# Compiler errors

1. **Save all files** (CTRL + SHIFT + S).

2. Start from top of the file, and **search down** for the first compiler error.
   Look for a red underline in the code.
   You can also look for a red X in the left margin, or a red rectangle in the right margin.

3. **Hover your mouse** over the red underline. This should display a message that describes/explains the error, and it will usually provide several "Quick fixes".

4. **Read the error message**. Try to figure out what caused the problem.

5. If you can't find it, **read the different available quick fixes** as ideas for possible solutions.

6. If you still can't find it, **try looking in the code above the error**. Sometimes syntax errors can confuse the compiler and cause compiler errors to appear after the actual source of the problem.

7. **Fix the problem manually**, or use one of the **automatic quick fixes**.
   (Don't click a quick fix until you are certain it is actually what you want to do.)

8. **Continue from 1**, until all compiler errors in the file are fixed.
   (Don't forget to save after fixing each new error!)

# Common compiler errors

- Syntax error on token "*x*", *y* expected
    - You have broken the rules of syntax. The compiler expects to see "y" after a certain element in the code, but doesn't.
- Syntax error: insert "…" to complete …
    - You have not correctly completed a statement or declaration.
      Insert the specified character to complete it.
      Example: insert ";" to complete a statement.
- Illegal modifier
    - You used an invalid modifier keyword in your declaration. Remove the invalid modifier.
- *TypeName* cannot be resolved to a type
    - If the type (= class/interface/enum) already exists:
        - You forgot to import the type (if it is in another package)
        - You did import the type (if necessary), but made a spelling mistake when using it
    - If the type doesn't exist yet:
        - You forgot to create it
- *variableName* cannot be resolved to a variable
    - You forgot to declare the variable before using it
    - You did declare the variable, but made a spelling mistake when using it
- The method *methodName(…)* is undefined for the type *Type*
    - You forgot to define the method before using it
    - You forgot to define the method with the right parameter list, before using it
    - You did define the method, but made a spelling mistake when using it

# Runtime errors

1. **Read the first line** of the red stack trace error message in the console. This will tell you the type of exception/error, and possibly give you a short message or explanation.

2. Once you know what type of exception you are dealing with, then scroll all the way to the bottom.

3. From the bottom (beginning), **search up through each line**, until you find a line from your own code.

   Each line is a method in the call stack. Each of these methods was called by the method directly below it, in a sequence moving from the bottom to the top.

   Sometimes you'll have to scroll up through methods from a library you're using but didn't write.

4. **For each line** of your own code that you find, click on the **blue link**.
   This will take you to that line of code in the editor, which typically contains a call to the next method.
   Repeating this process will trace the path taken through your code to arrive at the error.

5. Typically, the **top (last) line** of your own code in the stack trace will contain the **source of the problem**.

6. If you see chained exceptions (one stack trace, followed by "Caused by:" and another stack trace, etc.), then continue this same process from the bottom all the way to the top.

7. **Fix the problem**. **Test your program again**, and try to reproduce the error to prove that you fixed it.

# Common runtime errors

## NullPointerException

You are using a variable that *should* refer to an object (you are trying to use/change an object), but the variable is *actually* null (the reference variables' value is null, instead of an object reference).

You must assign the variable an object reference first, before using it.

## IndexOutOfBoundsException / ArrayIndexOutOfBoundsException

You tried to access an array index (or collection index) that is invalid (index < 0 or index >= size).

## ClassCastException

You tried to cast an object reference value to an invalid type. (Cast = convert value from one type to another.)
For example, you tried to downcast an object reference to a type lower than the object's type itself.

## NumberFormatException

You provided a String that should represent a certain type of number, but is not formatted correctly for a number of that type. That is, it cannot be parsed. (Parsed = converted from a string to a number.)

## OutOfMemoryError

Fatal JVM (Java Virtual Machine) error. You have run out of memory. Normal operation cannot continue.

## StackOverflowError

Fatal JVM error. The thread's call stack has run out of space: too many methods have been called.
This can be caused by accidental/improper recursion. For example, a method calls itself infinitely.