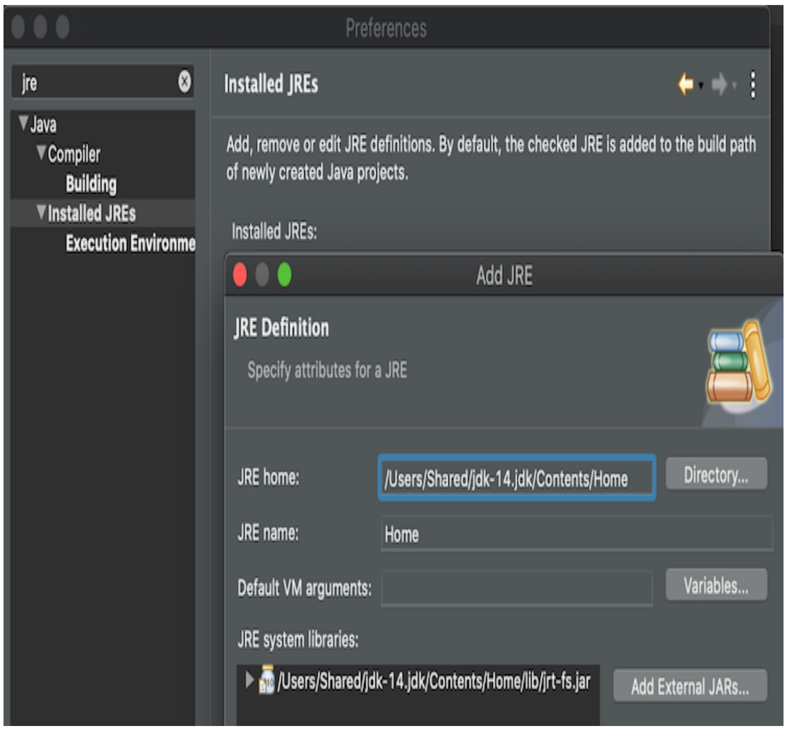
Java14/Examples

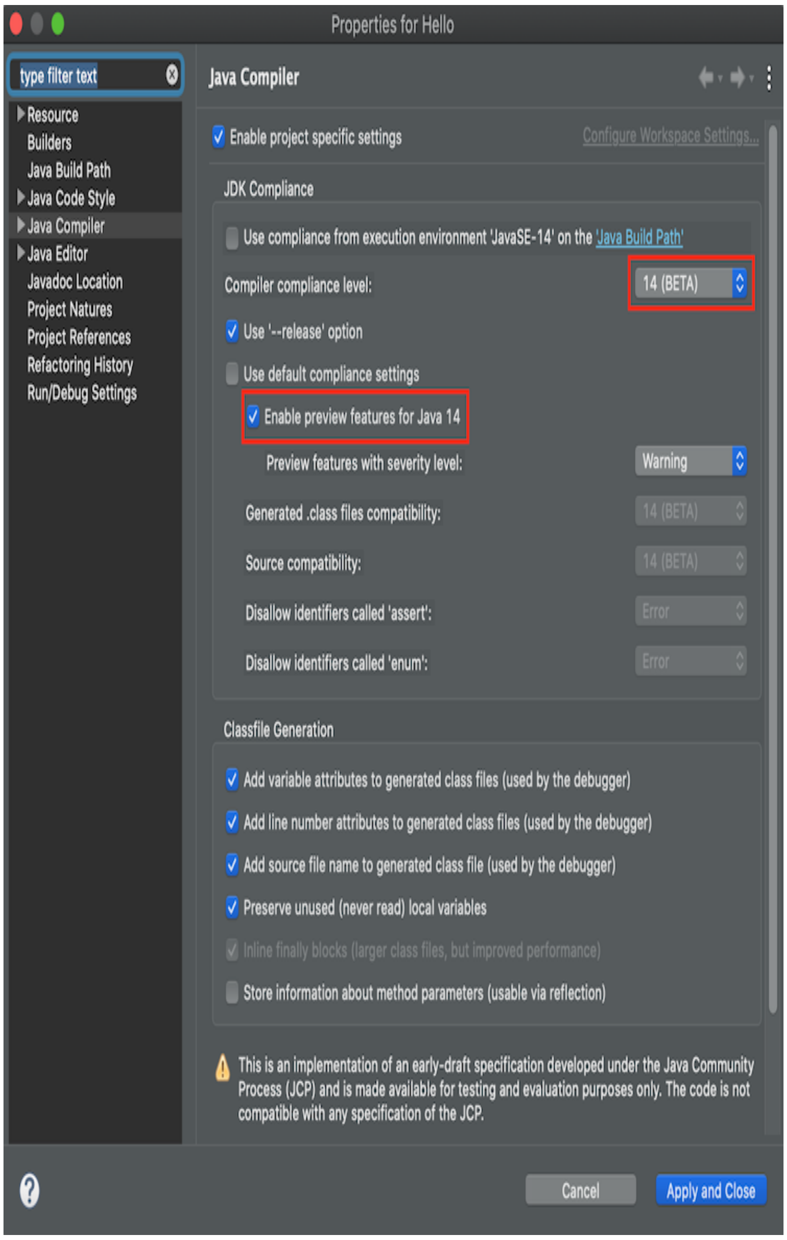
This is an informal page listing examples of features that are implemented by the Java 14 Support, which can be installed from the Marketplace (https://marketplace.eclipse.org/content/java-14-support-eclipse-2020-03-415). You are welcome to try out these examples. If you find bugs, please file a bug after checking for a duplicate entry here (https://bit.ly/2TRS4CO)

