



Software Development Methods – Java – Part 1



Paolo Vercesi
Technical Program Manager





Hello, World!

The Java platform

Data types

Operators

Control structures

Hello, World!

HelloWorld.java

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        System.out.println("Hello world!");  
    }  
}
```

Launch Single-File Source-Code Programs

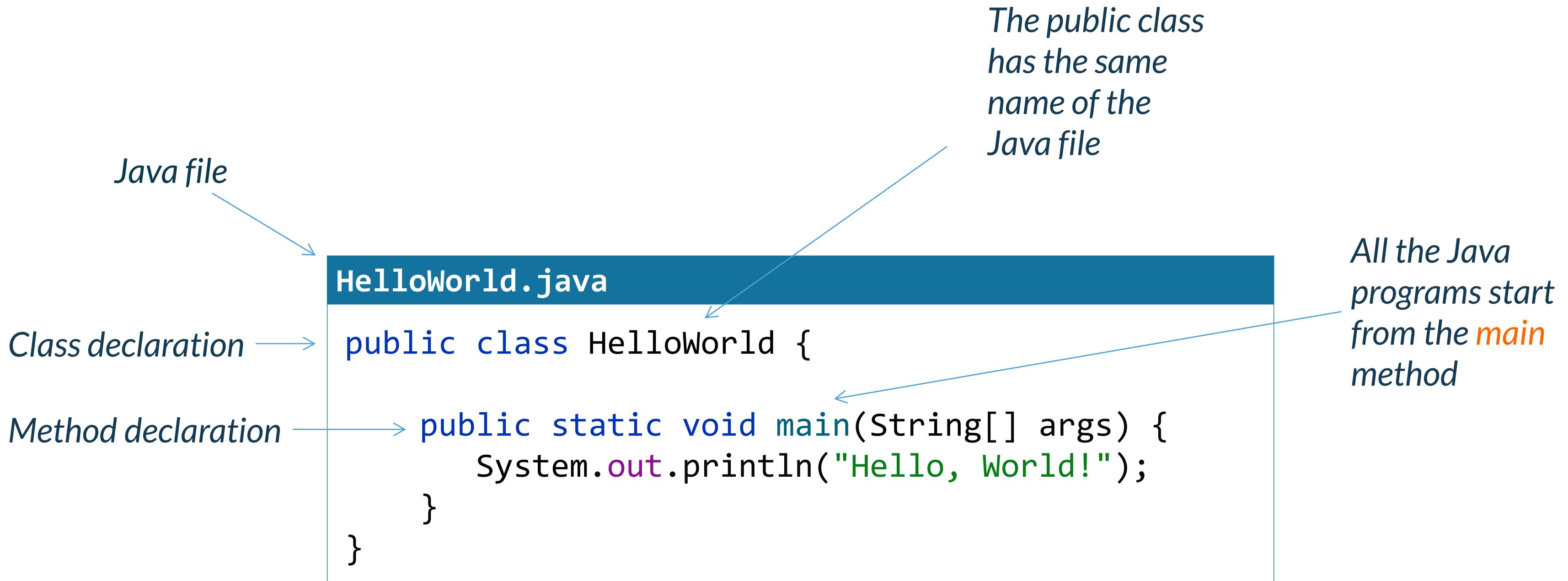
```
$ java HelloWorld.java  
Hello world!
```

Compile and Run

```
$ javac HelloWorld.java  
$ ls  
HelloWorld.class  HelloWorld.java  
$ java HelloWorld  
Hello world!
```



Hello, World! – Analysis of the program



Hello, World! - Compilation

The java compiler **javac** takes a list of source Java files and it compiles the corresponding class files

