## Types of Java Program Errors

Program errors are also referred to as program bugs.

A Java program may have one or more of three types of errors:

* Syntax errors (Compiler errors or Compile-time errors)
* Runtime errors
* Logic errors

Usually, the errors become more difficult to find and fix as you move down the above list.

**Syntax errors**

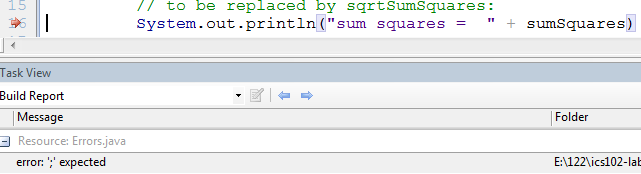
Syntax errors represent *grammar errors* in the use of the programming language.  Common examples are:

* Misspelled variable and method names
* Missing semicolons
* Unmatched parentheses, square brackets, and curly braces
* Unclosed String constants
* Using a variable that has not been declared
* Incorrect format in selection and loop statements

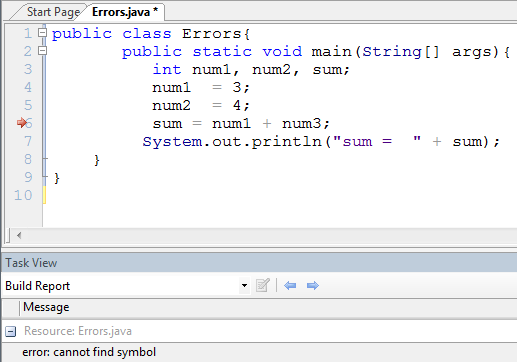
Syntax errors are the easiest to find and fix. Over the years, compiler developers have worked hard to make compilers smarter so that they can catch errors at compile time that might otherwise turn out to be runtime errors.

Examples of Syntax errors:

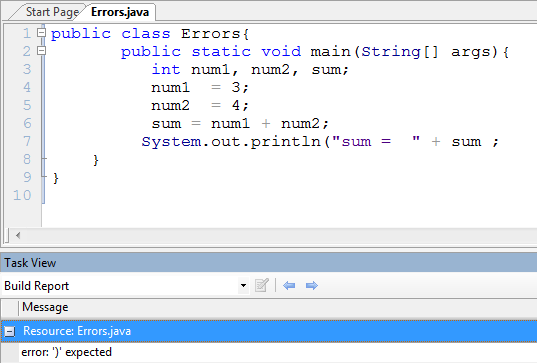
1. Simple statement not terminated by semicolon:



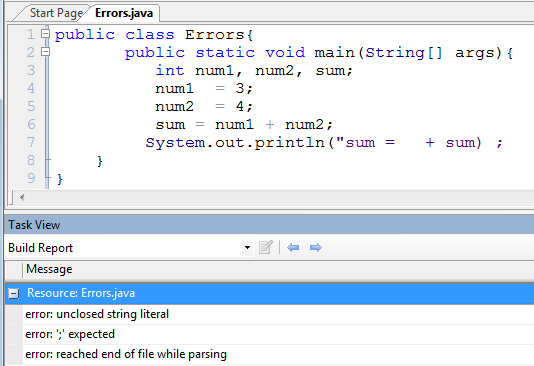
2. Using a variable before it is declared:



3. Missing closing parenthesis, ) :



4. Unclosed String constants (String literals):



**Runtime errors**

A type of error that occurs during the execution of a program is known as run-time error or Exception. Runtime errors (or Exceptions) may crash your program when you run it. Runtime errors occur when a program with no syntax errors directs the  [computer to execute an illegal operation. Common examples are:](http://www.blurtit.com/q947430.html" \l "#" \t "_top)

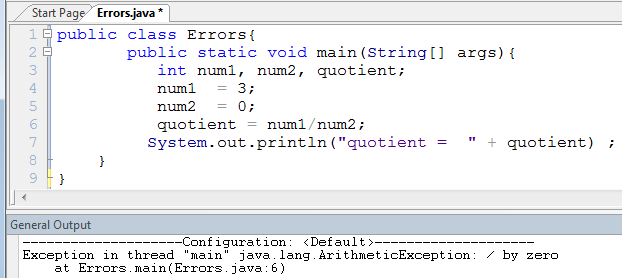
[grey_loader](http://www.blurtit.com/q947430.html" \l "#" \t "_top)

* Trying to perform integer division by a variable that contains a value of zero
* Trying to open a file that does not exist

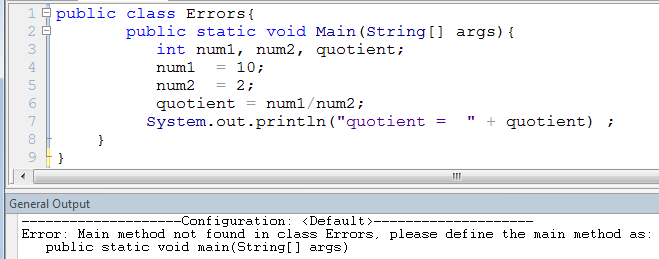
There is no way for the compiler to know about these kinds of errors when the program is compiled. Runtime errors are commonly due to wrong input from the user. Runtime errors are usually more difficult to find and fix than syntax errors.

To find the source of a run-time error in a program, usually a software called **debugger** is used.

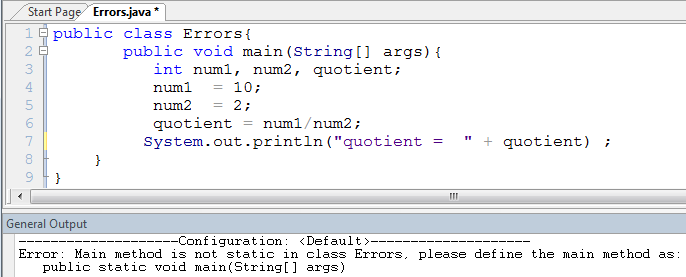
Example: When the following program is executed, a run-time error occurs due to division by zero in the expression **num1 / num2**



The following program has a run-time error because the **main** method is misspelled:



The following program has run-time error because the main method is not declared as static:



**Logic errors**

Logic errors occur when a programmer implements the algorithm for solving a problem incorrectly. A statement with logical error may produce unexpected and wrong results in the program. Common examples are:

* Multiplying when you should be dividing
* Adding when you should be subtracting
* Opening and using data from the wrong file
* Displaying the wrong message

Logic errors are the hardest to find and fix because:

* The compiler does not detect these errors
* There is no indication of error when the program is executed.
* The program may produce correct results for some input data and wrong results for other input data.

Logic errors can only be detected by examining the program thoroughly. This is usually done by using a **debugger**.

Example: The following program has a logic error in the **System.out.println( )** statement:

public class Errors{

public static void main(String[] args){

int num1 = 4, num2 = 9;

**System.out.println(num1 + " + " + num2 + " = " + (num1 \* num2));** }

}

Notice that the program produces correct results for some initializations (num1 = 2, num2 = 2 and num1 = 0, num2 = 0) and wrong results for others:

|  |  |
| --- | --- |
| 2 + 2 = 4 | 3 + 4 = 12 |

Since a Java program may have syntax, logic, or run-time errors, the edit-compile-execute cycle consists of the steps similar to those depicted in the diagram below:

