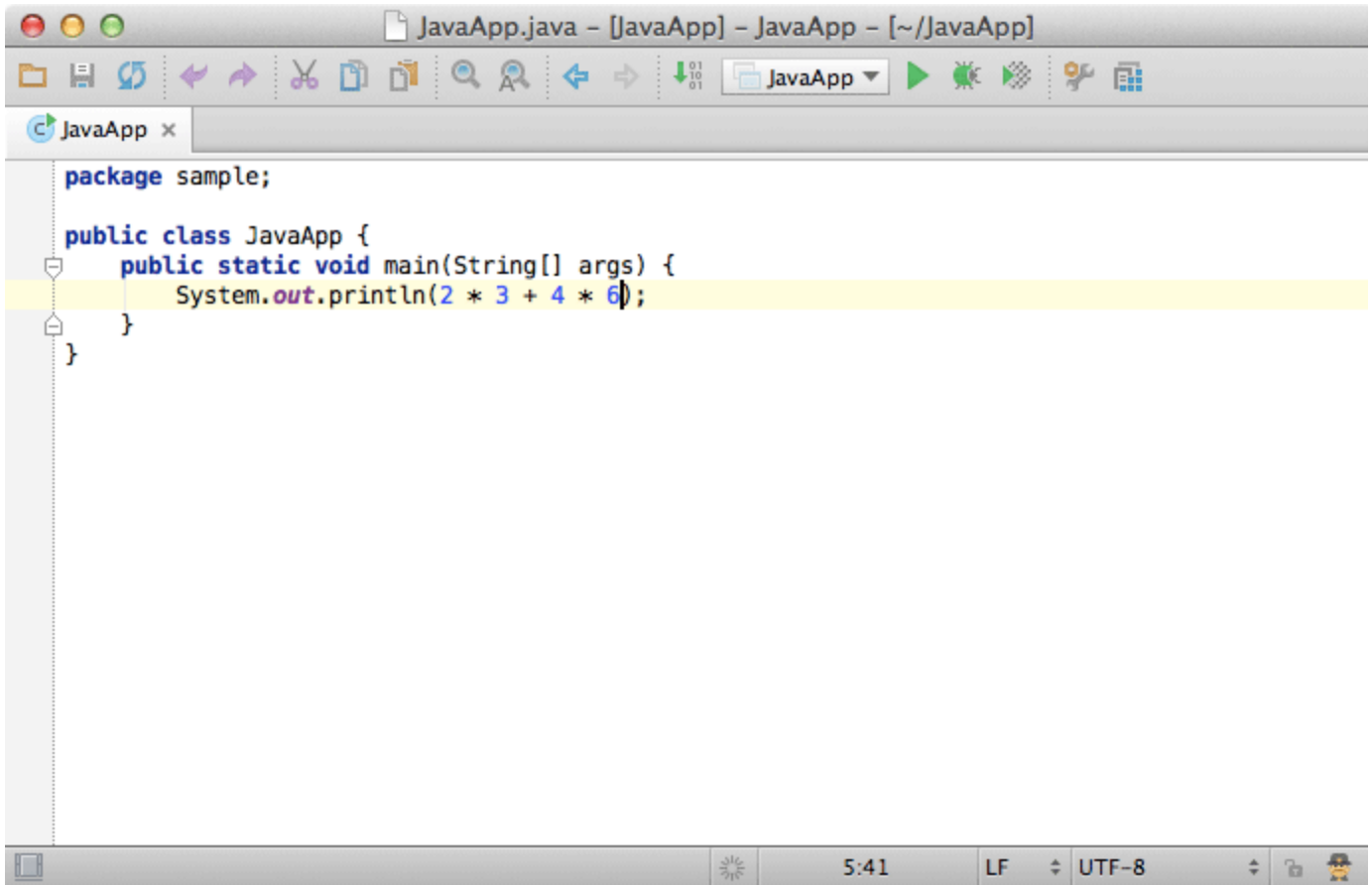
import java.util.Arrays;
import java.util.List;

public class JavaApp {
    public static void main(String[] args) {
        List<String> languages = Arrays.asList("Java", "Groovy", "Scala");
        for (String language : languages) {
            System.out.println(language);
        }
    }
}
```

The IDE interface includes a toolbar with various icons for file operations and execution. The status bar at the bottom shows the time as 9:9, the line ending as LF, and the encoding as UTF-8.

4. Math expressions

If you have a complicated math expression, the IDE will offer you to simplify an expression or possibly clarify it with parentheses.