```
$ javac HelloWorld.java  
$ ls  
HelloWorld.class  HelloWorld.java  
$ java HelloWorld  
Hello, World!
```

A class file is compiled for each class defined in the source files

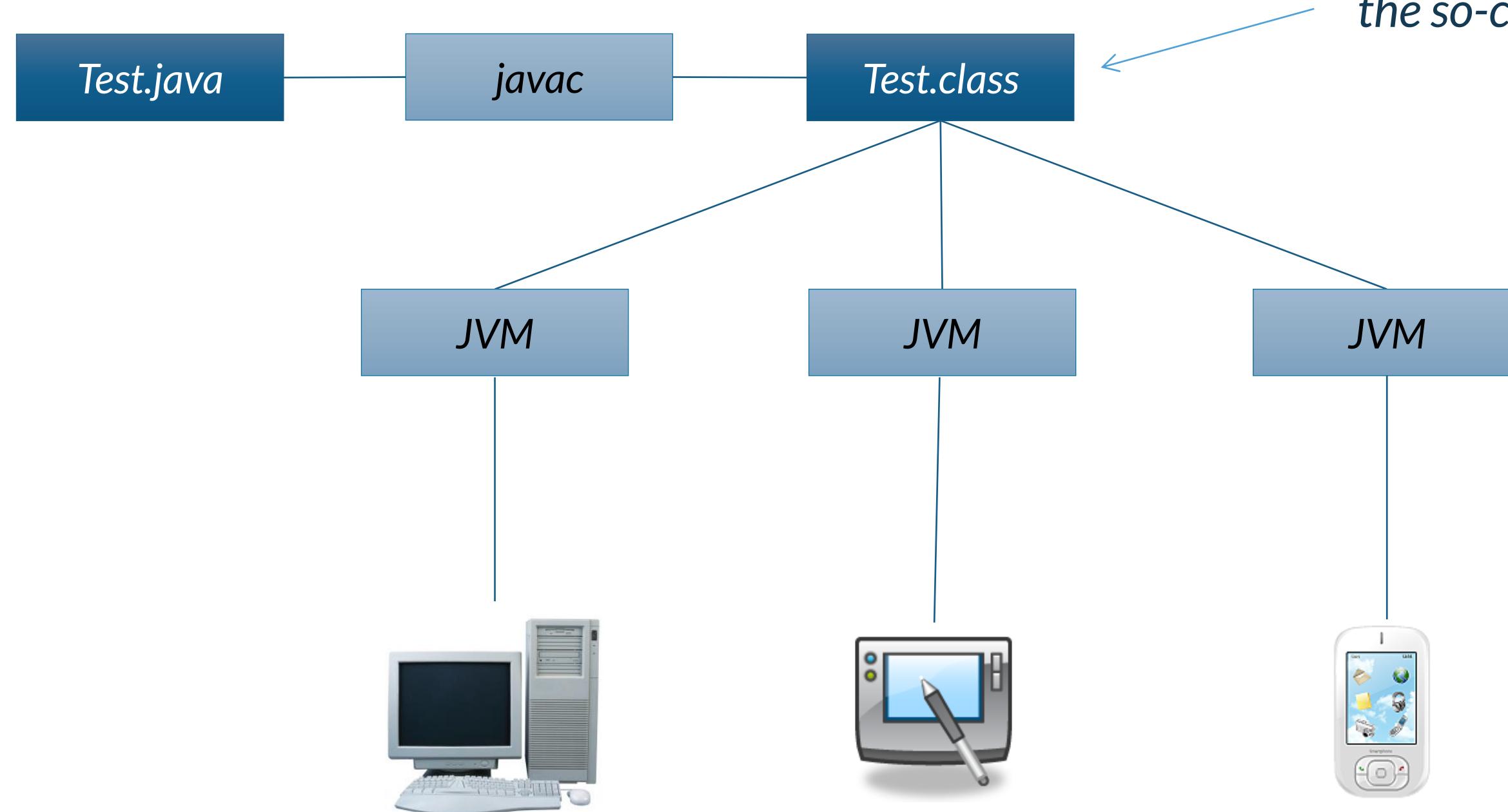
A java program is run invoking the java virtual machine (JVM) on the class containing the main method

Why do we need both **java** and **javac** to run a Java program?



The Java platform

*The output of the java compiler is not executable code but it is the so-called **bytecode***



*The compiled code is **independent** of the architecture of the computer*

Which Java?

- *The latest Java version is Java 17*
 - *released on 14 September 2021*
 - *Java releases follow a 6 months cycle*
 - *Java 17 is a Long Term Support (LTS) release*
 - *LTS are planned every 3 years*
- *There are many vendors*
 - *Oracle*
 - *Amazon*
 - *IBM*
 - *openJDK*



JRE or JDK

Java Runtime Environment (JRE)

*The **JRE** is the Java distribution that includes the JVM used to run Java programs*

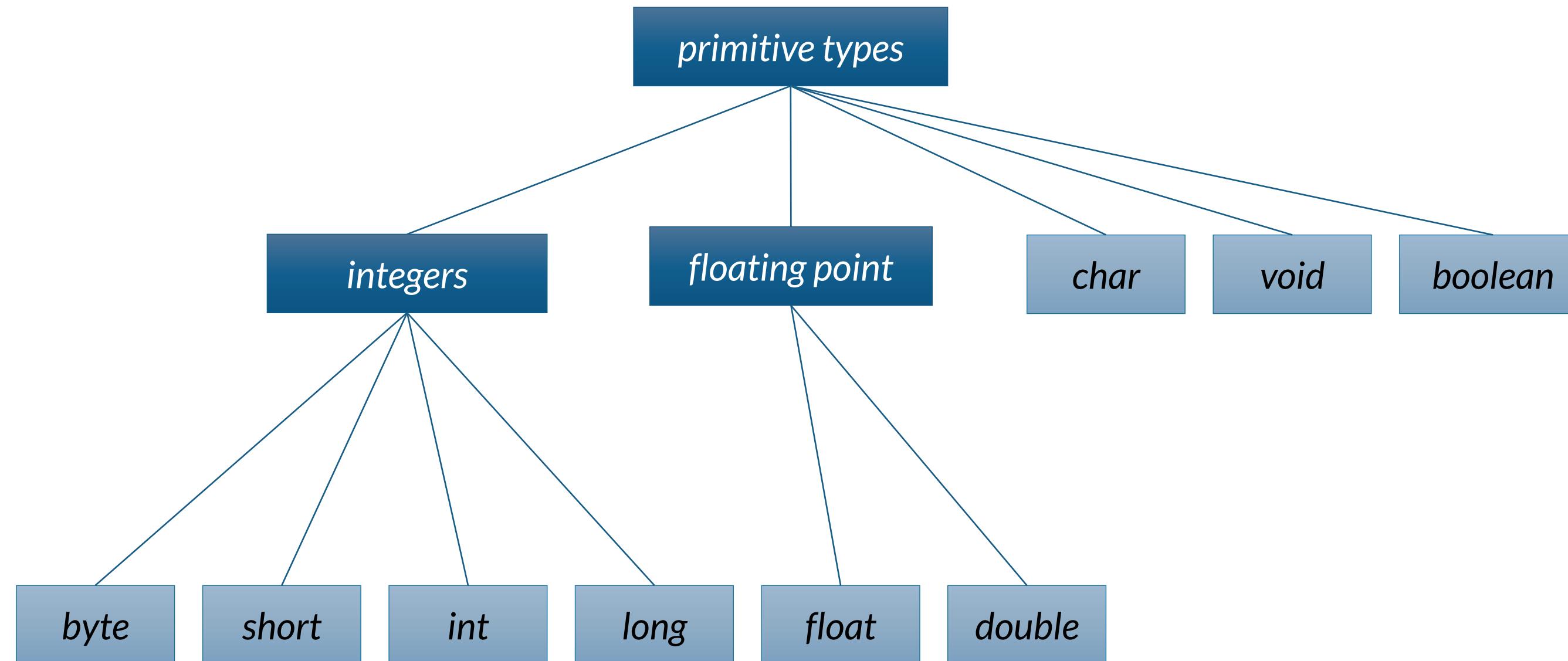
Java Development Kit (JDK)

*The **JDK** is the Java distribution that includes the compiler used to compile the Java files, it includes the JRE*



Primitive types

Java provides the following primitive types



Data type ranges

Type	Width	Range
byte	8	-128 to 127
short	16	-32,768 to 32767
int	32	-2,147,483,648 to 2,147,483,647
long	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
char	16	0 to 65535
float	32	1.4e-45 to 3.4e+38
double	64	4.9e-324 to 1.8e+308



Variable and constant definition

```
int x;  
double d = 0.33;  
float f = 0.22F;  
char c = 'a';  
boolean ready = true;  
  
x = 15;
```

Variables are declared *specifying* their type and name, and initialized in the point of declaration, or later with the assignment expression

Constants are declared with the word **final** in front. The specification of the initial value is compulsory

```
final double pi = 3.1415;  
final int maxSize = 100_000;  
final char lastLetter = 'z';
```

```
var f = 10.0; // a double variable  
var i = 50; // an int variable
```

Only **local variables** can be declared without an explicitly declared type by using the so-called **type inference**



Type conversion and casting

Java performs automatic conversions when there is no risk for data to be lost, **widening conversion**

- from **int** to **long**
- from **long** to **double**
- from **float** to **double**
- ...

When there is the risk for data to be lost, you must declare the explicit type conversion, **narrowing conversion or casting**

Conversion	Rules
from integer to integer (e.g., long to int)	Integer component is reduced modulo the target type size
from floating point to integer (e.g., double to int)	Fractional component is truncated Integer component is reduced modulo the target type size
from double to float	The number is rounded to the closest float, including +Infinity and -Infinity



Casting

To specify conversions where data can be lost it is necessary to use the *cast* operator.

TestCast.java

