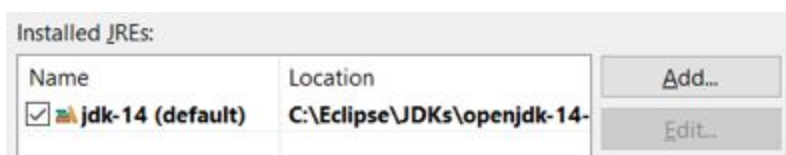


Eclipse IDE Java™ 14 Support Overview

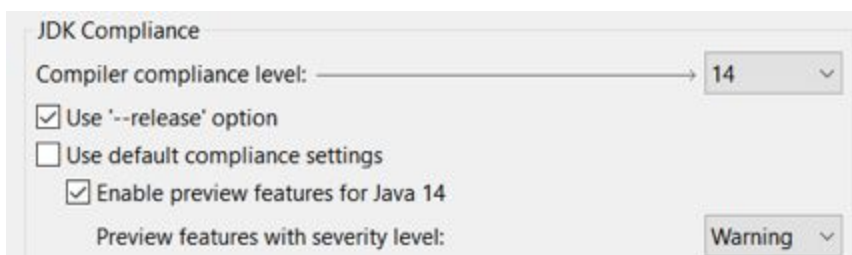
Here's a brief overview of how to use some of the main Java 14 features in the Eclipse IDE 2020-06 release.

Activating Java 14 and Enabling Preview Features

To start using Java 14 in the Eclipse IDE 2020-06 release, choose **Preferences > Java > Installed JREs**, then select the Java 14 Java Development Kit (JDK), as shown below.



To set the JDK compliance to 14 and enable the preview features, choose **Preferences > Java > Compiler**, then select the options shown below.



When a preview feature is used in the code, the compiler provides a default warning that the preview feature may not be supported in a future release. You can ignore the warning or set it to Info by changing its severity level on the page shown above.

To quickly enable the preview features on an existing Java project, right-click on it in the **Package/Project Explorer** and select **Configure > Enable preview features**, as shown below.



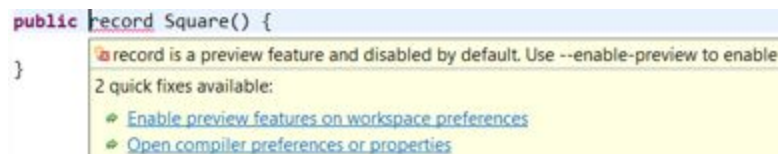
Creating and Using Records

Java 14 introduces records as a new preview feature.

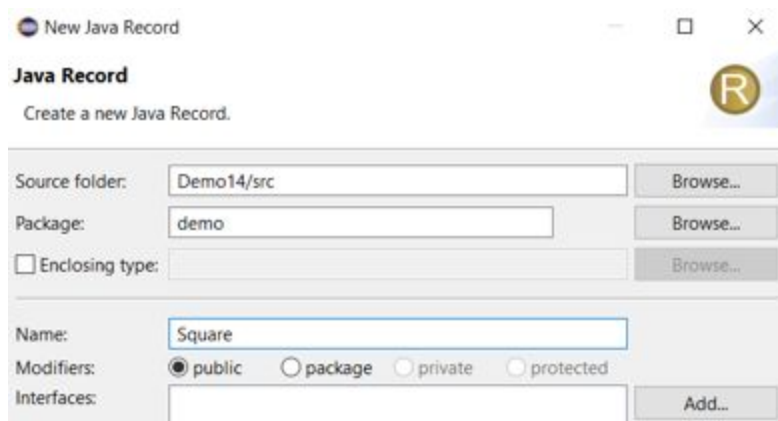
To create a record in the Eclipse IDE, use the `new_record` template in an empty `.java` file, as shown below.