Watch out for additional examples being added soon.

NOTE:

- Switch expression, Enhanced switch statement and Multi-constant case labels are **standard features** in Java 14.
- TextBlock is also another preview feature in Java 14. They are not enabled by default and can by enabled using **--enable-preview**.
- Records is also another preview feature in Java 14. They are not enabled by default and can by enabled using **--enable-preview**.
- Pattern instanceof is also another preview feature in Java 14. They are not enabled by default and can by enabled using **--enable-preview**.
- In Eclipse, **--enable-preview** can be enabled from the Preferences. It is implicitly added while launching a java program if the feature has been enabled for the project/workspace.

	Feature / Steps	**Expected Result**
The Pre-requisite: Java 14 JRE Support		
Add Java 14 JRE	Use Window -> Preferences-> Java -> Installed JREs -> Add...  Preferences jre Java Compiler Building Installed JREs Execution Environme Installed JREs Installed JREs: Add JRE JRE Definition Directory... Variables... Add External JARs... Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects. Specify attributes for a JRE JRE home: /Users/Shared/jdk-14.jdk/Contents/Home JRE name: Home Default VM arguments: JRE system libraries: /Users/Shared/jdk-14.jdk/Contents/Home/lib/jrt-fs.jar (/File:FileAddJ14.png) [note: Eclipse -> Preferences in Mac / Window -> Preferences in Windows]	Java 14 JRE recognized as a valid JRE
Project JRE	In Package Explorer Use project's context menu and add Java 14 JRE	JRE specific (eg Object) gets resolved in the project.
Package Explorer	Go to Package Explorer and expand the Java 14 JRE	Modules (eg java.base etc) are listed in the package explorer view
The First Step: Java 14 Compliance		
Set Project Compliance in Package Explorer	Context Menu of Project -> Properties -> Set project-specific, drop down to 14	14 is shown in the drop down list. A checkbox to enable preview features is available on the preference page.



Properties for Hello

type filter text

- Resource
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project Natures
- Project References
- Refactoring History
- Run/Debug Settings

Java Compiler

✓ Enable project specific settings — Configure Workspace Settings...