```
public class TestCast {  
    public static void main(String[] args) {  
  
        int a = 'x';                      // 'x' is a character  
        long b = 34;                      // 34 is an int  
        float c = 1002;                   // 1002 is an int  
        double d = 3.45F;                 // 3.45F is a float  
  
        long e = 34;  
        int f = (int) e;                  // e is a long  
        double g = 3.45;  
        float h = (float) g;              // g is a double  
    }  
}
```



Strings

Strings are not a basic type, but defined as a class, more details later!

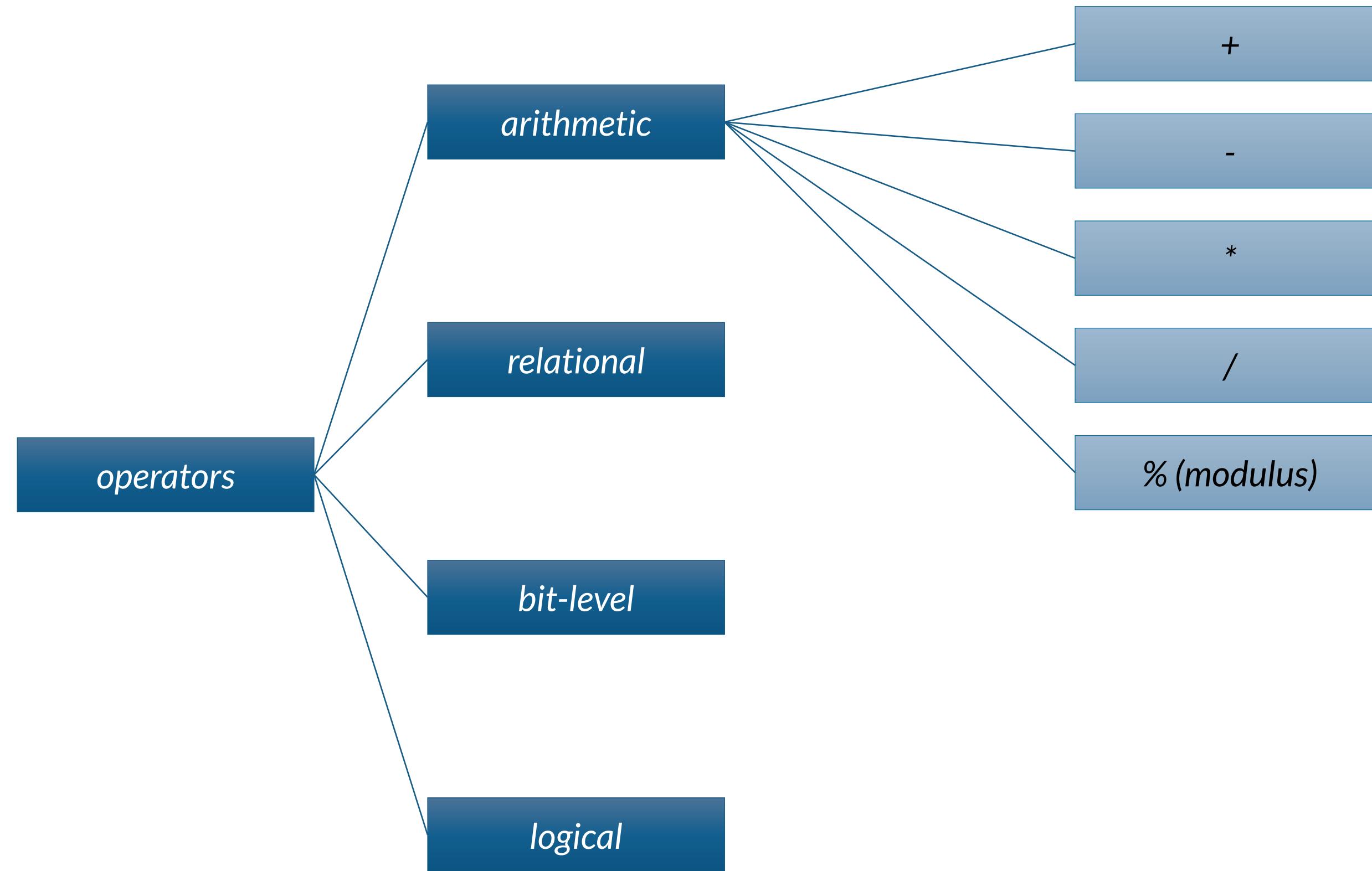
```
String a = "abc";
```

*If the expression begins with a string and uses the + operator, then
the next argument is converted to a string*

```
int cost = 22;  
String b = "the cost is " + cost + " euro";
```



Arithmetic operators



Type promotion

byte, **short** and **char** operands are always converted to **int** in arithmetic expressions

If an operand is a **long**, the whole expression is converted to **long**

If one operand is a **float**, the whole expression is converted to **float**

If one operand is a **double**, the whole expression is converted to **double**

```
byte b1 = 3;  
byte b2 = 4;  
byte b3 = b1 * b2; // Incompatible types  
byte b4 = (byte) (b1 * b2);
```

Can you explain this result?

