

# Java Langage & JRE Internal Basics

## Compile – (Link -) Runtime

This document:

<https://github.com/Arnaud-Nauwynck/presentations/blob/main/java/Java-Langage-JRE-Internal-Basics.pdf>

# Outline

- Overview compile – runtime chain
- Compiler basics: grammar, parser to AST
  - Declaration-Statement-Expression
  - Bytecode, stack
- Language Class Symbol resolution
  - Class.forName() / ClassLoader
  - First reference, Hot swap code
- Method Symbol resolution + call
  - Invokestatic, invokespecial
  - Invokevirtual
  - Invokeinterface
  - invokedyynamic

# Compile – Runtime Chain

UTF-8 files  
src/main/java/\*.java

```
Main.java ×  
1 package test;  
2  
3 public class Main {  
4  
5     public static void main(String[] args) {  
6         System.out.println("Hello world");  
7     }  
8  
9 }
```

javac

Bytecode (binary) files  
target/classes/\*.class

jar

jar file = zip of \*.class  
target/\*.jar

\*.jar or .class  
In CLASSPATH  
+ main FQN

java

JRE

Symbols

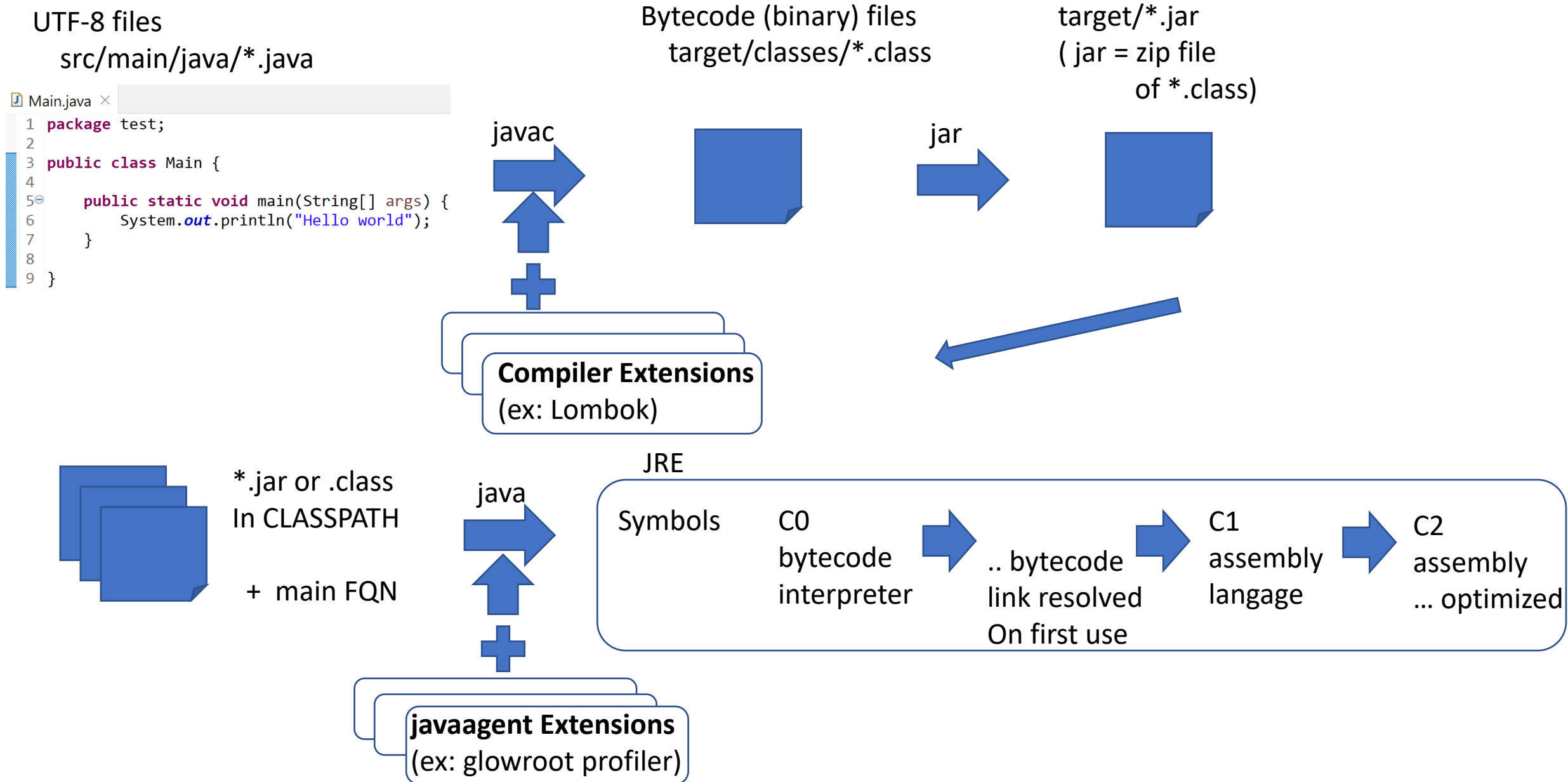
C0  
bytecode  
interpreter

.. bytecode  
link resolved  
On first use

C1  
assembly  
language

C2  
assembly  
... optimized

# Compile (+Extension) – Runtime(+JVM Agent) Chain



# Compile steps

UTF-8 files

```
src/main/java/*.java
```

```
1 package test;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Hello world");
7     }
8
9 }
```

javac

Bytecode (binary) files  
target/classes/\*.class

```
$ cat target/classes/test/Main.class | java -cp target/classes system.out.println("Hello world");
```

00000000 ; +  
00000000 ♡ ♥ ♦ ♠ ▶ java/lang/Object@ <init>() V  
00000000 ♀ ♂ ♢ ▶ java/lang/System@ out@ \$Ljava/io/PrintStream \$0 Ⓜ Hello world  
00000000 ▶ ◀ ! ? !! ☺ !! java/io/PrintStream@ println@ \$(Ljava/lang/String;)V @  
00000000 test/Main@ ◆ Code@ ☼ LineNumberTable@ ◆ main@ ■([Ljava/lang/String  
00000000 );V@  
00000000 SourceFile@ Main.java ! § @ @ @ + @ @ ↑ + @ @ + \* @ @ @ @ ↑  
00000000 @ @ @ ↑ → @ @ % @ @ @ @ ↓  
00000000 @ @ @ ? 2004h

jar

Zip file  
target/\*.jar

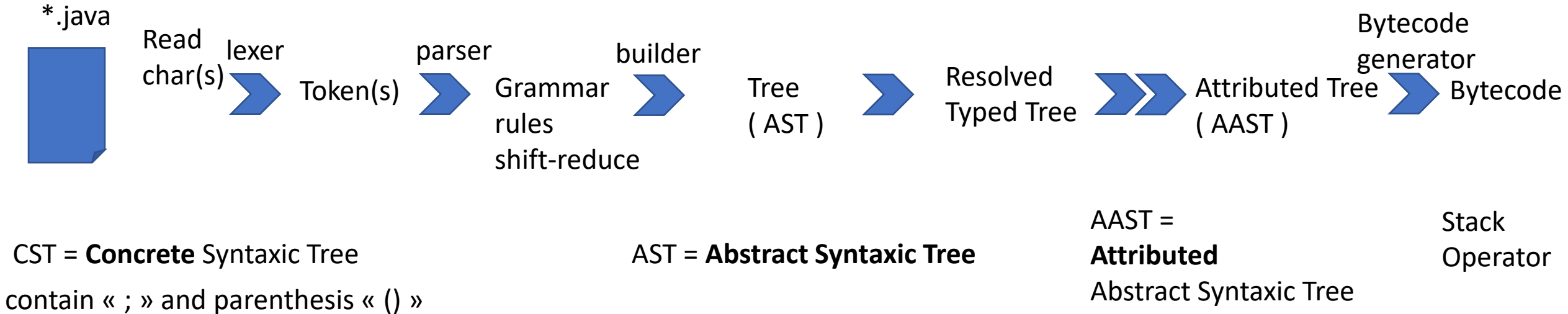
## mvn package

```
$ javac -verbose -d target/classes src/main/java/test/Main.java
[parsing started SimpleFileObject[C:\Users\arnaud\eclipse-ws\ws1\test\src\main\java\test\Main.java]]
[parsing completed 30ms]
[loading /modules/jdk.security.jgss/module-info.class]
[loading /modules/java.smartcardio/module-info.class]
[loading /modules/jdk.crypto.ec/module-info.class]
[loading /modules/jdk.charsets/module-info.class]
```

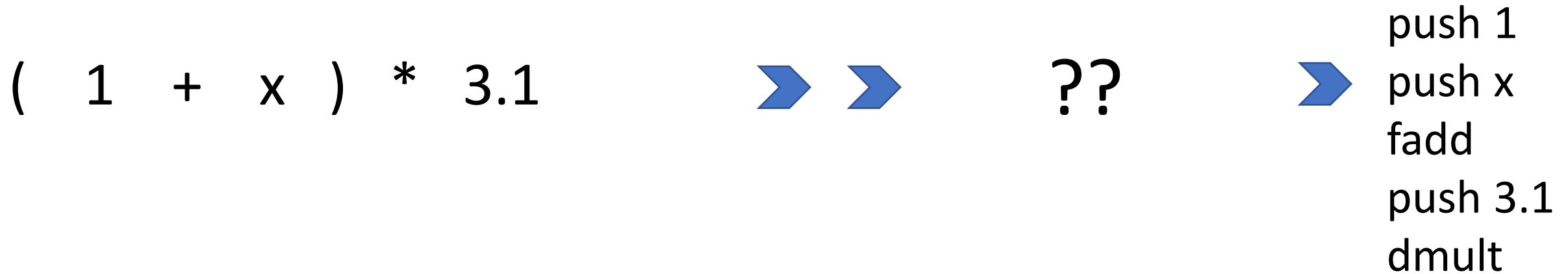
```
[checking test.Main]
```

```
[wrote target\classes\test\Main.class]
[total 480ms]
```