JDK Compliance

☐ Use compliance from execution environment 'JavaSE-14' on the 'Java Build Path'

Compiler compliance level: 14 (BETA)

✓ Use '--release' option

☐ Use default compliance settings

✓ Enable preview features for Java 14

Preview features with severity level: Warning

Generated .class files compatibility: 14 (BETA)

Source compatibility: 14 (BETA)

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

✓ Add variable attributes to generated class files (used by the debugger)

✓ Add line number attributes to generated class files (used by the debugger)

✓ Add source file name to generated class file (used by the debugger)

✓ Preserve unused (never read) local variables

✓ Inline finally blocks (larger class files, but improved performance)

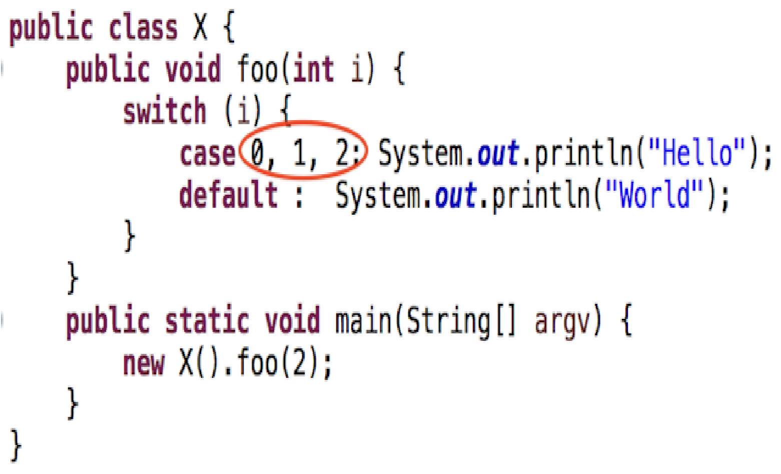
☐ Store information about method parameters (usable via reflection)

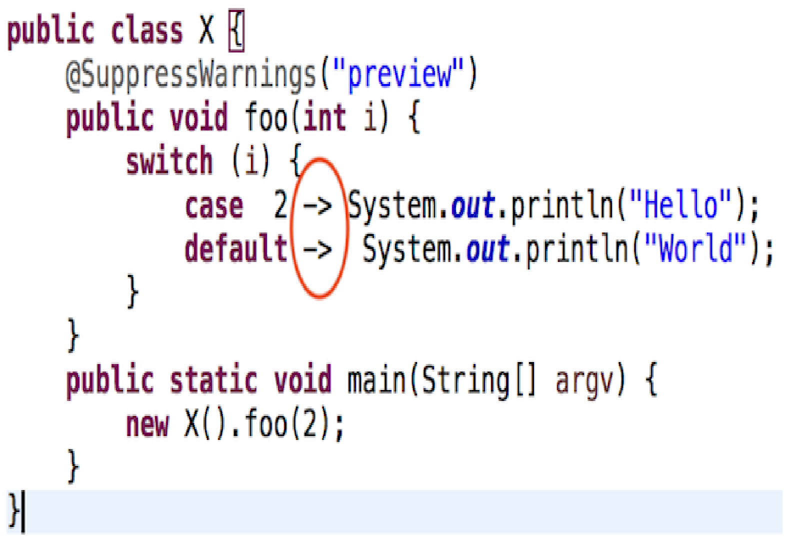
This is an implementation of an early-draft specification developed under the Java Community Process (JCP) and is made available for testing and evaluation purposes only. The code is not compatible with any specification of the JCP.

Cancel | Apply and Close

(/File:J14.compliance.png)

Standard Feature: Switch Expressions, Enhanced Switch Statement and Multi-Label Case Statements.