```
double q = 3 / 2; // 1 !!!!
```



Example with arithmetic operators

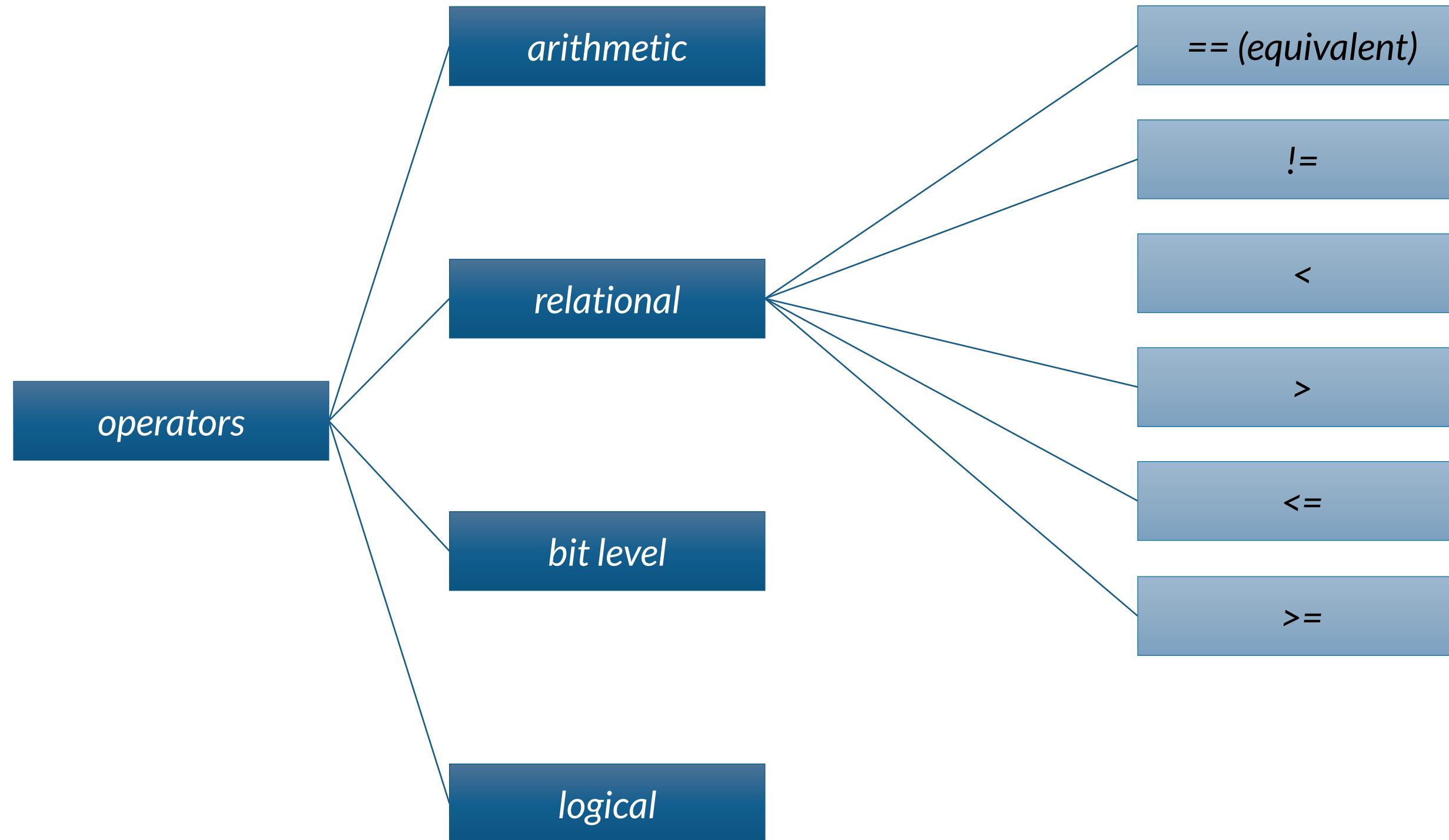
Arithmetic.java

```
public class Arithmetic {  
    public static void main(String[] args) {  
        int x = 12;  
        x += 5; // x = x + 5  
        System.out.println(x);  
  
        int a = 12, b = 12;  
        System.out.print(a++); // printed and then incremented  
        System.out.print(a);  
  
        System.out.print(++b); // incremented and then printed  
        System.out.println(b);  
    }  
}
```

```
$ java Arithmetic  
17  
12 13 13 13
```



Relational operators



Example with relational operators

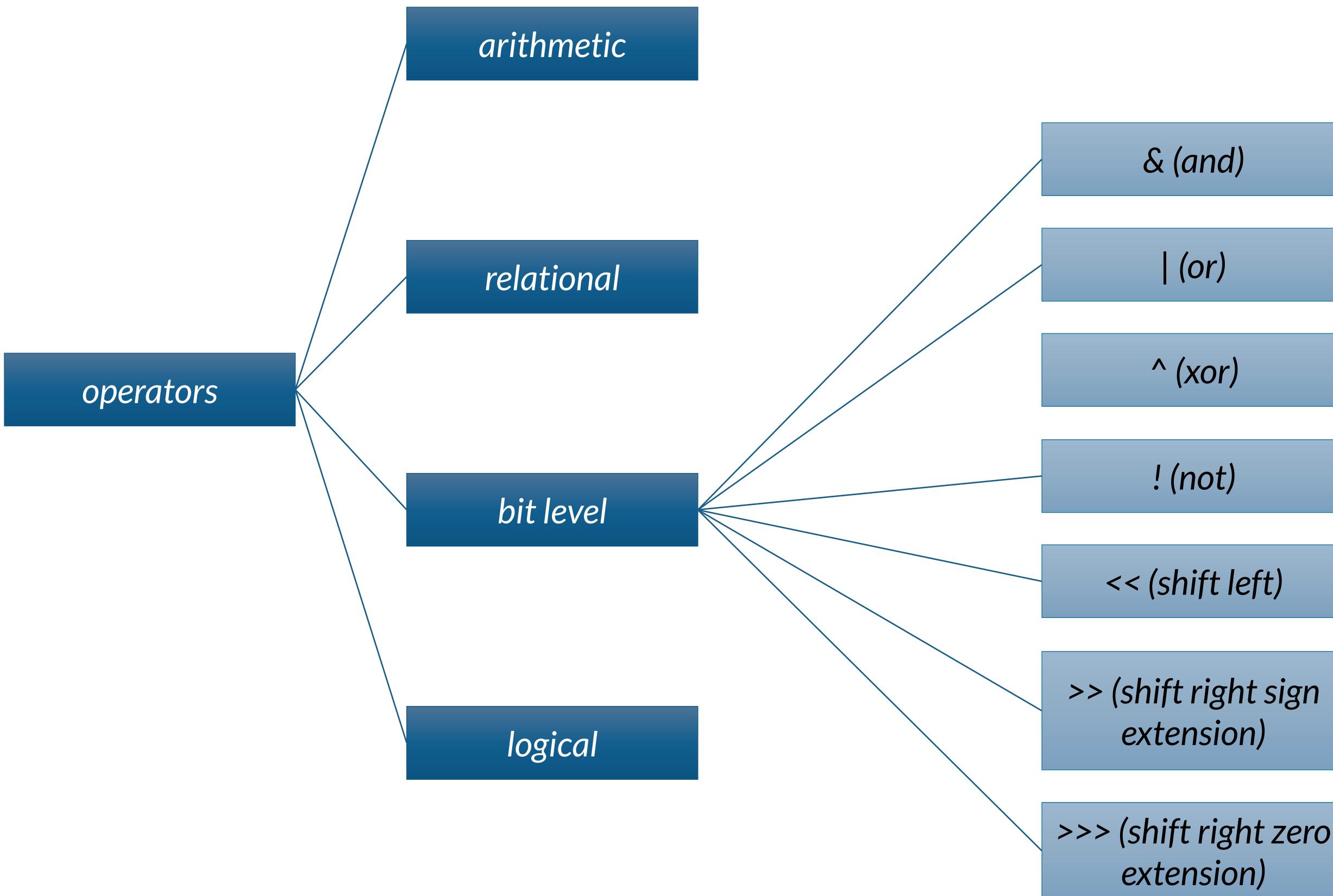
TestBoolean.java

```
public class TestBoolean {  
    public static void main(String[] args) {  
        int x = 12, y = 33;  
  
        System.out.println(x < y);  
        System.out.println(x != y - 21);  
  
        boolean test = x >= 10;  
        System.out.println(test);  
    }  
}
```

```
$ java TestBoolean  
true  
false  
true
```



Bit level operators



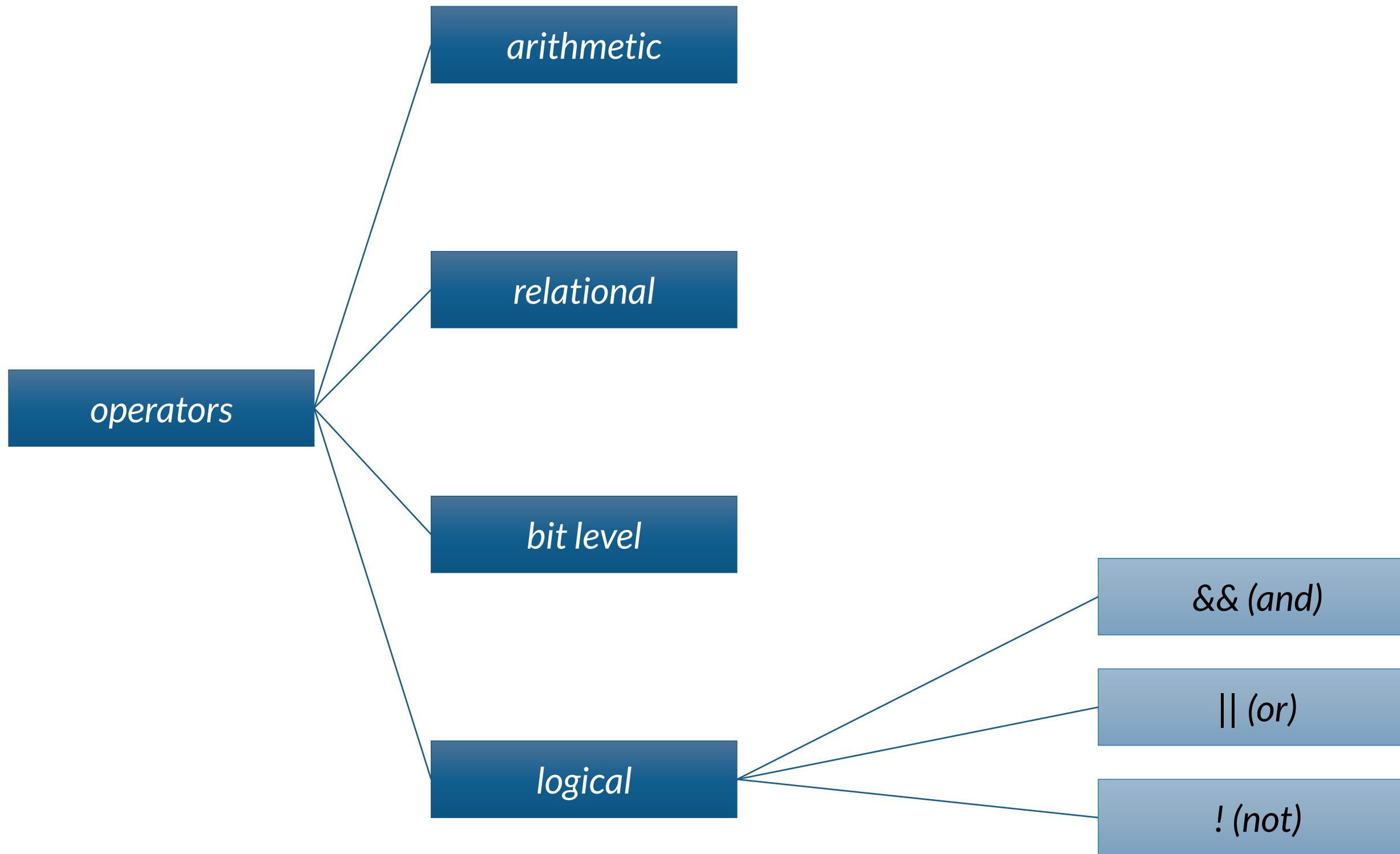
Example with bit-level operators

Bits.java

```
public class Bits {  
    public static void main(String[] args) {  
        int x = 0b000000000000000000000000000010110;  
        int y = 0b0000000000000000000000000000110011;  
  
        System.out.println(x & y);                      // 000000000000000000000000000010010  
        System.out.println(x | y);                      // 000000000000000000000000000011011  
        System.out.println(x ^ y);                      // 000000000000000000000000000010010  
        System.out.println(~x);                        // 111111111111111111111111111101001  
  
        x = 0b00000000000000000000000000001001; // 9  
        System.out.println(x >> 3);                  // 000000000000000000000000000000001  
        System.out.println(x >>>3);                 // 000000000000000000000000000000001  
  
        x = -9;                                     // 11111111111111111111111111110111  
        System.out.println(x >> 3);                  // 11111111111111111111111111111110  
        System.out.println(x >>>3);                 // 00011111111111111111111111111110  
    }  
}
```



Logical operators



Example with logical operators

Logical.java

```
public class Logical {  
    public static void main(String[] args) {  
        int x = 12, y = 33;  
        double d = 2.45, e = 4.54;  
  
        System.out.println(x < y && d < e);  
        System.out.println(!(x < y));  
  
        boolean test = 'a' > 'z';  
        System.out.println(test || d - 2.1 > 0);  
    }  
}
```

```
$ java Logical  
true  
false  
true
```

Please note that there are also logical non-short circuit operators. Investigate about them



The ? operator

Sort of if-then-else that given a conditional expression chooses between two expressions

```
condition ? expression1 : expression2
```

If condition is true, expression1 is evaluated, otherwise expression2 is evaluated.

The ?-expression assumes the result of the evaluated expression.

```
System.out.println(expression ? "It rains" : "It doesn't rain")
```

Control structures: if

If.java

```
public class If {  
    public static void main(String[] args) {  
        char c = 'x';  
  
        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))  
            System.out.println("letter: " + c);  
        else  
            if (c >= '0' && c <= '9')  
                System.out.println("digit: " + c);  
            else {  
                System.out.println("the character is: " + c);  
                System.out.println("it is not a letter nor a digit");  
            }  
    }  
}
```

```
$ java If  
letter: x
```



Control structures: while

While.java

```
public class While {  
    public static void main(String[] args) {  
        final float initialValue = 2.34F;  
        final float step = 0.11F;  
        final float limit = 4.69F;  
        float var = initialValue;  
  
        int counter = 0;  
        while (var < limit) {  
            var += step;  
            counter++;  
        }  
        System.out.println("Incremented " + counter + " times");  
    }  
}
```