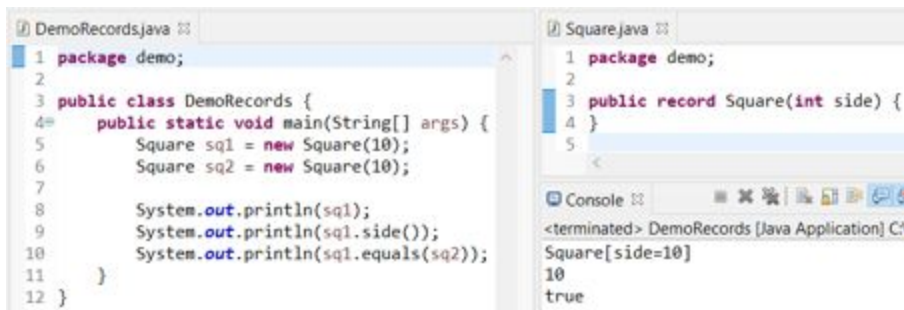
If preview is not already enabled, you can enable preview features and use the record through the quick fixes that are provided. Press `Ctrl + 1` to access quick fixes.



You can also create a record using the **New > Record wizard**, shown below. The wizard provides additional options, such as selecting the visibility modifier and adding the interfaces the record implements.



You can run a Java program using the record to verify that the record instance is provided with **auto-generated** constructor, component accessor, `toString`, `equals`, and `hashCode` methods, as shown below.

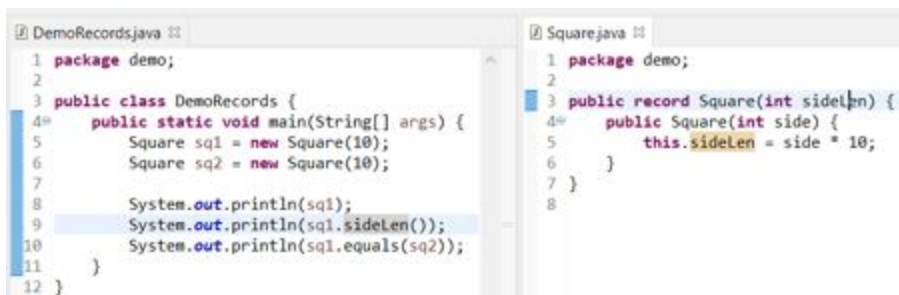


```
1 package demo;
2
3 public class DemoRecords {
4     public static void main(String[] args) {
5         Square sq1 = new Square(10);
6         Square sq2 = new Square(10);
7
8         System.out.println(sq1);
9         System.out.println(sq1.side());
10        System.out.println(sq1.equals(sq2));
11    }
12 }
```

```
1 package demo;
2
3 public record Square(int side) {
4 }
5
```

Console: <terminated> DemoRecords [Java Application] C:\
Square[side=10]
10
true

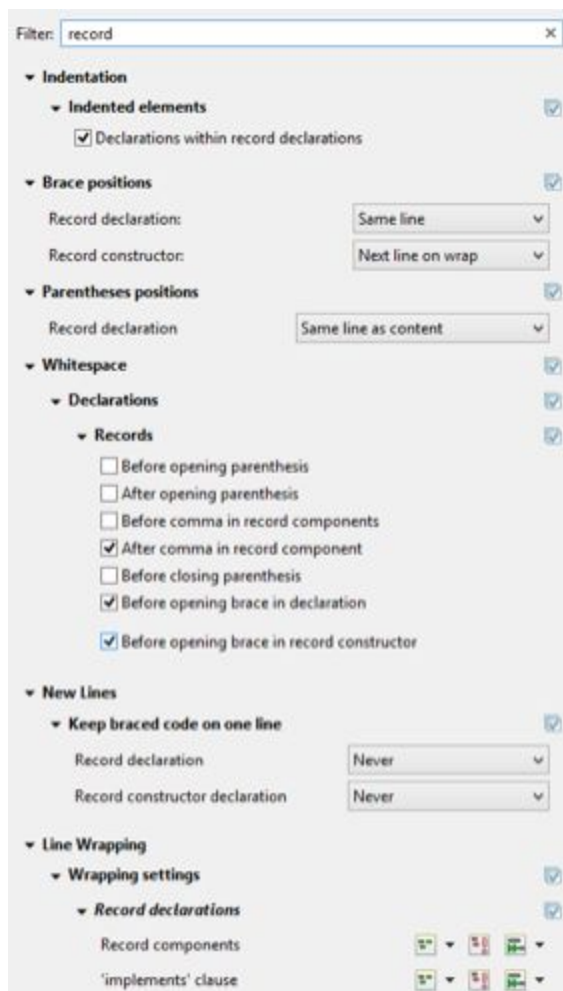
You can also perform the rename refactoring on record components and update the accessor method names along with component references, as shown below.



```
1 package demo;
2
3 public class DemoRecords {
4     public static void main(String[] args) {
5         Square sq1 = new Square(10);
6         Square sq2 = new Square(10);
7
8         System.out.println(sq1);
9         System.out.println(sq1.sideLen());
10        System.out.println(sq1.equals(sq2));
11    }
12 }
```

```
1 package demo;
2
3 public record Square(int sideLen) {
4     public Square(int side) {
5         this.sideLen = side * 10;
6     }
7 }
8
```