Positive Compilation 1 (Switch Statement with multi-label case with colon)	Use the following code:
	`public class X {`
	`    public void foo(int i) {`
	`        switch (i) {`
	`                case 0, 1, 2: System.out.println("Hello");`
	`                default :  System.out.println("World");`
	`        }`
	`    }`
	`    public static void main(String[] argv) {`
	`        new X().foo(2);`
	`    }`
	`}`
	
	`public class X {`
	`    public void foo(int i) {`
	`        switch (i) {`
	`            case 0, 1, 2: System.out.println("Hello");`
	`            default :  System.out.println("World");`
	`        }`
	`    }`
	`    public static void main(String[] argv) {`
	`        new X().foo(2);`
	`    }`
	`}`
	(/File:Switch-statement-multi.png)
		Code compiles and while running prints both "Hello" "World"

Positive Compilation 2 (Switch Statement with case with arrow)	Use the following code:
	`public class X {`
	`        public void foo(int i) {`
	`        switch (i) {`
	`                case  2 -> System.out.println("Hello");`
	`                default ->  System.out.println("World");`
	`        }`
	`    }`
	`    public static void main(String[] argv) {`
	`        new X().foo(2);`
	`    }`
	`}`
	
	`public class X {`
	`    @SuppressWarnings("preview")`
	`    public void foo(int i) {`
	`        switch (i) {`
	`            case  2 -> System.out.println("Hello");`
	`            default -> System.out.println("World");`
	`        }`
	`    }`
	`    public static void main(String[] argv) {`
	`        new X().foo(2);`
	`    }`
	`}`
	(/File:Switch-statement-arrow.png)
		Code compiles and while running prints only "Hello" (because a break is implicit after every case with an arrow.

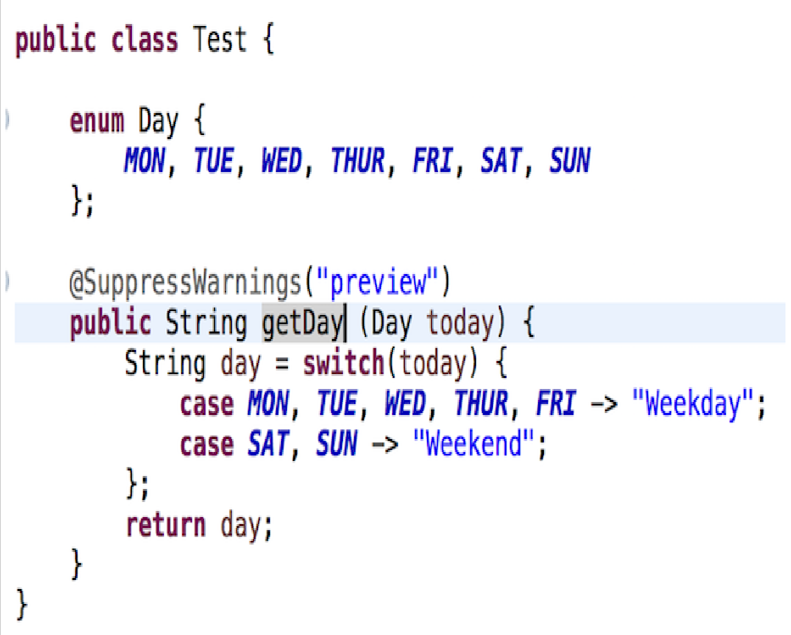
Positive Compilation (Switch Expression)

Use the following code:

```
public class Test {

        enum Day {
                MON, TUE, WED, THUR, FRI, SAT, SUN
        };

        public String getDay_1 (Day today) {
                String day = switch(today) {
                        case MON, TUE, WED, THUR, FRI -> "Weekday";
                        case SAT, SUN -> "Weekend";
                };
                return day;
        }
}
```



```
public class Test {

    enum Day {
        MON, TUE, WED, THUR, FRI, SAT, SUN
    };

    @SuppressWarnings("preview")
    public String getDay (Day today) {
        String day = switch(today) {
            case MON, TUE, WED, THUR, FRI -> "Weekday";
            case SAT, SUN -> "Weekend";
        };
        return day;
    }
}
```

(/File:Switch-exp.compile.png)

Code compiles

Preview Feature: Records

Postive compilation1 (Record Example)

Compile and run the following code:

```
@SuppressWarnings("preview")
record Point(int x, int y) {
}
public class X1 {
        public static void main(String[] args) {
                Point p = new Point(100, 200);
                System.out.println(p.x());
        }
}
```

Code compiles and prints 100.