```
$ java While  
Incremented 22 times
```



Control structures: for

For.java

```
public class For {  
    public static void main(String[] args) {  
        final float initialValue = 2.34F;  
        final float step = 0.11F;  
        final float limit = 4.69F;  
        int counter = 0;  
  
        for (float var = initialValue; var < limit; var += step)  
            counter++;  
        System.out.println("Incremented " + counter + " times");  
    }  
}
```

```
$ java For  
Incremented 22 times
```



Control structures: break and continue

BreakContinue.java

```
public class BreakContinue {  
    public static void main(String[] args) {  
  
        for (int counter = 0; counter < 10; counter++) {  
  
            if (counter % 2 == 1) continue; // start a new iteration if the counter is odd  
            if (counter == 8) break; // abandon the loop if the counter is equal to 8  
  
            System.out.println(counter);  
        }  
        System.out.println("done.");  
    }  
}
```

```
$ java BreakContinue  
0 2 4 6 done.
```



Control structures: switch

Switch.java

```
public class Switch {  
    public static void main(String[] args) {  
  
        boolean leapYear = true;  
        int days = 0;  
  
        for (int month = 1; month <= 12; month++) {  
            switch(month) {  
                case 1:// months with 31 days  
                case 3:  
                case 5:  
                case 7:  
                case 8:  
                case 10:  
                case 12: days += 31;  
                break;  
  
                case 2:// February is a special case  
                if (leapYear)  
                    days += 29;  
                else  
                    days += 28;  
                break;  
                default:// a month with 30 days  
                days += 30;  
                break;  
            }  
        }  
        System.out.println(days);  
    }  
}
```

```
$ java Switch  
366
```

The switch-expression must evaluate to byte, short, char, int, enum, or String



Arrays

*Arrays can be used to store elements of the **same** type*

```
int[] a;  
float[] b;  
String[] c;
```

```
int[] a = {13, 56, 2034, 4, 55};  
float[] b = {1.23F, 2.1F};  
String[] c = {"Java", "is", "great"};
```

*Important: The declaration does not specify a **size**. However, it can be inferred when initialized*

*Another possibility to allocate space for arrays consists in the use of the operator **new***

```
int i = 3, j = 5;  
double[] d;  
  
d = new double[i+j];
```



Arrays

*Java arrays are **0-based**. The components can be accessed with an integer **index** with values from **0** to **length-1**.*

```
int len = a.length;
```

*Every array has a member called **length** that can be used to get the length of the array*

*Components of the arrays are initialized with **default** values*

```
int []a = new int[3];
for (int i = 0; i < a.length; i++) {
    System.out.println(a[i]);
}
```

```
0  
0  
0
```



Arrays

Arrays.java

```
public class Arrays {  
    public static void main(String[] args) {  
        int[] a = {2, 4, 3, 1};  
  
        // compute the summation of the elements of a  
        int sum = 0;  
        for(int i = 0; i < a.length; i++) sum += a[i];  
  
        // create an array of the size computed before  
        float[] d = new float[sum];  
        for (int i = 0; i < d.length; i++) d[i] = 1.0F / (i+1);  
  
        // print values in odd positions  
        for (int i = 1; i < d.length; i += 2)  
            System.out.println("d[" + i + "]=" + d[i]);  
    }  
}
```

```
$ java Arrays  
d[1]=0.5  
d[3]=0.25  
d[5]=0.16666667  
d[7]=0.125  
d[9]=0.1
```



The for-each iteration

ForEach.java

```
public class ForEach {  
    public static void main(String[] args) {  
        int[] a = {2,4,3,1};  
  
        // compute the summation of the elements of a  
        int sum = 0;  
        for (int x : a) sum += x;  
  
        // create an array of the size computed before  
        float[] d = new float[sum];  
        for (int i = 0; i < d.length; i++) d[i] = 1.0F / (i+1);  
  
        // print all values (note the use of type inference!!)  
        for (var f : d)  
            System.out.println(f);  
    }  
}
```



Assignment

*Implement a **Calculator** class to perform arithmetic operations.*

```
$ java Calculator 6 + 4.1  
10.1  
$ java Calculator 3.6 / -2  
-1.8  
$ java Calculator 8.5 * 9  
76.5  
$ java Calculator -3.14  
-3.14
```

I let you discover how to convert strings to numbers

Enhance the calculator so that it can handle concatenated operations