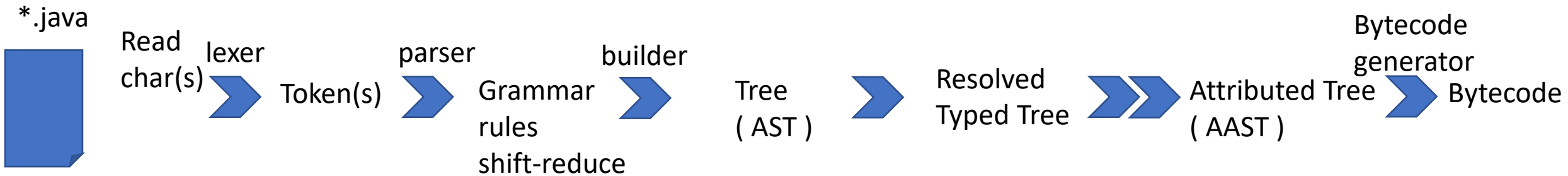
# Javac .. steps



# Example.. Compiling « $(1+x)*3.1$ » from source to bytecode



# Javac .. steps



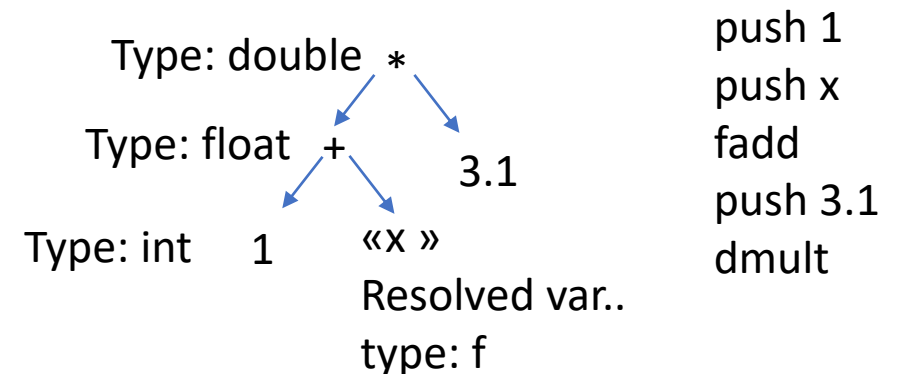
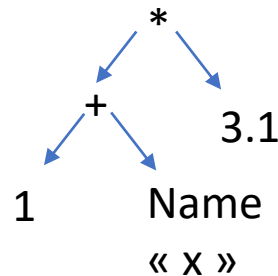
CST = **Concrete** Syntactic Tree  
contain « ; » and parenthesis « ( ) »

AST = **Abstract** Syntactic Tree

AAST =  
**Attributed**  
Abstract Syntactic Tree

Stack  
Operator

« ( 1 + x ) \* 3.1 »



push 1  
push x  
fadd  
push 3.1  
dmult



# Lexer to Tokens

« ( 1 + x ) \* 3.1 »

(   Literal   op   Name   )   op   Literal  
          number                                   number

UTF-8 source file



API to read char file:

```
reader = ..  
for(;;) {  
    char c = reader.read();  
    ..  
}
```



API to read tokens:

```
tokenLexer = ..  
for(;;) {  
    token tk = lexer.readToken();  
    ..  
}
```

# Grammar Rule

Expression ::= LitteralNumberExpr | VariableExpr | ParenthesisExpr |  
UnaryOpExpr | BinaryOp | FunctionCallExpr

LitteralNumberExpr ::= <number>

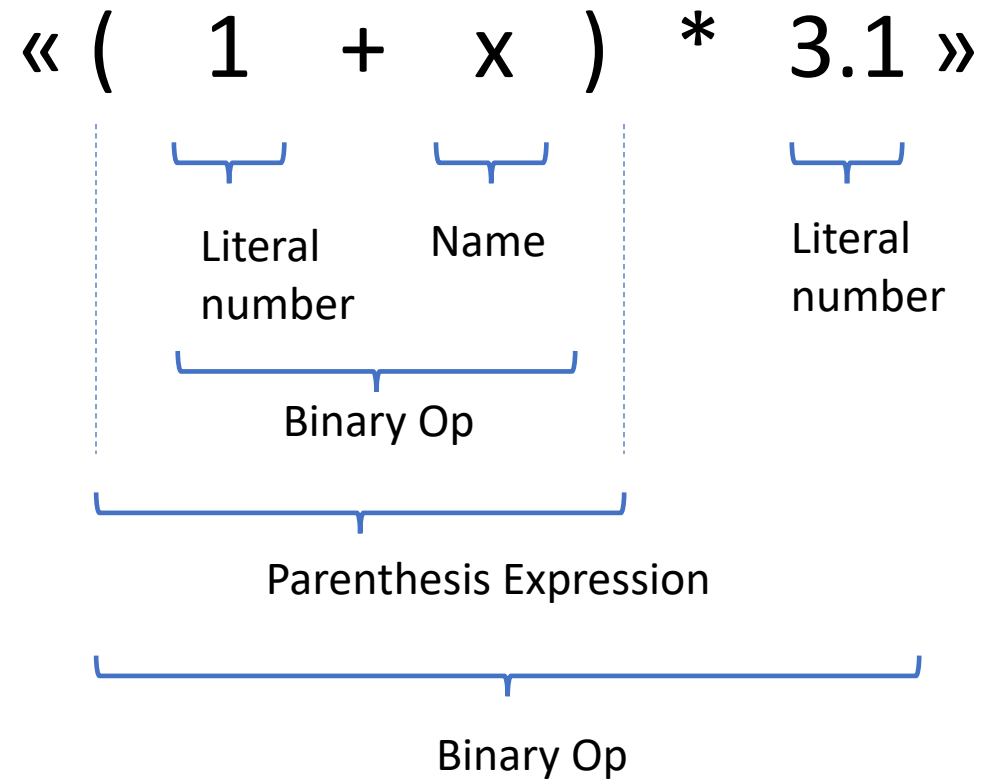
VariableExpr ::= <name>

ParenthesisExpr ::= « ( » Expression « ) »

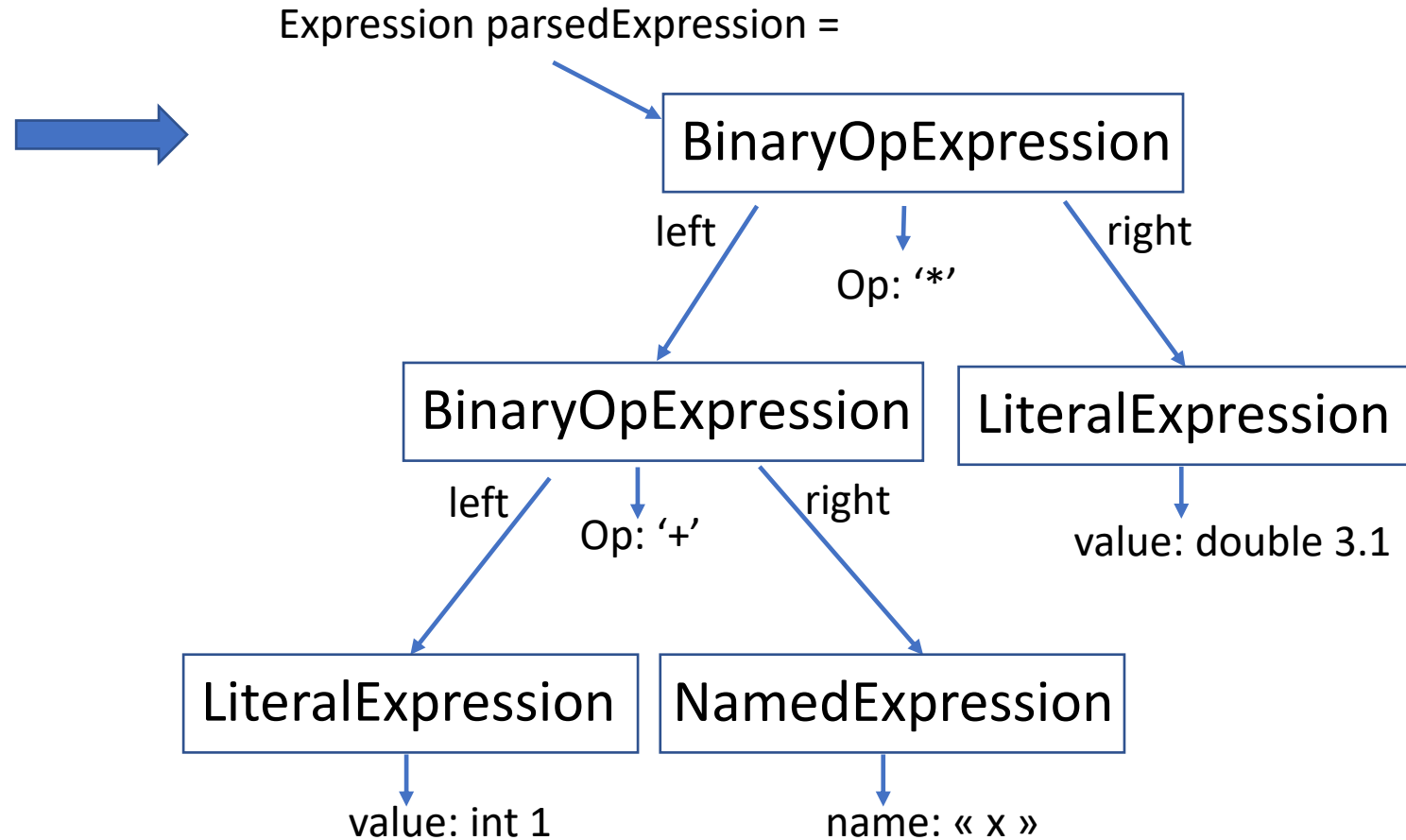
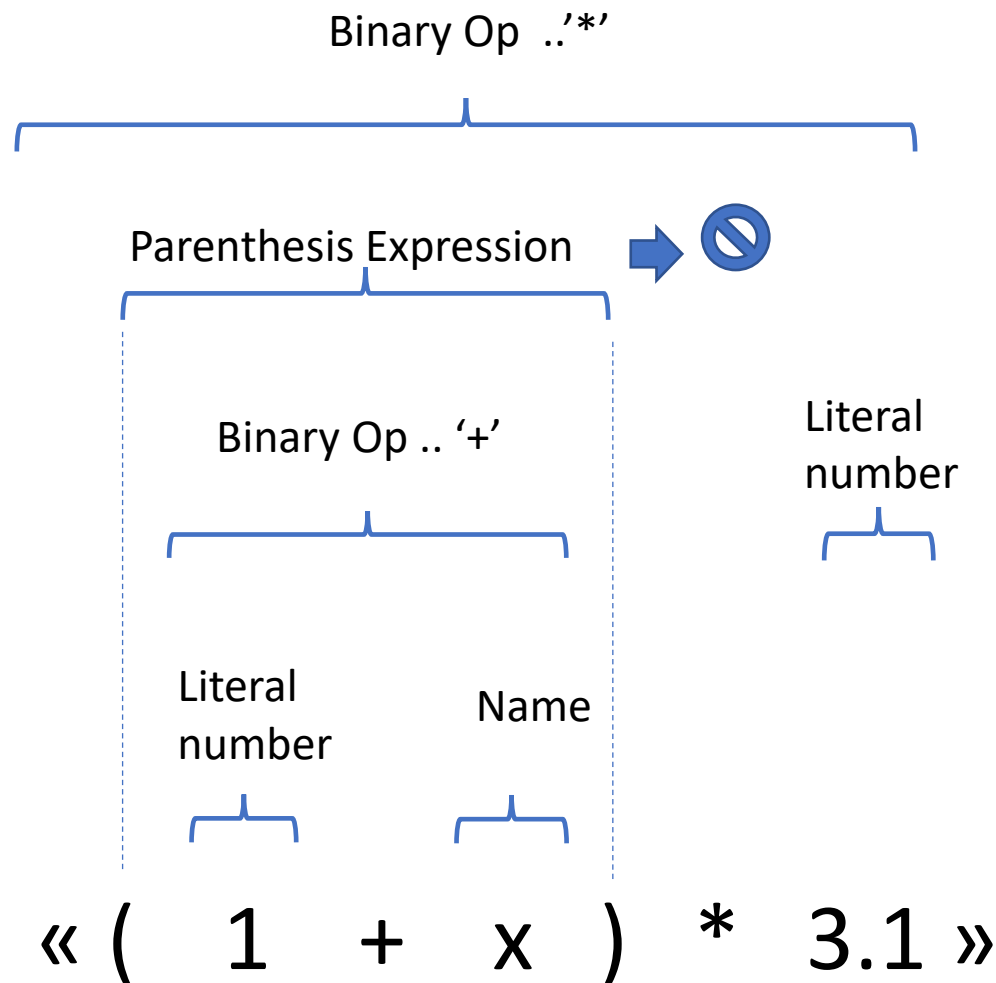
UnaryOpExpr ::= unoperator Expression  
+, -, !