Positive compilation2 (Nested Record Example)

Compile and run the following code:

```
class X2 {
        public static void main(String[] args) {
                System.out.println(0);
        }
        @SuppressWarnings("preview")
        record Point(int x, int y) {
        }
}
```

Code compiles and prints 0.

Positive compilation3 (Record Example)

Compile and run the following code:

```
class X3 {
        public static void main(String[] args) {
                System.out.println(0);
        }
}
@SuppressWarnings("preview")
final record Point(int x, int y) {
}
```

Code compiles and prints 0. Though a record declaration is implicitly final, it is permitted for the declaration of a record type to redundantly specify the final modifier

Positive compilation4	Compile and run the following code:
	`@SuppressWarnings("preview")`
	`record R() {`
	`}`
	`class X4 {`
	`        public static void main(String[] args) {`
	`                System.out.println(new R().hashCode());`
	`        }`
	`}`
		Code compiles and prints 0.

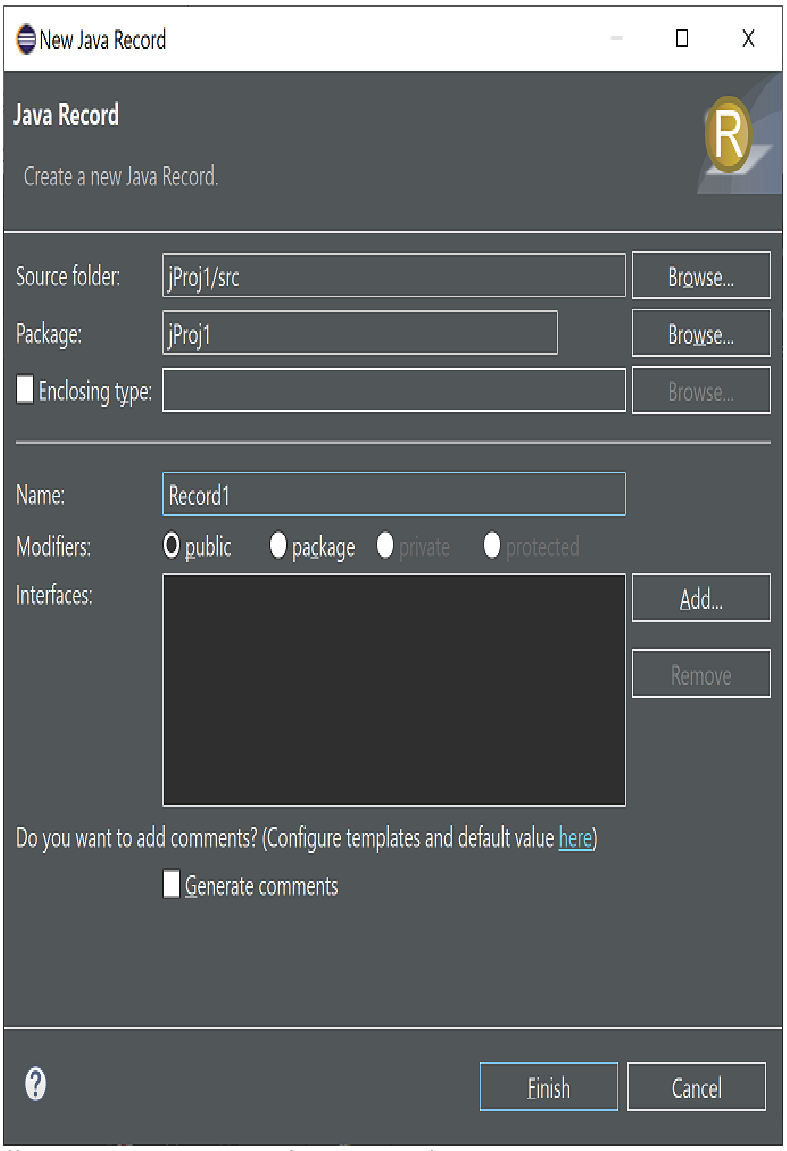
Positive compilation5	Compile and run the following code:
	`import java.lang.annotation.Target;`
	`import java.lang.annotation.ElementType;`
	`@Target({ ElementType.PARAMETER })`
	`@interface MyAnnot {`
	`}`
	`@SuppressWarnings("preview")`
	`record R(@MyAnnot()int i, int j) {`
	`}`
	`class X5 {`
	`        public static void main(String[] args) {`
	`                System.out.println(new R(100, 200).hashCode() != 0);`
	`        }`
	`}`
		Code compiles and prints true.