```
$ java Calculator 6 + 4.1 * 3  
10.1  
30.3  
$ java Calculator 3.6 / 2 + -0.3 / .5  
1.8  
1.5  
3
```



The Java specification

The Java Language and Virtual Machine Specification are available here
<https://docs.oracle.com/javase/specs/>

The API documentation is available here
<https://docs.oracle.com/en/java/javase/17/docs/api/index.html>

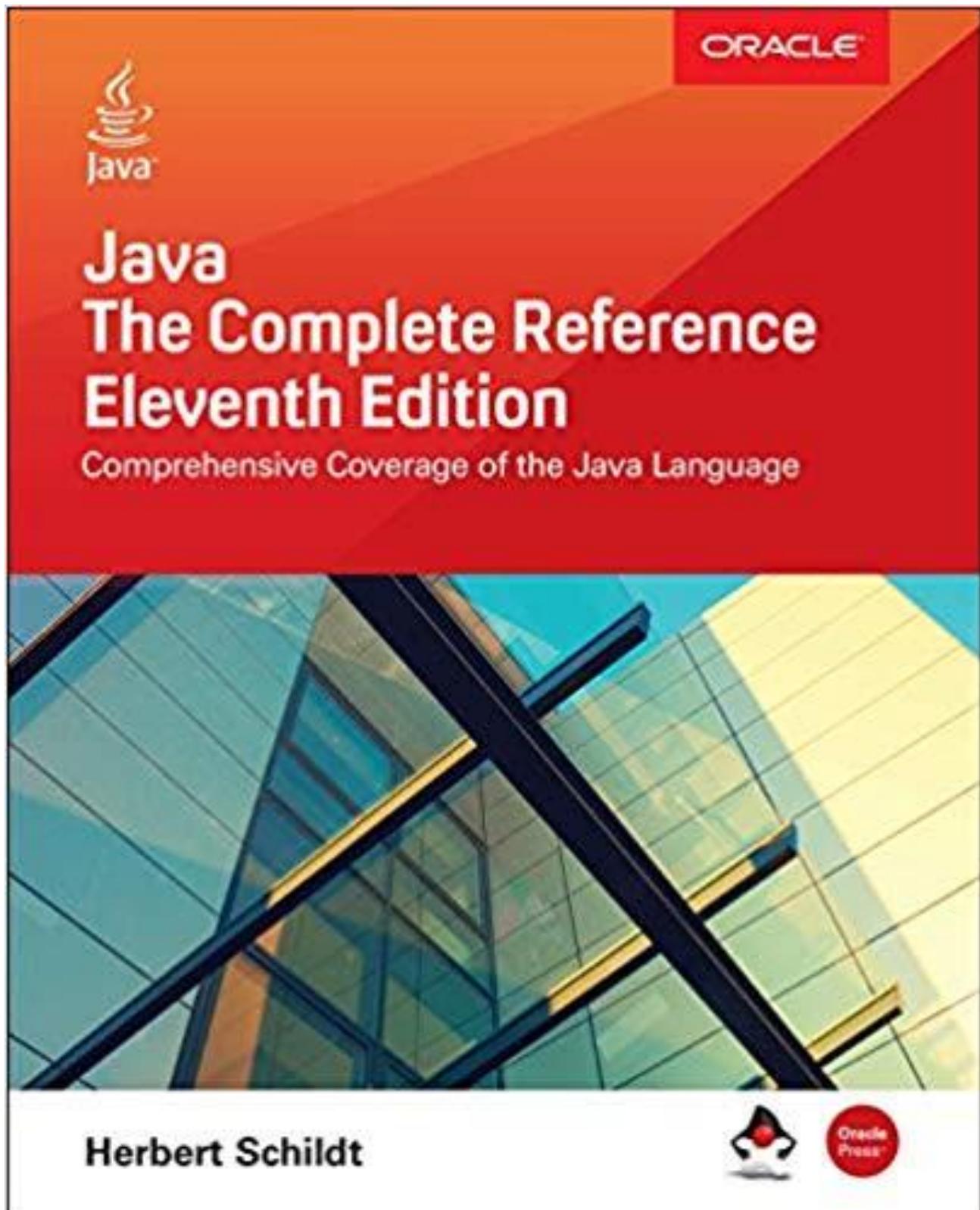
The Java language evolution is driven by the Java Community Process (JCP)
<https://www.jcp.org/en/home/index>

The JCP is the mechanism for developing standard technical specifications for Java technology. Anyone can register for the site and participate in reviewing and providing feedback for the Java Specification Requests (JSRs), and anyone can sign up to become a JCP Member and then participate on the Expert Group of a JSR or even submit their own JSR Proposals.

A more informal place to discuss the new features of Java is the JDK Enhancement Proposals (JEP)
<https://openjdk.java.net/jeps/0>



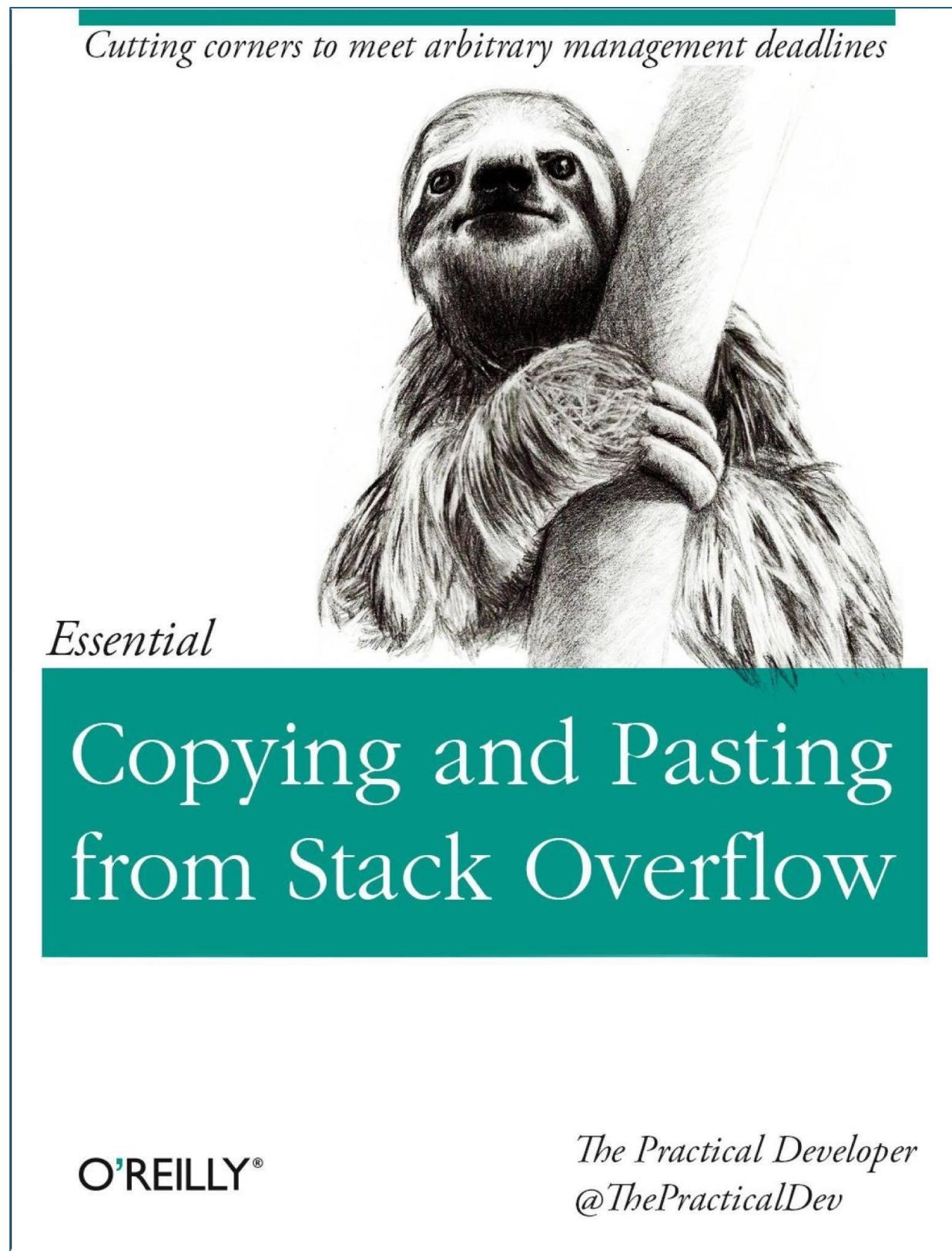
If you need a book reference



The list of good Java books is endless and continuously growing. They are good as introduction to the language at the beginning and as a reference later.

The same information they provide can be found by googling but usually in the web it is more fragmented and less structured.

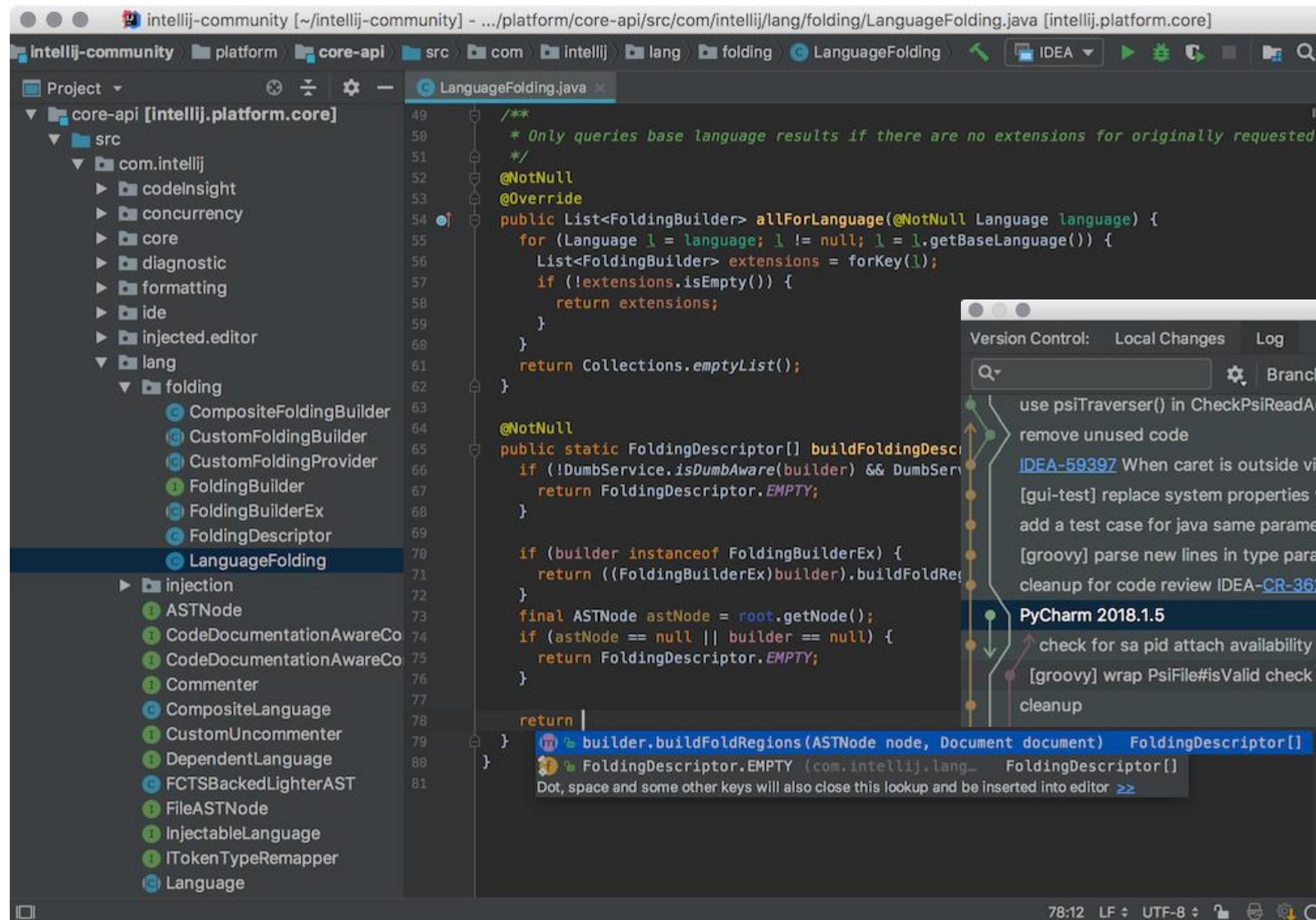
Don't buy this book



Or at least don't copy code that is 12 years old...



Recommended development tool



intellij-community [~/intellij-community] - .../platform/core-api/src/com/intellij/lang/folding/LanguageFolding.java [intellij.platform.core]

Project LanguageFolding.java

core-api [intellij.platform.core]

src

com.intellij

codeInsight

concurrency

core

diagnostic

formatting

ide

injected.editor

lang

folding

CompositeFoldingBuilder

CustomFoldingBuilder

CustomFoldingProvider

FoldingBuilder

FoldingBuilderEx

FoldingDescriptor

LanguageFolding

ASTNode

CodeDocumentationAwareCo

CodeDocumentationAwareCo

Commenter

CompositeLanguage

CustomUncommenter

DependentLanguage

FCTSBackedLighterAST

FileASTNode

InjectableLanguage

ITokenTypeRemapper

Language