BinaryOpExpr ::= Expression binayOperator Expression  
+, -, \*, /, %, &, |, ^, && ..

# Grammar Rule Parsed



# Tree (in-memory) Representation



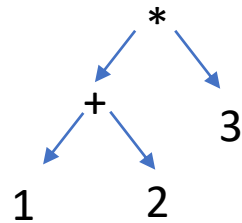
# Concrete -> Abstract Syntax Tree

lost parenthesis, « ; », indentation, comment...

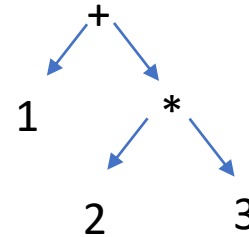
AST is NOT ambiguous, no need parenthesis

( parser need handling operator precedence )

$(1 + 2) * 3$

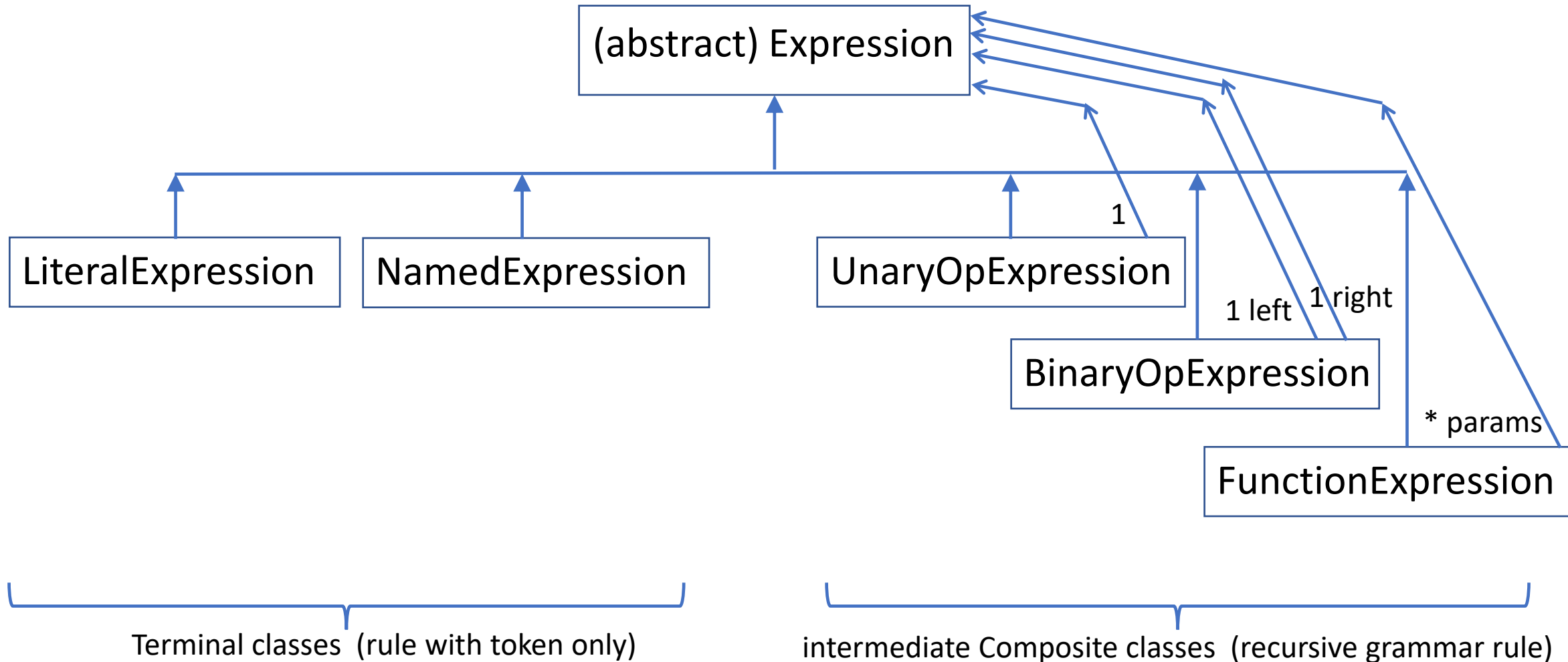


$1 + (2 * 3) = 1 + 2 * 3$

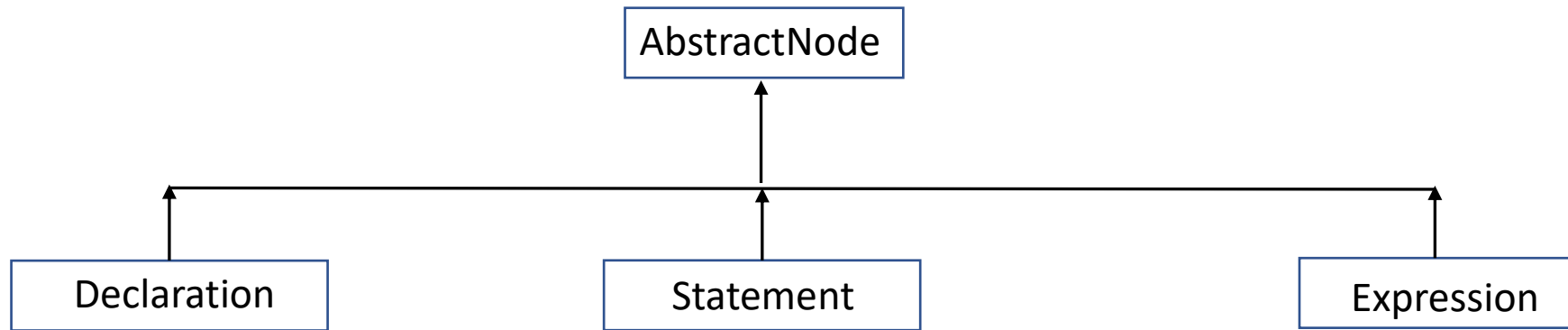


# Expression classes

## Abstract Syntax Tree (AST)



# .. Not only Expressions need Procedural (Statement) + Declarations

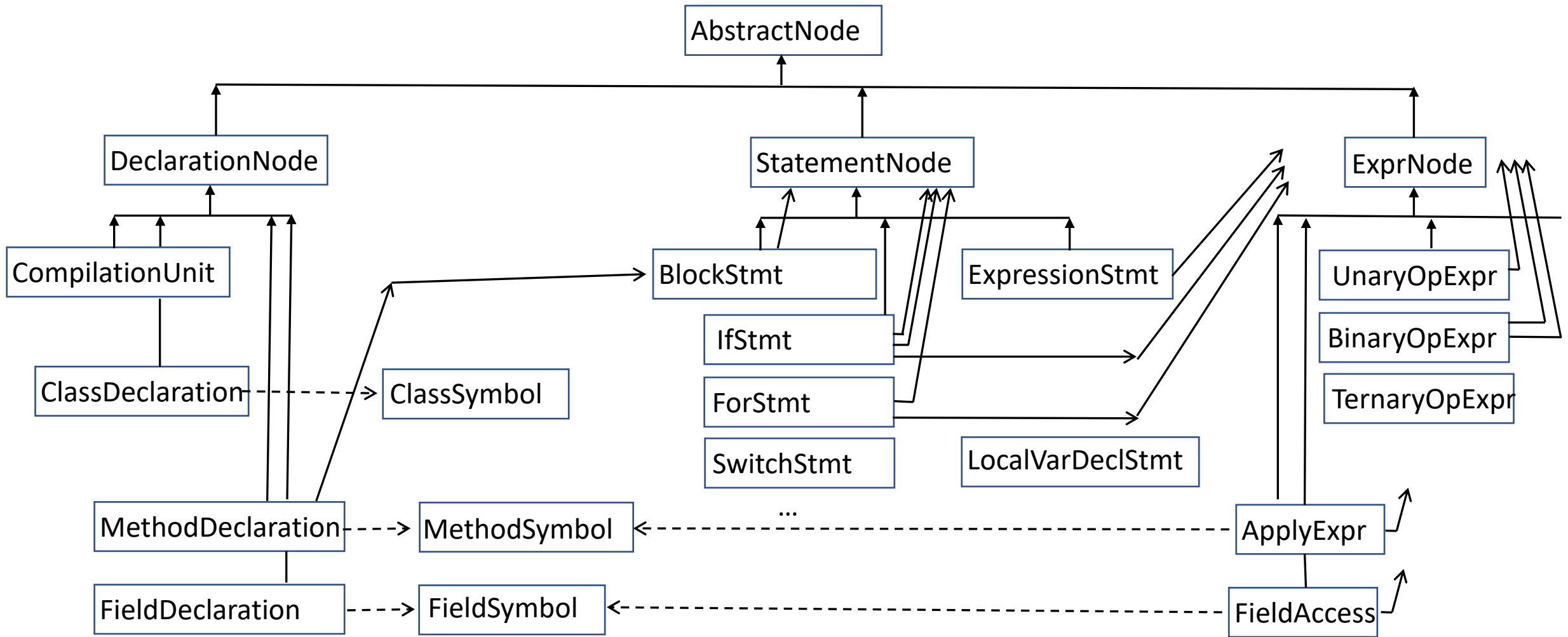


For program structure  
package, class, method, ..

For procedural parts  
« if », « for », « switch »..

For values,  
and operators

# Abstract Syntax Tree





# Correspondance

## Grammar Rule $\leftrightarrow$ AST Classes

1 grammar Rule  $\sim$  1 AST sub-class

Parse 1 grammar rule  $\Rightarrow$  build 1 AST sub tree

Example

Rule XYZ::= <expr1> token <expr2> .. token <exprN>

//  $\Rightarrow$  trigger AST builder code:

```
{ return new AstXYZ (expr1, expr2, .. exprN); }
```

# Parser

Using Code Generated  
from Grammar Rule

?

Example tools:

« lex & yacc » (origin C)

« Antlr », « JavaCC », ..