Positive compilation6	Compile and run the following code:
	`class X6 {`
	`        @SuppressWarnings("preview")`
	`        public static void main(String[] args) {`
	`                record R(int x,int y){}`
	`                R r = new R(100, 200);`
	`                System.out.println(r.x());`
	`        }`
	`}`
		Code compiles and prints 100.

Negative compilation1 (Record Example)	Compile and run the following code:
	`@SuppressWarnings("preview")`
	`abstract record Point(int x, int y){`
	`}`
	`class X7 {`
	`        public static void main(String[] args){`
	`                System.out.println(0);`
	`        }`
	`}`
		Code fails to compile with error "Illegal modifier for the record Point; only public, final and strictfp are permitted"

Negative compilation2 (Record Example)	Compile and run the following code:
	`@SuppressWarnings("preview")`
	`record Point1(int myInt, char myChar) implements I {`
	`        public Point1 {`
	`                this.myInt = myInt;`
	`                this.myChar = myChar;`
	`        }`
	`}`
	`public class X8 {`
	`        public static void main(String[] args) {`
	`                System.out.println(0);`
	`        }`
	`}`
	`interface I {`
	`}`
		Code fails to compile with error "The canonical constructor Point1 of a record declaration must be declared public."

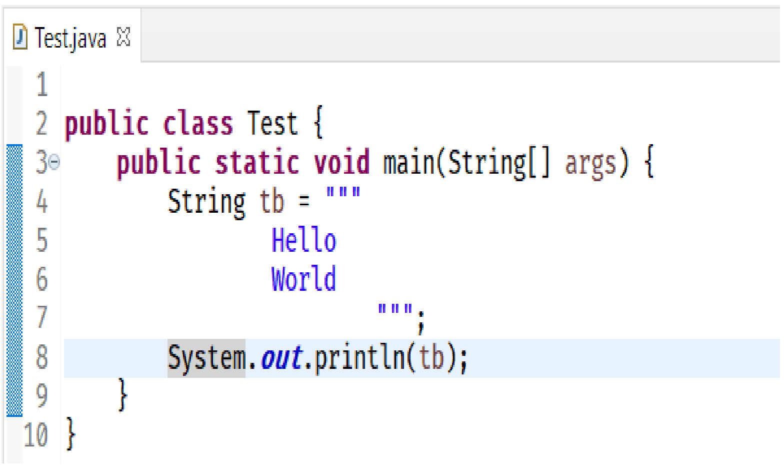
Negative compilation3 (Record Example)	Compile and run the following code:
	`class record {`
	`        public static void main(String[] args) {`
	`                System.out.println(0);`
	`        }`
	`}`
		Code fails to compile with error "Record is a restricted identifier and hence not a valid type name"