```
JavaApp.java - [JavaApp] - JavaApp - [~/JavaApp]
package sample;

public class JavaApp {
    public static void main(String[] args) {
        System.out.println(2 * 3 + 4 * 6);
    }
}
```

The screenshot shows an IDE window titled "JavaApp.java - [JavaApp] - JavaApp - [~/JavaApp]". The code editor contains the following Java code:

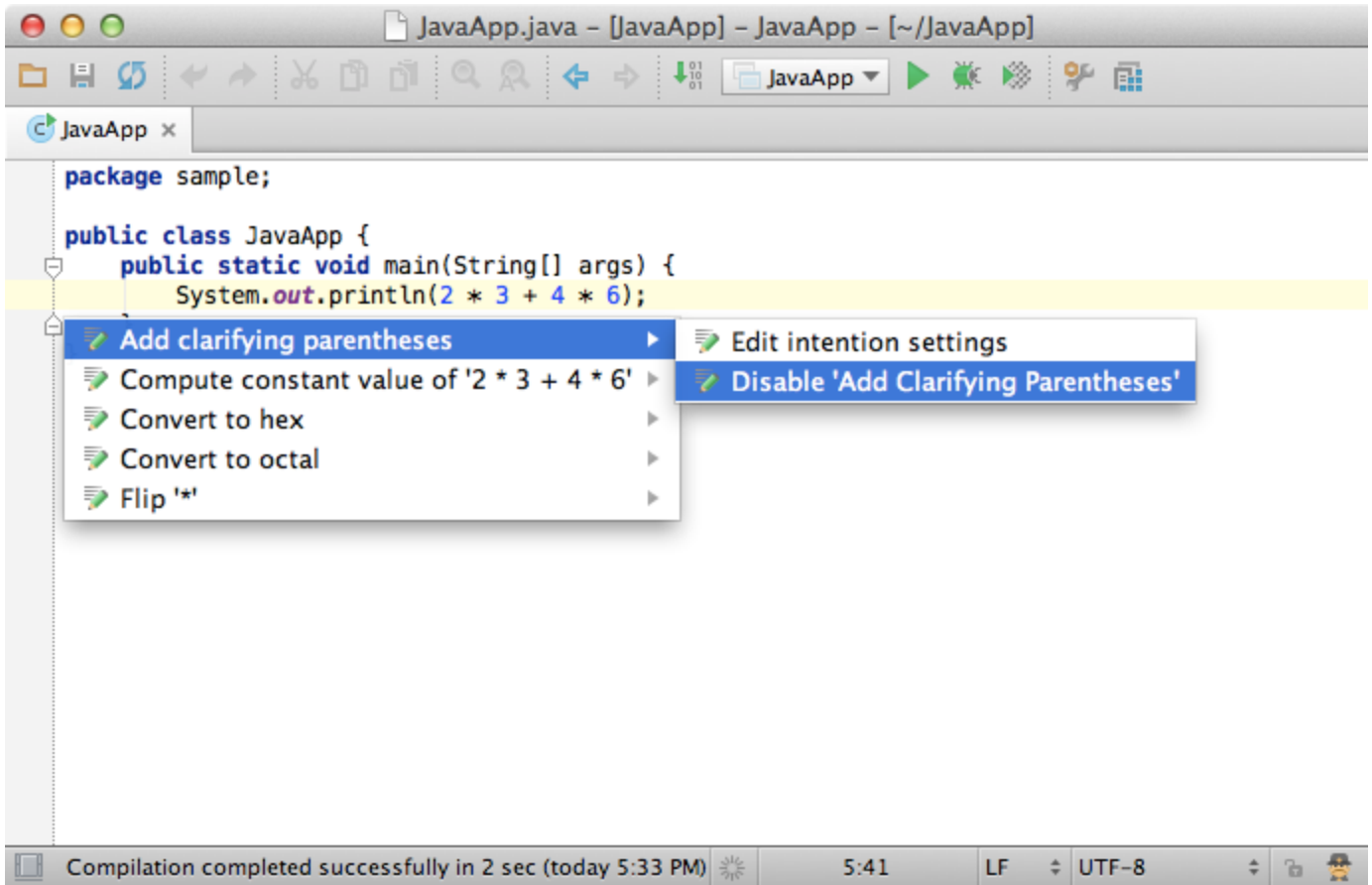
```
package sample;

public class JavaApp {
    public static void main(String[] args) {
        System.out.println(2 * 3 + 4 * 6);
    }
}
```

The line `System.out.println(2 * 3 + 4 * 6);` is highlighted in yellow. The IDE interface includes a toolbar with various icons for file operations, search, and execution. The status bar at the bottom shows the time as 5:41, the line ending as LF, and the encoding as UTF-8.

5. Disable intention

If you don't want to see a certain intention, you can disable it by selecting the right arrow on the intention.



6. Settings

The complete list of intentions (several hundred in total), grouped by language and framework, can be found in `SettingsIntention`s. Here you can enable/disable whole groups of intentions:

