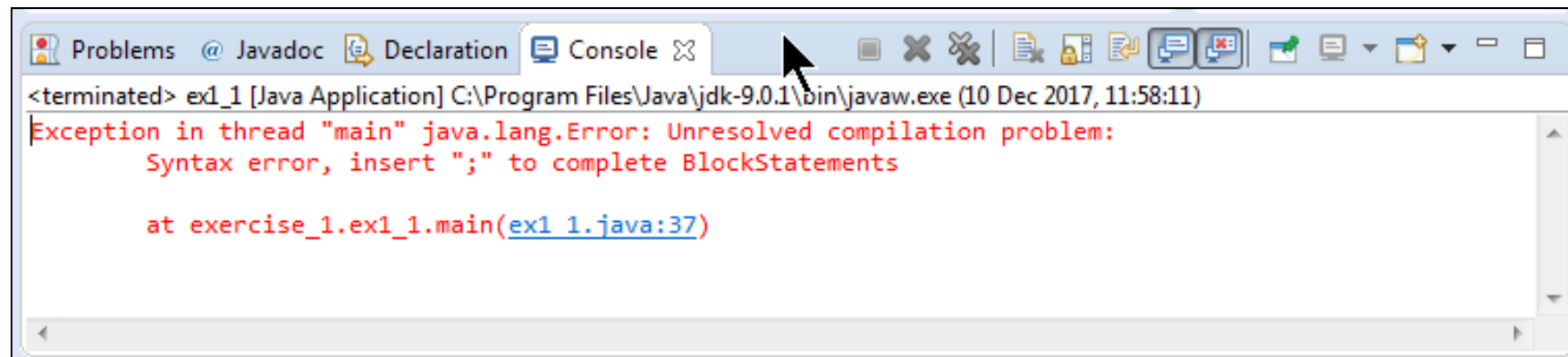


Using the Java Editor – Error Checking



Adding a semi-colon will terminate the command and clear the red cross.

If the Developer chooses to ignore the error and attempt to run the code, the error will appear in the Console area and the program will not run ...



The screenshot shows the Eclipse IDE's Console window. The title bar indicates the application is terminated. The console output shows a red error message: "Exception in thread 'main' java.lang.Error: Unresolved compilation problem: Syntax error, insert ';' to complete BlockStatements". The error occurred in the file "exercise_1.ex1_1.java" at line 37.

```
<terminated> ex1_1 [Java Application] C:\Program Files\Java\jdk-9.0.1\bin\javaw.exe (10 Dec 2017, 11:58:11)
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error, insert ";" to complete BlockStatements

    at exercise_1.ex1_1.main(ex1_1.java:37)
```

Using the Java Editor – Colour Coding



The Java Editor will highlight commented out code in a light blue colour (when using `/**`, single line comments are green) ...

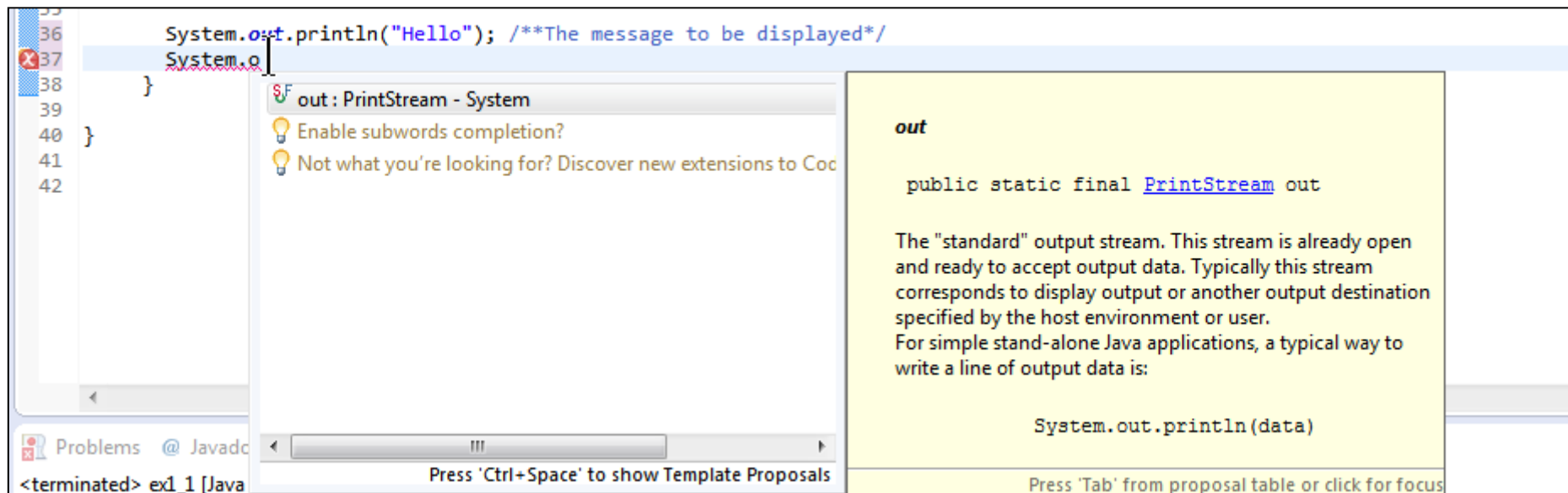
```
16  /**
17  * @param args
18  * @author Tom
19  * @throws ArithmeticException
20  */
21  public static void main (int[] args) throws ArithmeticException
22  {
23
24      try {
25  |   char sex = 'F';
26
27
28      if (sex != 'F')
29
30          throw new RuntimeException("Sex value is incorrect");
31
```

A dark blue is used for Strings and characters, while reserved words are displayed in black.



Using the Java Editor – Hints while typing

The Java Editor will detect familiar patterns when the Developer is typing and will open up a window with hints for completion, it will also report the line as an error until complete !! ...





Using the Java Editor – Hints while typing

Note that the closer you get to a completed class call the more comprehensive the information on the right hand side.

The screenshot displays two instances of the Eclipse IDE's Java editor, illustrating how the information provided in the completion hints becomes more detailed as the user types more of the method name.

Top Screenshot: The cursor is positioned at the end of the line `System.out.`. The completion list on the right shows several methods from the `PrintStream` class, including `append(char c)`, `append(CharSequence csq)`, `append(CharSequence csq, int start, int end)`, `checkError()`, `close()`, `equals(Object obj)`, `flush()`, `format(String format, Object... args)`, and `format(Locale l, String format, Object... args)`. The hint for `append(char c)` is expanded, showing its signature `append(char c) : PrintStream - PrintStream`, a description: "Appends the specified character to this output stream.", and a note: "An invocation of this method of the form `out.append(c)` behaves in exactly the same way as the invocation `out.print(c)`". It also states "Specified by: [append\(...\)](#) in [Appendable](#)".

Bottom Screenshot: The cursor is positioned at the end of the line `System.out.println`. The completion list shows various `println` methods: `println()`, `println(boolean x)`, `println(char x)`, `println(char[] x)`, `println(double x)`, `println(float x)`, `println(int x)`, `println(long x)`, and `println(Object x)`. The hint for `println()` is expanded, showing its signature `println() : void - PrintStream` and a description: "Terminates the current line by writing the line separator string. The line separator string is defined by the system property `line.separator`, and is not necessarily a single newline character (`'\n'`).". At the bottom of the hint, it says "Press 'Ctrl+Space' to show Template Proposals" and "Press 'Tab' from proposal table or click for focus".



Using the Java Editor – Bracket Location

Java code is very dependant on brackets, braces and parentheses being matched, that is, an opened brace must have a closing brace, the editor in Eclipse will match the brace when clicked on

...

```
public static void main(String[] arguments)
{
    System.out.println("Hello");
}
```

```
public static void main(String[] arguments)
{
    System.out.println("Hello");
}
```