Record Creation Wizard	Right Click on the Project -> New -> Record **or** Right Click on the Project -> New -> Other and search for Record **or** Right Click on the Project -> New -> Other -> Java -> Record  New Java Record **Java Record** Create a new Java Record. Source folder: jProj1/src Package: jProj1 ☐ Enclosing type: Name: Record1 Modifiers: [✓ public] package private protected Interfaces: Do you want to add comments? (Configure templates and default value here) ☐ Generate comments Browse... Browse... Browse... Add... Remove Finish Cancel (/File:FileAddJ14RecordCreation.png) [note: In older workspaces Record option may not appear diectly under New menu in java perspective. To resolve this, use a new workspace or relaunch eclipse with -clearPersistedState for the same workspace]	Record is created
Preview Feature: Text Blocks		

Text Block Example

Compile and run the following code:

```
@SuppressWarnings("preview")
public class Test {
        public static void main(String[] args) {
                String tb = """
                Hello
                World
                                """;
                System.out.println(tb);
        }
}
```



Test.java

```
public class Test {
    public static void main(String[] args) {
        String tb = """
                Hello
                World
                     """;
        System.out.println(tb);
    }
}
```

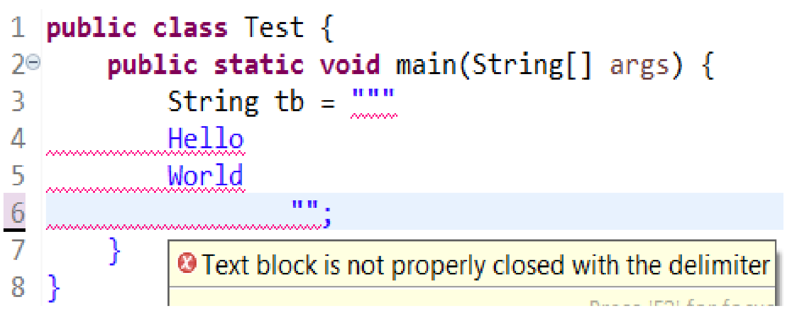
(/File:Textblock.png)

Code compiles and prints both "Hello" "World" as it is - notice that "World" is printed in the next line.

Text Block Compilation Error Example

Use the following code:

```
public class Test {
        public static void main(String[] args) {
                String tb = """
                Hello
                World
                                "";
        }
}
```



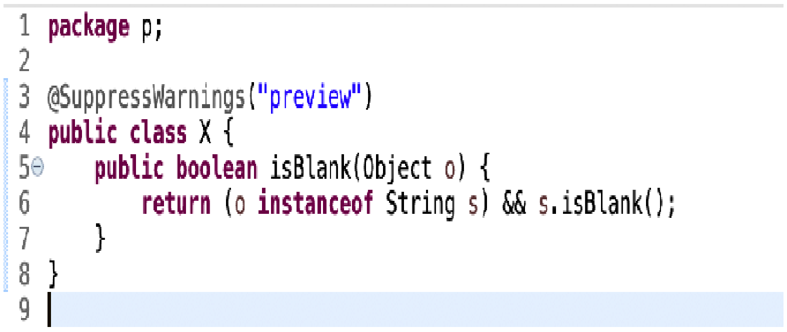
```
public class Test {
    public static void main(String[] args) {
        String tb = """
        Hello
        World
                "";
    }
}
```

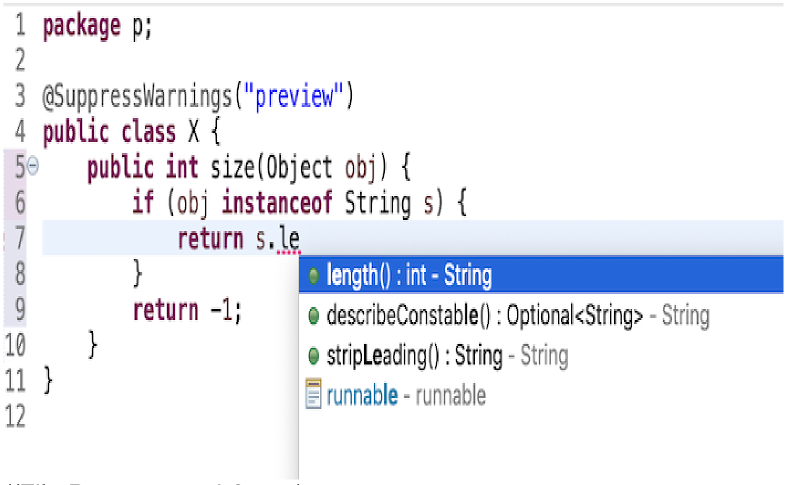
Text block is not properly closed with the delimiter