```
49  */
50  * Only queries base language results if there are no extensions for originally requested
51  */
52  @NotNull
53  @Override
54  public List<FoldingBuilder> allForLanguage(@NotNull Language language) {
55      for (Language l = language; l != null; l = l.getBaseLanguage()) {
56          List<FoldingBuilder> extensions = forKey(l);
57          if (!extensions.isEmpty()) {
58              return extensions;
59          }
60      }
61      return Collections.emptyList();
62  }
63
64  @NotNull
65  public static FoldingDescriptor[] buildFoldingDescriptor(
66      if (!DumbService.isDumbAware(builder) && DumbService.isDumbAware(root)) {
67          return FoldingDescriptor.EMPTY;
68      }
69
70      if (builder instanceof FoldingBuilderEx) {
71          return ((FoldingBuilderEx)builder).buildFoldRegions(root);
72      }
73      final ASTNode astNode = root.getNode();
74      if (astNode == null || builder == null) {
75          return FoldingDescriptor.EMPTY;
76      }
77
78      return [
79          builder.buildFoldRegions(astNode, Document document)  FoldingDescriptor[]
80          FoldingDescriptor.EMPTY (com.intellij.lang.FoldingDescriptor[])
81      Dot, space and some other keys will also close this lookup and be inserted into editor >>
```

Version Control		
Local Changes		
Q	Branch: All	Date: All
use psiTraverser() in CheckPsiReadAccessors	Danill Ovchinnikov	21.08.18, 15:23
remove unused code	Dmitry Batkovich	21.08.18, 15:03
IDEA-59397 When caret is outside visible editor area, on Alt-EndLine it goes to the end of the line	Dmitry Batrak	21.08.18, 15:01
[gui-test] replace system properties with env variables	Vladislav Shishov	21.08.18, 14:12
add a test case for java same parameter value inspection	Dmitry Batkovich	21.08.18, 14:16
[groovy] parse new lines in type parameter list (IDEA-197524)	Danill Ovchinnikov	21.08.18, 14:09
cleanup for code review IDEA-CR-36249	Nikita Skvortsov	21.08.18, 14:01
PyCharm 2018.1.5	Aleksey Rostovskiy	21.08.18, 14:06
origin/181.5540	Aleksey Rostovskiy	21.08.18 at 14:06
check for sa pid attach availability - IDEA-168185	Egor Ushakov	21.08.18, 14:01
[groovy] wrap PsiFile#isValid check into a read action (IDEA-168185)	Danill Ovchinnikov*	20.08.18, 21:05
cleanup	Alexey Kudravtsev	21.08.18, 12:28



IntelliJ





esteco.com



Thank you!