In addition, several new settings have been added to the formatter profile to control record formatting. Use the filter, shown below, to quickly view these configurable settings.



Creating and Formatting Text Blocks

Text blocks received a second round of preview in Java 14.

You can create a text block by enclosing it in triple quotes. The Eclipse IDE makes it easier to add these delimiters with the new keyboard shortcut **Ctrl + Shift + ' (apostrophe)**. You can also select an existing text block and use this key binding to quickly enclose it in text block delimiters, as shown below.

```
public void foo() {
    """
    Hello
    World
    """
}
```

Ctrl+Shift+' – Add Text Block

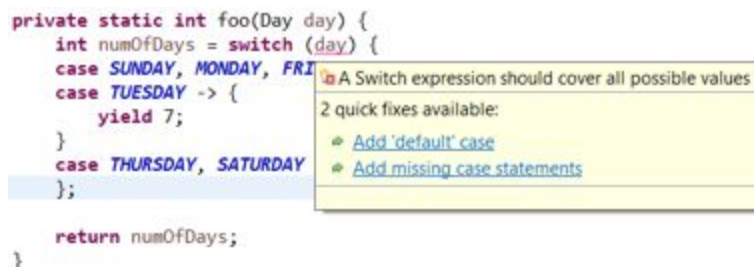
Use the formatter profile to configure text block indentation, as shown below.



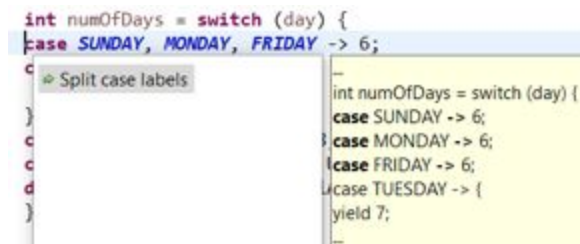
Handling Switch Expressions

Java 14 has promoted switch expressions to a standard feature.

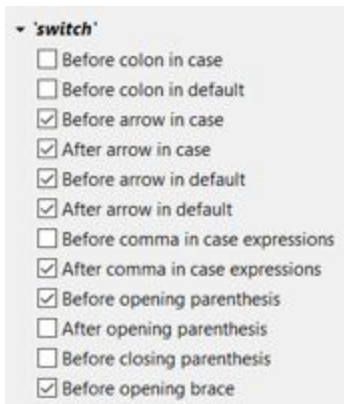
The Eclipse IDE provides many quick fixes, quick assists, templates, and tooling features to help you write code with switch improvements. For example, there are quick fixes (Ctrl + 1) to add the 'default' case or the missing case statements in a switch expression where the proposals are inserted in linked mode so you can quickly replace them with the required values.



A quick assist (Ctrl + 1) allows you to split multiple labels in a single case statement so you can provide separate case values when needed.



In addition, the formatter profile, shown below, has new settings that allow you to control spaces at various locations in switch expressions.



Using instanceof Pattern Matching

Java 14 introduces pattern matching for `instanceof` as a preview feature that provides a pattern variable with the `instanceof` operator to simplify the code by reducing explicit casts.

The Eclipse IDE understands the type and scope of the `pattern` variable, allowing you to perform actions, such as invoking the content assist (Ctrl + Space) and renaming the pattern variable, as shown below.



```
public int size(Object obj) {  
    if (obj instanceof String str) {  
        return str.trim().length();  
    }  
    if (obj instanceof List l) {  
        return l.size();  
    }  
    return -1;  
}
```

```
public int size(Object obj) {  
    if (obj instanceof String s) {  
        return s.trim().length();  
    }  
    if (obj instanceof List<?> l) {  
        return l.size();  
    }  
    return -1;  
}
```