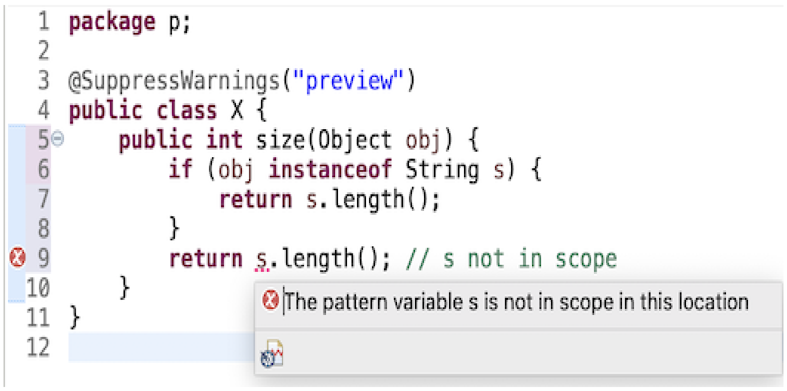
(/File:Textblock.error.png)

Compilation error - text block not closed properly

Preview Feature: Instanceof Pattern Matching

Instanceof Pattern Matching Example	Use the following code:
	`@SuppressWarnings("preview")`
	`public class X {`
	`        public boolean isBlank(Object o) {`
	`                return (o instanceof String s) && s.isBlank();`
	`        }`
	`}`
	
	`package p;`
	`@SuppressWarnings("preview")`
	`public class X {`
	`    public boolean isBlank(Object o) {`
	`        return (o instanceof String s) && s.isBlank();`
	`    }`
	`}`
	(/File:Pattern-match1.png)
		The pattern variable 's' is in current scope

Instanceof Pattern Matching Example	Use the following code:
	`@SuppressWarnings("preview")`
	`public class X {`
	`        public int size(Object obj) {`
	`                if (obj instanceof String s) {`
	`                        return s.le`
	`                }`
	`                return -1;`
	`        }`
	`}`
	
	`package p;`
	`@SuppressWarnings("preview")`
	`public class X {`
	`    public int size(Object obj) {`
	`        if (obj instanceof String s) {`
	`            return s.le`
	`        }`
	`        return -1;`
	`    }`
	`}`
	length() : int - String
	describeConstable() : Optional<String> - String
	stripLeading() : String - String
	runnable - runnable
	(/File:Pattern-match2.png)
		The pattern variable 's' is in current scope inside 'then' statement and completion proposes applicable methods on String, the pattern matched type.

Instanceof Pattern Matching Example	Use the following code:
	`package p;`
	
	`@SuppressWarnings("preview")`
	`public class X {`
	`        public int size(Object obj) {`
	`                if (obj instanceof String s) {`
	`                        return s.length();`
	`                }`
	`                return s.length(); // s not in scope`
	`        }`
	`}`
	
	
	`package p;`
	
	`@SuppressWarnings("preview")`
	`public class X {`
	`    public int size(Object obj) {`
	`        if (obj instanceof String s) {`
	`            return s.length();`
	`        }`
	`        return s.length(); // s not in scope`
	`    }`
	`}`
	The pattern variable s is not in scope in this location
	(/File:Pattern-match3.png)
		The pattern variable 's' is rejected by the compiler when not in scope outside the 'then' statement.