Using « hand-coded »  
Recursive parser

... example « javac »

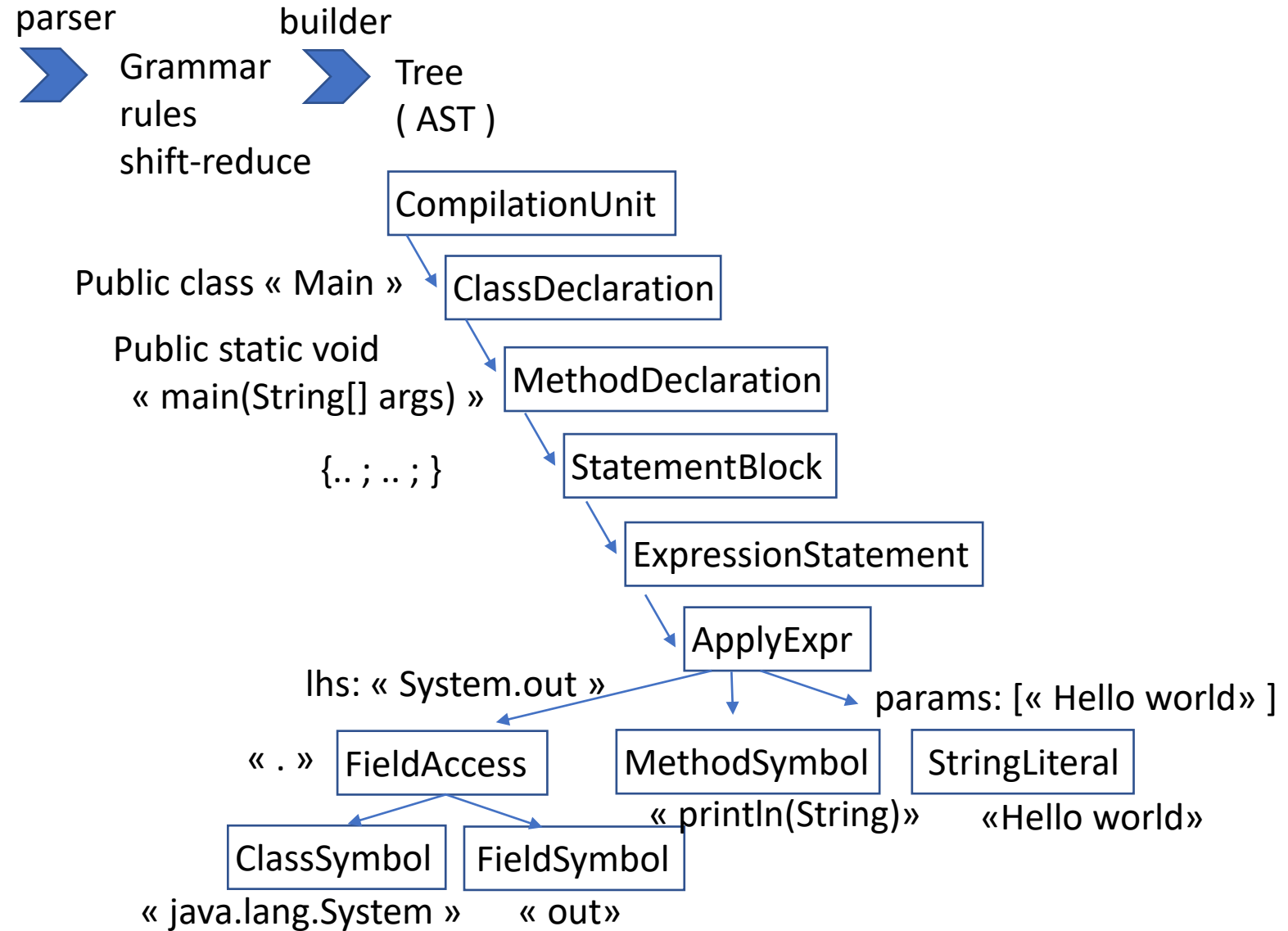
# Detailed JDK extract.. JavacParser.java

```
case FOR: {
    nextToken();
    accept(LPAREN);
    List<JCStatement> inits = token.kind == SEMI ? List.nil() : forInit();
    if (inits.length() == 1 &&
        inits.head.hasTag(VARDEF) &&
        ((JCVariableDecl) inits.head).init == null &&
        token.kind == COLON) {
        JCVariableDecl var = (JCVariableDecl) inits.head;
        accept(COLON);
        JCEXpression expr = parseExpression();
        accept(RPAREN);
        JCStatement body = parseStatementAsBlock();
        return F.at(pos).ForeachLoop(var, expr, body);
    } else {
        accept(SEMI);
        JCEXpression cond = token.kind == SEMI ? null : parseExpression();
        accept(SEMI);
        List<JCEXpressionStatement> steps = token.kind == RPAREN ? List.nil() : forUpdate();
        accept(RPAREN);
        JCStatement body = parseStatementAsBlock();
        return F.at(pos).ForLoop(inits, cond, steps, body);
    }
}
```

// <=Rule for  
**for( Type var : collection ) stmt**

// <=Rule for  
**for( init ; cond ; update ) stmt**

# Full AST « Hello World »



# AST to Text ... Pretty.java

## read easy code to understand AST

```
Pretty.java ×
781
782 public void visitForLoop(JCForLoop tree) {
783     try {
784         print("for (");
785         if (tree.init.nonEmpty()) {
786             if (tree.init.head.hasTag(VARDEF)) {
787                 printExpr(tree.init.head);
788                 for (List<JCStatement> l = tree.init.tail; l.nonEmpty(); l = l.tail) {
789                     JCVariableDecl vdef = (JCVariableDecl)l.head;
790                     print(", " + vdef.name);
791                     if (vdef.init != null) {
792                         print(" = ");
793                         printExpr(vdef.init);
794                     }
795                 }
796             } else {
797                 printExprs(tree.init);
798             }
799         }
800         print("; ");
801         if (tree.cond != null) printExpr(tree.cond);
802         print("; ");
803         printExprs(tree.step);
804         print(") ");
805         printStat(tree.body);
806     } catch (IOException e) {
807         throw new UncheckedIOException(e);
808     }
809 }
```

# Declaration

Something that can be prefixed with access modifier  
« public|protected|private » « static » « final »

A declaration produces a « symbol », that can be imported / used

# Statement

Something that can be suffixed by « ; »  
or wrapped in « { ; ; } »

A statement has no type

No push/pop values on stack ... statement need expressions

# Expression

Something that can be wrapped by « ( .. ) »

A statement has a type among

- Primitive type
- Pointer to class/enum/interface/..
- Array

Evaluating an expression => pop arguments

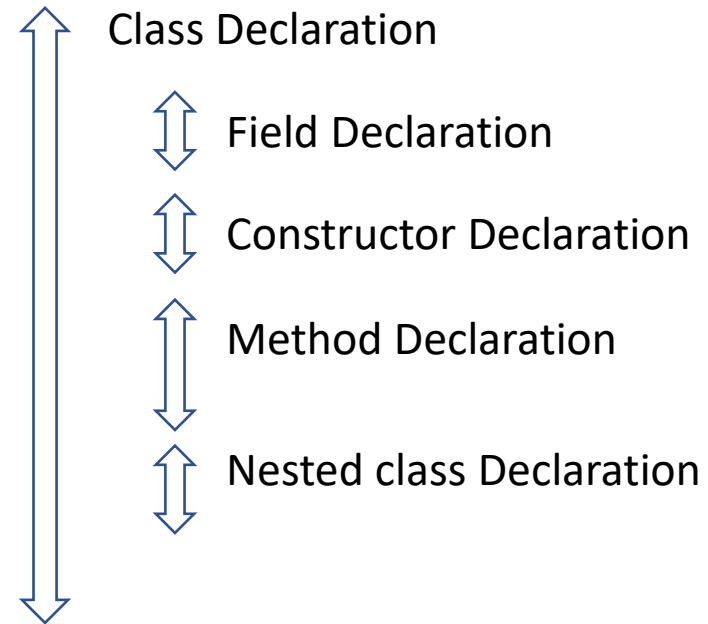
+ eval

+ push result value on stack



# Declaration examples

```
public class Foo {  
    private int field1;  
    public Foo() {  
    }  
    public int getField1() {  
        return field1;  
    }  
    public static class NestedBar {  
    }  
}
```



# Statement / Expression Examples

`int x = 0 ;`

LocalVariableDeclaration ... statement, initializer: expression

`x = 1 ;`

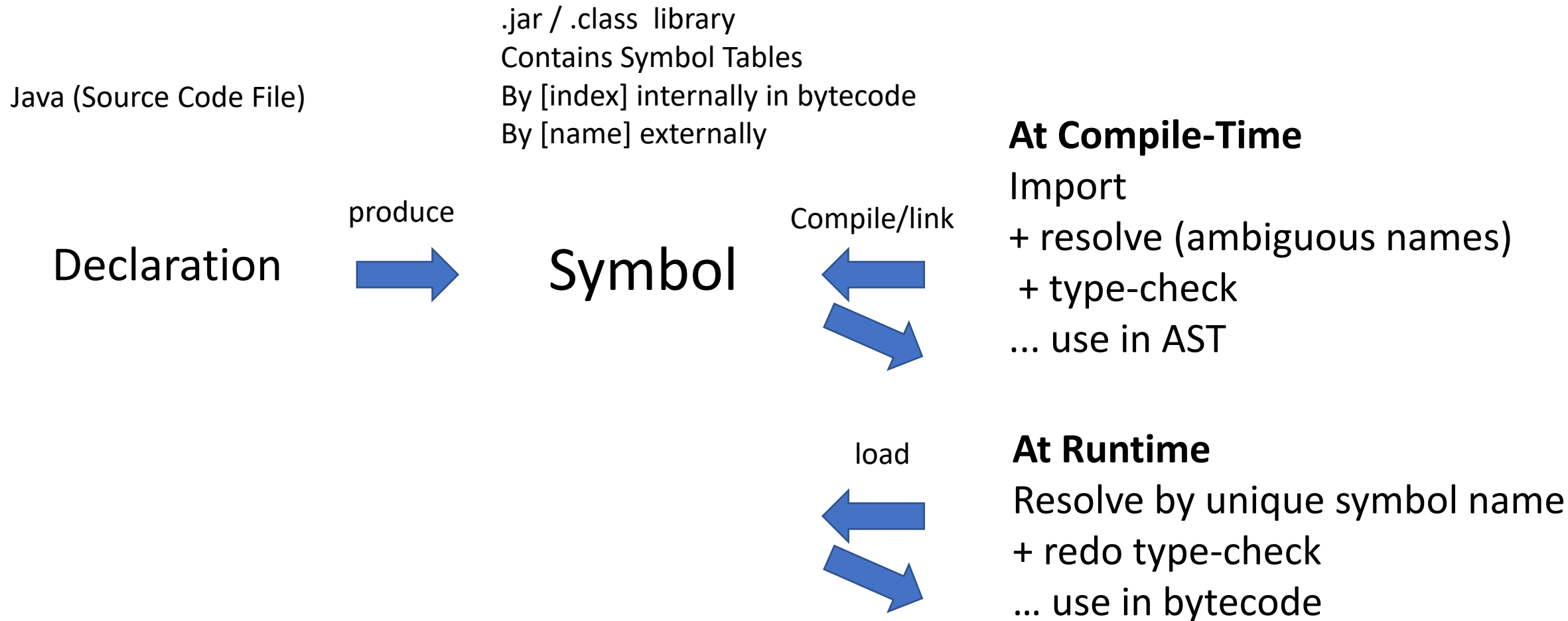
`int y = x + 1 ;`

`y = x = 2 ;`

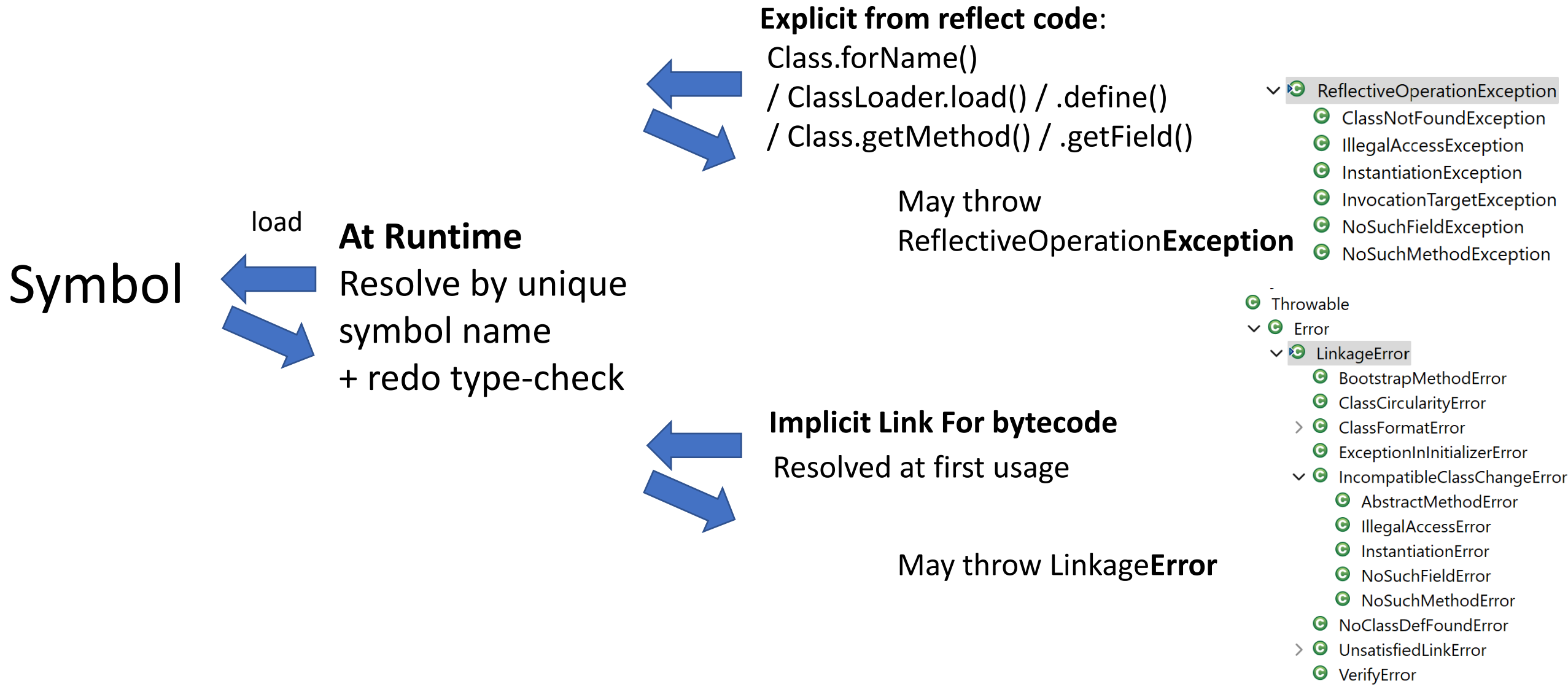
`f() ;`

ExpressionStatement ... statement containing Expression

# Symbol



# Explicit Reflection Loading / Implicit Link for bytecode



# Type Descriptor & Symbol Name Mangling

Primitive Type => V, Z(boolean), B(byte), I(int), J(long), F(float), D(double), ..

Array Type => « [ » typeName

Class => type descriptor: « L » « pack » / « subpack » / « ClassName » « ; »  
symbol name : « pack » . « subpack » . « ClassName »  
FQN (Fully Qualified Name)

Field => FQN « \$ » « fieldName »


Method => type descriptor: « (type1 type2 ..typeN) returnType »  
symbol name : «methodName(type1 type2 ..typeN) »

# Example Type Descriptor & Symbol Names

```
void fVoid();
byte fByte(byte p);
char fChar(char p);
int fInt(int p);
long fLong(long p);
float fFloat(float p);
double fDouble(double p);
boolean fBool(boolean p);

Boolean fBoolean(Boolean p);

boolean[] fArrayBool(boolean[] p);
Boolean[] fArrayBoolean(Boolean[] p);
```



```
$ javap -c -s target/classes/test/TestSignatures.class
Compiled from "TestSignatures.java"
interface test.TestSignatures {
    public abstract void fVoid();
        descriptor: ()V

    public abstract byte fByte(byte);
        descriptor: (B)B

    public abstract char fChar(char);
        descriptor: (C)C

    public abstract int fInt(int);
        descriptor: (I)I

    public abstract long fLong(long);
        descriptor: (J)J

    public abstract float fFloat(float);
        descriptor: (F)F

    public abstract double fDouble(double);
        descriptor: (D)D

    public abstract boolean fBool(boolean);
        descriptor: (Z)Z

    public abstract java.lang.Boolean fBoolean(java.lang.Boolean);
        descriptor: (Ljava/lang/Boolean;)Ljava/lang/Boolean;

    public abstract boolean[] fArrayBool(boolean[]);
        descriptor: ([Z)[Z

    public abstract java.lang.Boolean[] fArrayBoolean(java.lang.Boolean[]);
        descriptor: ([Ljava/lang/Boolean;)[Ljava/lang/Boolean;
}
```

Notice: Generic Types => same as <Object>  
exact Type Erased in Descriptor  
« Type Erasure »

```
<T> void fList1(List<? extends T> p1);
```

```
<T> void fList2(List<T> p1);
```

```
void fList3(List<Foo> p1);
```

```
void fList4(@SuppressWarnings("rawtypes") List p1);
```



```
public abstract <T> void fList1(java.util.List<? extends T>);  
descriptor: (Ljava/util/List;)V  
  
public abstract <T> void fList2(java.util.List<T>);  
descriptor: (Ljava/util/List;)V  
  
public abstract void fList3(java.util.List<test.Foo>);  
descriptor: (Ljava/util/List;)V  
  
public abstract void fList4(java.util.List);  
descriptor: (Ljava/util/List;)V
```

# Overload method signatures

Can not have 2 methods overload  
differing only by template type

```
38 public void compileErrorOverload(List<Foo> p);  
39 public void compileErrorOverload(List<Bar> p);
```

✖ Errors (2 items)

- ✖ Erasure of method `compileErrorOverload(List<Bar>)` is the same as another method in type `TestSignatures`
- ✖ Erasure of method `compileErrorOverload(List<Foo>)` is the same as another method in type `TestSignatures`

( not an error in other languages like C++)



## Notice 2 : return type not in symbol name

Can not have 2 methods overload  
differing only by return type or template type

```
38 public int compileErrorOverload();  
39 public double compileErrorOverload();
```

❌ Errors (2 items)

💡 Duplicate method compileErrorOverload() in type TestSignatures

💡 Duplicate method compileErrorOverload() in type TestSignatures

# Compiled OK ... BUT change in CLASSPATH => LinkError

```
public class Bar {  
  
    public int field1;
```



```
// previously compiled with..  
/*  
public int field1;  
*/  
// changed at runtime with..  
public long field1;
```

```
Debug ×  
LinkTestApp [Java Application]  
  test.LinkTestApp at localhost:49773  
    Thread [main] (Suspended (uncaught exception NoSuchFieldError))  
      LinkTestApp.testBarField() line: 20  
      LinkTestApp.testFields() line: 12  
      LinkTestApp.main(String[]) line: 6  
      C:\apps\jdk\jdk-8\bin\javaw.exe (1 août 2022, 14:02:57) [pid: 4852]  
  
LinkTestApp.java ×  
11      testFooField();  
12      testBarField(); // <= will throw Link error on first call  
13  }  
14  private static void testFooField() {  
15      Foo obj = new Foo();  
16      System.out.println(obj.field1);  
17  }  
18  private static void testBarField() { // method contains Link error: referenced field changed at runtime  
19      Bar obj = new Bar();  
20      System.out.println(obj.field1); // <= cause Link error... ".field1" changed at runtime  
21  }
```

Program start OK

Error method partially execute .. OK !  
but fail on first bytecode Link error

```
0  
Exception in thread "main" java.lang.NoSuchFieldError: field1  
    at test.LinkTestApp.testBarField(LinkTestApp.java:20)  
    at test.LinkTestApp.testFields(LinkTestApp.java:12)  
    at test.LinkTestApp.main(LinkTestApp.java:6)
```

# Internally ... « getfield » => « \_fast\_\*getfield »

```
public static void testBarField(); static void testBarMethod() { // method contains Link error
Code:      28      Bar obj = new Bar();
           29      #51      obj.f(); // class test/Bar
           30      }
           31      // Method test/Bar."<init>":()V
           32      public static void testFooMethod() {
           33      Foo obj = new Foo();
           34      obj.f(); // Field java/lang/System.out:Ljava/io/PrintStream;
           35      }
           36      }
           37      }
           38      }
           39      }
           40      }
           41      }
           42      }
           43      }
           44      }
           45      }
           46      }
           47      }
           48      }
           49      }
           50      }
           51      }
           52      }
           53      }
           54      }
           55      }
           56      }
           57      }
           58      }
           59      }
           60      }
           61      }
           62      }
           63      }
           64      }
           65      }
           66      }
           67      }
           68      }
           69      }
           70      }
           71      }
           72      }
           73      }
           74      }
           75      }
           76      }
           77      }
           78      }
           79      }
           80      }
           81      }
           82      }
           83      }
           84      }
           85      }
           86      }
           87      }
           88      }
           89      }
           90      }
           91      }
           92      }
           93      }
           94      }
           95      }
           96      }
           97      }
           98      }
           99      }
          100      }
          101      }
          102      }
          103      }
          104      }
          105      }
          106      }
          107      }
          108      }
          109      }
          110      }
          111      }
          112      }
          113      }
          114      }
          115      }
          116      }
          117      }
          118      }
          119      }
          120      }
          121      }
          122      }
          123      }
          124      }
          125      }
          126      }
          127      }
          128      }
          129      }
          130      }
          131      }
          132      }
          133      }
          134      }
          135      }
          136      }
          137      }
          138      }
          139      }
          140      }
          141      }
          142      }
          143      }
          144      }
          145      }
          146      }
          147      }
          148      }
          149      }
          150      }
          151      }
          152      }
          153      }
          154      }
          155      }
          156      }
          157      }
          158      }
          159      }
          160      }
          161      }
          162      }
          163      }
          164      }
          165      }
          166      }
          167      }
          168      }
          169      }
          170      }
          171      }
          172      }
          173      }
          174      }
          175      }
          176      }
          177      }
          178      }
          179      }
          180      }
          181      }
          182      }
          183      }
          184      }
          185      }
          186      }
          187      }
          188      }
          189      }
          190      }
          191      }
          192      }
          193      }
          194      }
          195      }
          196      }
          197      }
          198      }
          199      }
          200      }
          201      }
          202      }
          203      }
          204      }
          205      }
          206      }
          207      }
          208      }
          209      }
          210      }
          211      }
          212      }
          213      }
          214      }
          215      }
          216      }
          217      }
          218      }
          219      }
          220      }
          221      }
          222      }
          223      }
          224      }
          225      }
          226      }
          227      }
          228      }
          229      }
          230      }
          231      }
          232      }
          233      }
          234      }
          235      }
          236      }
          237      }
          238      }
          239      }
          240      }
          241      }
          242      }
          243      }
          244      }
          245      }
          246      }
          247      }
          248      }
          249      }
          250      }
          251      }
          252      }
          253      }
          254      }
          255      }
          256      }
          257      }
          258      }
          259      }
          260      }
          261      }
          262      }
          263      }
          264      }
          265      }
          266      }
          267      }
          268      }
          269      }
          270      }
          271      }
          272      }
          273      }
          274      }
          275      }
          276      }
          277      }
          278      }
          279      }
          280      }
          281      }
          282      }
          283      }
          284      }
          285      }
          286      }
          287      }
          288      }
          289      }
          290      }
          291      }
          292      }
          293      }
          294      }
          295      }
          296      }
          297      }
          298      }
          299      }
          300      }
          301      }
          302      }
          303      }
          304      }
          305      }
          306      }
          307      }
          308      }
          309      }
          310      }
          311      }
          312      }
          313      }
          314      }
          315      }
          316      }
          317      }
          318      }
          319      }
          320      }
          321      }
          322      }
          323      }
          324      }
          325      }
          326      }
          327      }
          328      }
          329      }
          330      }
          331      }
          332      }
          333      }
          334      }
          335      }
          336      }
          337      }
          338      }
          339      }
          340      }
          341      }
          342      }
          343      }
          344      }
          345      }
          346      }
          347      }
          348      }
          349      }
          350      }
          351      }
          352      }
          353      }
          354      }
          355      }
          356      }
          357      }
          358      }
          359      }
          360      }
          361      }
          362      }
          363      }
          364      }
          365      }
          366      }
          367      }
          368      }
          369      }
          370      }
          371      }
          372      }
          373      }
          374      }
          375      }
          376      }
          377      }
          378      }
          379      }
          380      }
          381      }
          382      }
          383      }
          384      }
          385      }
          386      }
          387      }
          388      }
          389      }
          390      }
          391      }
          392      }
          393      }
          394      }
          395      }
          396      }
          397      }
          398      }
          399      }
          400      }
          401      }
          402      }
          403      }
          404      }
          405      }
          406      }
          407      }
          408      }
          409      }
          410      }
          411      }
          412      }
          413      }
          414      }
          415      }
          416      }
          417      }
          418      }
          419      }
          420      }
          421      }
          422      }
          423      }
          424      }
          425      }
          426      }
          427      }
          428      }
          429      }
          430      }
          431      }
          432      }
          433      }
          434      }
          435      }
          436      }
          437      }
          438      }
          439      }
          440      }
          441      }
          442      }
          443      }
          444      }
          445      }
          446      }
          447      }
          448      }
          449      }
          450      }
          451      }
          452      }
          453      }
          454      }
          455      }
          456      }
          457      }
          458      }
          459      }
          460      }
          461      }
          462      }
          463      }
          464      }
          465      }
          466      }
          467      }
          468      }
          469      }
          470      }
          471      }
          472      }
          473      }
          474      }
          475      }
          476      }
          477      }
          478      }
          479      }
          480      }
          481      }
          482      }
          483      }
          484      }
          485      }
          486      }
          487      }
          488      }
          489      }
          490      }
          491      }
          492      }
          493      }
          494      }
          495      }
          496      }
          497      }
          498      }
          499      }
          500      }
          501      }
          502      }
          503      }
          504      }
          505      }
          506      }
          507      }
          508      }
          509      }
          510      }
          511      }
          512      }
          513      }
          514      }
          515      }
          516      }
          517      }
          518      }
          519      }
          520      }
          521      }
          522      }
          523      }
          524      }
          525      }
          526      }
          527      }
          528      }
          529      }
          530      }
          531      }
          532      }
          533      }
          534      }
          535      }
          536      }
          537      }
          538      }
          539      }
          540      }
          541      }
          542      }
          543      }
          544      }
          545      }
          546      }
          547      }
          548      }
          549      }
          550      }
          551      }
          552      }
          553      }
          554      }
          555      }
          556      }
          557      }
          558      }
          559      }
          560      }
          561      }
          562      }
          563      }
          564      }
          565      }
          566      }
          567      }
          568      }
          569      }
          570      }
          571      }
          572      }
          573      }
          574      }
          575      }
          576      }
          577      }
          578      }
          579      }
          580      }
          581      }
          582      }
          583      }
          584      }
          585      }
          586      }
          587      }
          588      }
          589      }
          590      }
          591      }
          592      }
          593      }
          594      }
          595      }
          596      }
          597      }
          598      }
          599      }
          600      }
          601      }
          602      }
          603      }
          604      }
          605      }
          606      }
          607      }
          608      }
          609      }
          610      }
          611      }
          612      }
          613      }
          614      }
          615      }
          616      }
          617      }
          618      }
          619      }
          620      }
          621      }
          622      }
          623      }
          624      }
          625      }
          626      }
          627      }
          628      }
          629      }
          630      }
          631      }
          632      }
          633      }
          634      }
          635      }
          636      }
          637      }
          638      }
          639      }
          640      }
          641      }
          642      }
          643      }
          644      }
          645      }
          646      }
          647      }
          648      }
          649      }
          650      }
          651      }
          652      }
          653      }
          654      }
          655      }
          656      }
          657      }
          658      }
          659      }
          660      }
          661      }
          662      }
          663      }
          664      }
          665      }
          666      }
          667      }
          668      }
          669      }
          670      }
          671      }
          672      }
          673      }
          674      }
          675      }
          676      }
          677      }
          678      }
          679      }
          680      }
          681      }
          682      }
          683      }
          684      }
          685      }
          686      }
          687      }
          688      }
          689      }
          690      }
          691      }
          692      }
          693      }
          694      }
          695      }
          696      }
          697      }
          698      }
          699      }
          700      }
          701      }
          702      }
          703      }
          704      }
          705      }
          706      }
          707      }
          708      }
          709      }
          710      }
          711      }
          712      }
          713      }
          714      }
          715      }
          716      }
          717      }
          718      }
          719      }
          720      }
          721      }
          722      }
          723      }
          724      }
          725      }
          726      }
          727      }
          728      }
          729      }
          730      }
          731      }
          732      }
          733      }
          734      }
          735      }
          736      }
          737      }
          738      }
          739      }
          740      }
          741      }
          742      }
          743      }
          744      }
          745      }
          746      }
          747      }
          748      }
          749      }
          750      }
          751      }
          752      }
          753      }
          754      }
          755      }
          756      }
          757      }
          758      }
          759      }
          760      }
          761      }
          762      }
          763      }
          764      }
          765      }
          766      }
          767      }
          768      }
          769      }
          770      }
          771      }
          772      }
          773      }
          774      }
          775      }
          776      }
          777      }
          778      }
          779      }
          780      }
          781      }
          782      }
          783      }
          784      }
          785      }
          786      }
          787      }
          788      }
          789      }
          790      }
          791      }
          792      }
          793      }
          794      }
          795      }
          796      }
          797      }
          798      }
          799      }
          800      }
          801      }
          802      }
          803      }
          804      }
          805      }
          806      }
          807      }
          808      }
          809      }
          810      }
          811      }
          812      }
          813      }
          814      }
          815      }
          816      }
          817      }
          818      }
          819      }
          820      }
          821      }
          822      }
          823      }
          824      }
          825      }
          826      }
          827      }
          828      }
          829      }
          830      }
          831      }
          832      }
          833      }
          834      }
          835      }
          836      }
          837      }
          838      }
          839      }
          840      }
          841      }
          842      }
          843      }
          844      }
          845      }
          846      }
          847      }
          848      }
          849      }
          850      }
          851      }
          852      }
          853      }
          854      }
          855      }
          856      }
          857      }
          858      }
          859      }
          860      }
          861      }
          862      }
          863      }
          864      }
          865      }
          866      }
          867      }
          868      }
          869      }
          870      }
          871      }
          872      }
          873      }
          874      }
          875      }
          876      }
          877      }
          878      }
          879      }
          880      }
          881      }
          882      }
          883      }
          884      }
          885      }
          886      }
          887      }
          888      }
          889      }
          890      }
          891      }
          892      }
          893      }
          894      }
          895      }
          896      }
          897      }
          898      }
          899      }
          900      }
          901      }
          902      }
          903      }
          904      }
          905      }
          906      }
          907      }
          908      }
          909      }
          910      }
          911      }
          912      }
          913      }
          914      }
          915      }
          916      }
          917      }
          918      }
          919      }
          920      }
          921      }
          922      }
          923      }
          924      }
          925      }
          926      }
          927      }
          928      }
          929      }
          930      }
          931      }
          932      }
          933      }
          934      }
          935      }
          936      }
          937      }
          938      }
          939      }
          940      }
          941      }
          942      }
          943      }
          944      }
          945      }
          946      }
          947      }
          948      }
          949      }
          950      }
          951      }
          952      }
          953      }
          954      }
          955      }
          956      }
          957      }
          958      }
          959      }
          960      }
          961      }
          962      }
          963      }
          964      }
          965      }
          966      }
          967      }
          968      }
          969      }
          970      }
          971      }
          972      }
          973      }
          974      }
          975      }
          976      }
          977      }
          978      }
          979      }
          980      }
          981      }
          982      }
          983      }
          984      }
          985      }
          986      }
          987      }
          988      }
          989      }
          990      }
          991      }
          992      }
          993      }
          994      }
          995      }
          996      }
          997      }
          998      }
          999      }
         1000      }
         1001      }
         1002      }
         1003      }
         1004      }
         1005      }
         1006      }
         1007      }
         1008      }
         1009      }
         1010      }
         1011      }
         1012      }
         1013      }
         1014      }
         1015      }
         1016      }
         1017      }
         1018      }
         1019      }
         1020      }
         1021      }
         1022      }
         1023      }
         1024      }
         1025      }
         1026      }
         1027      }
         1028      }
         1029      }
         1030      }
         1031      }
         1032      }
         1033      }
         1034      }
         1035      }
         1036      }
         1037      }
         1038      }
         1039      }
         1040      }
         1041      }
         1042      }
         1043      }
         1044      }
         1045      }
         1046      }
         1047      }
         1048      }
         1049      }
         1050      }
         1051      }
         1052      }
         1053      }
         1054      }
         1055      }
         1056      }
         1057      }
         1058      }
         1059      }
         1060      }
         1061      }
         1062      }
         1063      }
         1064      }
         1065      }
         1066      }
         1067      }
         1068      }
         1069      }
         1070      }
         1071      }
         1072      }
         1073      }
         1074      }
         1075      }
         1076      }
         1077      }
         1078      }
         1079      }
         1080      }
         1081      }
         1082      }
         1083      }
         1084      }
         1085      }
         1086      }
         1087      }
         1088      }
         1089      }
         1090      }
         1091      }
         1092      }
         1093      }
         1094      }
         1095      }
         1096      }
         1097      }
         1098      }
         1099      }
         1100      }
         1101      }
         1102      }
         1103      }
         1104      }
         1105      }
         1106      }
         1107      }
         1108      }
         1109      }
         1110      }
         1111      }
         1112      }
         1113      }
         1114      }
         1115      }
         1116      }
         1117      }
         1118      }
         1119      }
         1120      }
         1121      }
         1122      }
         1123      }
         1124      }
         1125      }
         1126      }
         1127      }
         1128      }
         1129      }
         1130      }
         1131      }
         1132      }
         1133      }
         1134      }
         1135      }
         1136      }
         1137      }
         1138      }
         1139      }
         1140      }
         1141      }
         1142      }
         1143      }
         1144      }
         1145      }
         1146      }
         1147      }
         1148      }
         1149      }
         1150      }
         1151      }
         1152      }
         1153      }
         1154      }
         1155      }
         1156      }
         1157      }
         1158      }
         1159      }
         1160      }
         1161      }
         1162      }
         1163      }
         1164      }
         1165      }
         1166      }
         1167      }
         1168      }
         1169      }
         1170      }
         1171      }
         1172      }
         1173      }
         1174      }
         1175      }
         1176      }
         1177      }
         1178      }
         1179      }
         1180      }
         1181      }
         1182      }
         1183      }
         1184      }
         1185      }
         1186      }
         1187      }
         1188      }
         1189      }
         1190      }
         1191      }
         1192      }
         1193      }
         1194      }
         1195      }
         1196      }
         1197      }
         1198      }
         1199      }
         1200      }
         1201      }
         1202      }
         1203      }
         1204      }
         1205      }
         1206      }
         1207      }
         1208      }
         1209      }
         1210      }
         1211      }
         1212      }
         1213      }
         1214      }
         1215      }
         1216      }
         1217      }
         1218      }
         1219      }
         1220      }
         1221      }
         1222      }
         1223      }
         1224      }
         1225      }
         1226      }
         1227      }
         1228      }
         1229      }
         1230      }
         1231      }
         1232      }
         1233      }
         1234      }
         1235      }
         1236      }
         1237      }
         1238      }
         1239      }
         1240      }
         1241      }
         1242      }
         1243      }
         1244      }
         1245      }
         1246      }
         1247      }
         1248      }
         1249      }
         1250      }
         1251      }
         1252      }
         1253      }
         1254      }
         1255      }
         1256      }
         1257      }
         1258      }
         1259      }
         1260      }
         1261      }
         1262      }
         1263      }
         1264      }
         1265      }
         1266      }
         1267      }
         1268      }
         1269      }
         1270      }
         1271      }
         1272      }
         1273      }
         1274      }
         1275      }
         1276      }
         1277      }
         1278      }
         1279      }
         1280      }
         1281      }
         1282      }
         1283      }
         1284      }
         1285      }
         1286      }
         1287      }
         1288      }
         1289      }
         1290      }
         1291      }
         1292      }
         1293      }
         1294      }
         1295      }
         1296      }
         1297      }
         1298      }
         1299      }
         1300      }
         1301      }
         1302      }
         1303      }
         1304      }
         1305      }
         1306      }
         1307      }
         1308      }
         1309      }
         1310      }
         1311      }
         1312      }
         1313      }
         1314      }
         1315      }
         1316      }
         1317      }
         1318      }
         1319      }
         1320      }
         1321      }
         1322      }
         1323      }
         1324      }
         1325      }
         1326      }
         1327      }
         1328      }
         1329      }
         1330      }
         1331      }
         1332      }
         1333      }
         1334      }
         1335      }
         1336      }
         1337      }
         1338      }
         1339      }
         1340      }
         1341      }
         1342      }
         1343      }
         1344      }
         1345      }
         1346      }
         1347      }
         1348      }
         1349      }
         1350      }
         1351      }
         1352      }
         1353      }
         1354      }
         1355      }
         1356      }
         1357      }
         1358      }
         1359      }
         1360      }
         1361      }
         1362      }
         1363      }
         1364      }
         1365      }
         1366      }
         1367      }
         1368      }
         1369      }
         1370      }
         1371      }
         1372      }
         1373      }
         1374      }
         1375      }
         1376      }
         1377      }
         1378      }
         1379      }
         1380      }
         1381      }
         1382      }
         1383      }
         1384      }
         1385      }
         1386      }
         1387      }
         1388      }
         1389      }
         1390      }
         1391      }
         1392      }
         1393      }
         1394      }
         1395      }
         1396      }
         1397      }
         1398      }
         1399      }
         1400      }
         1401      }
         1402      }
         1403      }
         1404      }
         1405      }
         1406      }
         1407      }
         1408      }
         1409      }
         1410      }
         1411      }
         1412      }
         1413      }
         1414      }
         1415      }
         1416      }
         1417      }
         1418      }
         1419      }
         1420      }
         1421      }
         1422      }
         1423      }
         1424      }
         1425      }
         1426      }
         1427      }
         1428      }
         1429      }
         1430      }
         1431      }
         1432      }
         1433      }
         1434      }
         1435      }
         1436      }
         1437      }
         1438      }
         1439      }
         1440      }
         1441      }
         1442      }
         1443      }
         1444      }
         1445      }
         1446      }
         1447      }
         1448      }
         1449      }
         1450      }
         1451      }
         1452      }
         1453      }
         1454      }
         1455      }
         1456      }
         1457      }
         1458      }
         1459      }
         1460      }
         1461      }
         1462      }
         1463      }
         1464      }
         1465      }
         1466      }
         1467      }
         1468      }
         1469      }
         1470      }
         1471      }
         1472      }
         1473      }
         1474      }
         1475      }
         1476      }
         1477      }
         1478      }
         1479      }
         1480      }
         1481      }
         1482      }
         1483      }
         1484      }
         1485      }
         1486      }
         1487      }
         1488      }
         1489      }
         1490      }
         1491      }
         1492      }
         1493      }
         1494      }
         1495      }
         1496      }
         1497      }
         1498      }
         1499      }
         1500      }
         1501      }
         1502     
```

# Cf OpenJdk ..

## src/hotspot/share/interpreter/bytecodes.hpp

bytecodes



Internal Reserved  
fast\_\* bytecodes

bytecodes.hpp			
219	_return	= 177, // 0xb1	
220	_getstatic	= 178, // 0xb2	
221	_putstatic	= 179, // 0xb3	
222	_getfield	= 180, // 0xb4	
223	_putfield	= 181, // 0xb5	
224	_invokevirtual	= 182, // 0xb6	
225	_invokespecial	= 183, // 0xb7	
226	_invokestatic	= 184, // 0xb8	
227	_invokeinterface	= 185, // 0xb9	
228	_invokedynamic	= 186, // 0xba	
229	_new	= 187, // 0xbb	

// JVM bytecodes

\_fast\_agetfield  
\_fast\_bgetfield  
\_fast\_cgetfield  
\_fast\_dgetfield  
\_fast\_fgetfield  
\_fast\_igetfield  
\_fast\_lgetfield  
\_fast\_sgetfield

\_fast\_awaitfield  
\_fast\_bputfield  
\_fast\_zputfield  
\_fast\_cputfield  
\_fast\_dputfield  
\_fast\_fputfield  
\_fast\_iputfield  
\_fast\_lputfield  
\_fast\_sputfield

\_fast\_aload\_0  
\_fast\_iaccess\_0  
\_fast\_aaccess\_0  
\_fast\_faccess\_0

\_fast\_ildload  
\_fast\_ildload2  
\_fast\_icaload

\_fast\_invokevfinal  
\_fast\_linearswitch  
\_fast\_binaryswitch

# Idem « invoke\* » => « fast\_invoke\* »

```
public static void testFooMethod();
Code:
  0: new          #30          // class test/Foo
  3: dup
  4: invokespecial #32          // Method test/Foo."<init>":()V
  7: astore_0
  8: aload_0
  9: invokevirtual #65          // Method test/Foo.f:()V
 12: return
```



## On First use....

Resolve Bytecode « invoke\* #methIdx »

Load + type-check

```
Class c = .. Class.forName(« test.Bar »)
m = c.getMethod(« field1(type1..typeN »);
assert m.getType().equals(...);
```



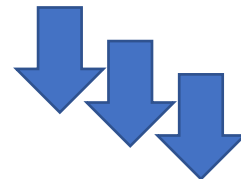
Bytecode instruction

« invoke\* #idx »

**HOT REPLACED BY** internal

« **\_fast\_invoke\* offset** »

Code replaced (?) or re-executed  
with « throw new ...Error() »



« fast\_invoke\* » on next uses

# Different « invoke\* » : {static | special | virtual | interface | dynamic}

**static** => ... to call fixed (known) function, and update stack

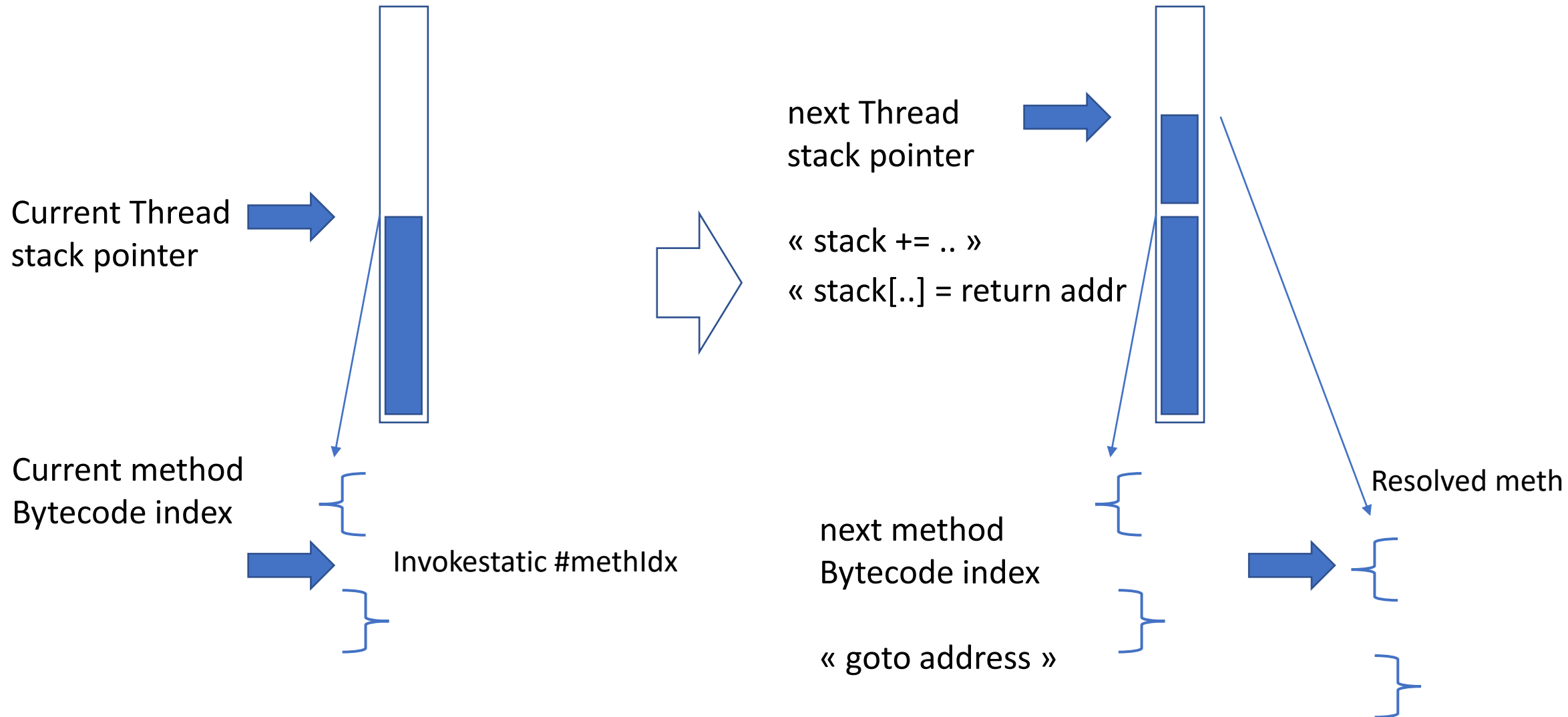
**special** => ... idem ... call fixed (known) function, and update stack  
after « new »: « <init> » method, or « super() » method

**virtual** => ... need array access to object class « virtual table », to determine exact method

**interface** => ... need lookup interfaces table ... then virtual table, to determine method

**dynamic** => ... internal for JRE, allowing type evolution

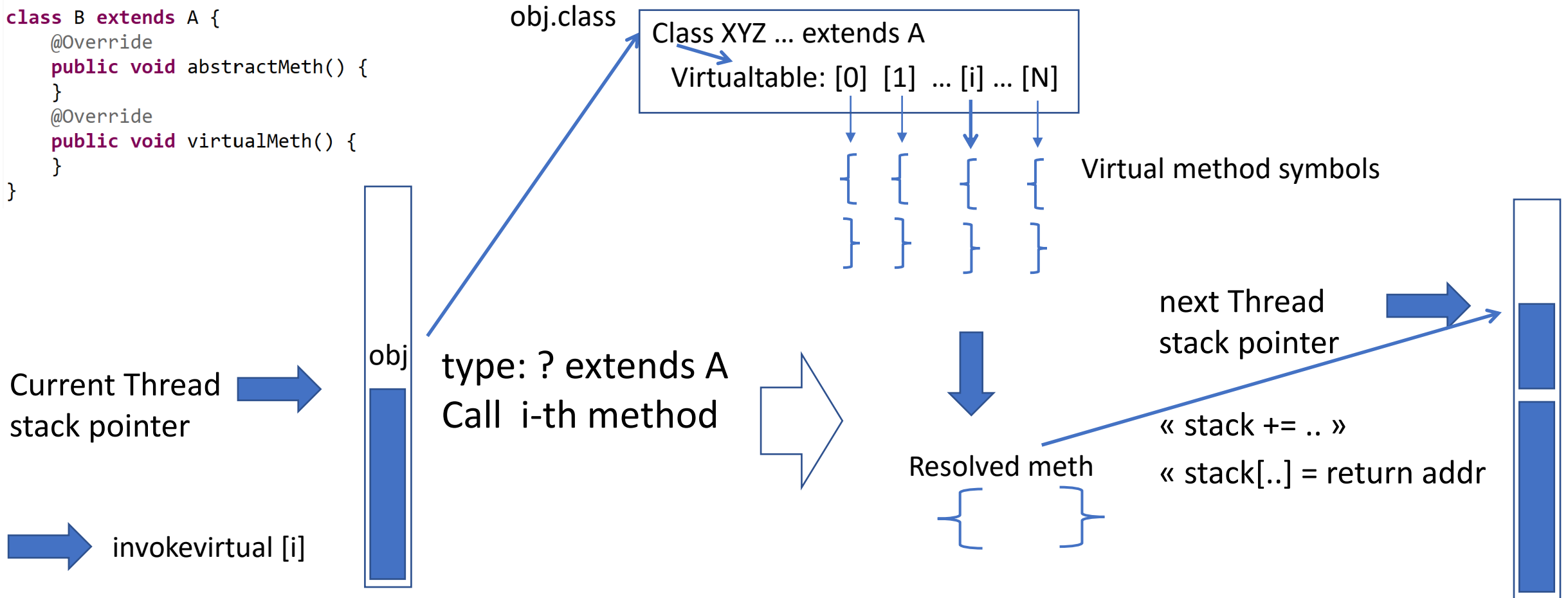
# invokestatic



# invokevirtual

```
public abstract class A {  
    public abstract void abstractMeth();  
    public void virtualMeth() {  
        // maybe overridden  
    }  
}
```

```
class B extends A {  
    @Override  
    public void abstractMeth() {  
    }  
    @Override  
    public void virtualMeth() {  
    }  
}
```





# invokevirtual .. slower than invokestatic

Invokestatic ....  $O(1 \text{ call})$

Invokevirtual ....  $O(\mathbf{1 \text{ array access} + \text{push 1 extra param} \ll \text{this} \gg + 1 \text{ call}})$

# invokeinterface

```
public interface IA {  
    public void meth();  
    public default void defaultMeth() {  
        // maybe overridden  
    }  
}
```

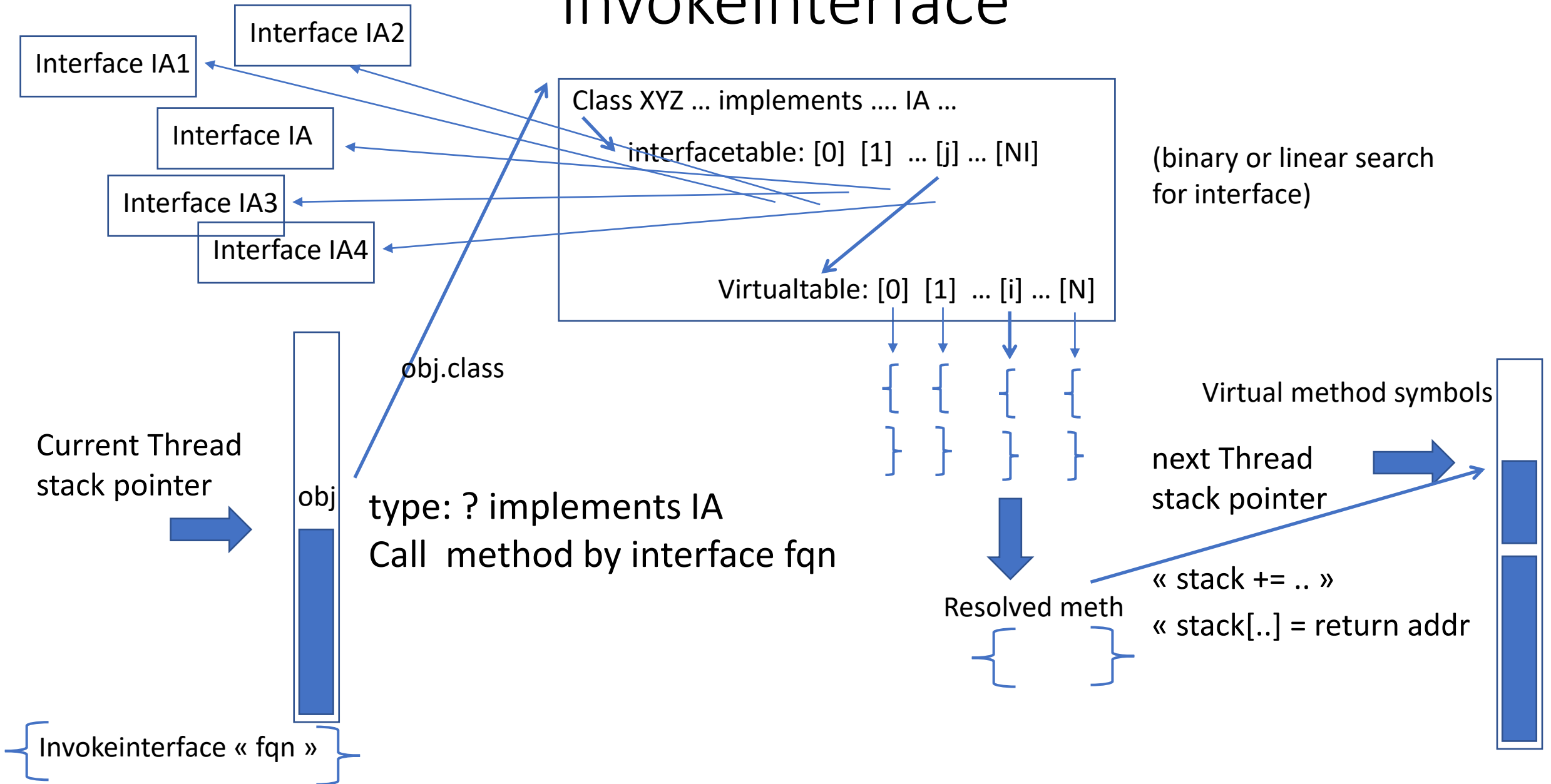
```
class B extends Foo implements IA1, IA2, IA3, IA, IA4, IA5 {  
    //  
    //  
    //  
    @Override  
    public void meth() {  
    }  
    @Override  
    public void defaultMeth() {  
    }  
}
```

```
IA anyObj = new B();  
anyObj.meth(); // <= invokeinterface
```

Performance Problem :

no relation between interface « IA » and extends class (here: « Foo »)  
.. can not use virtualtable directly

# invokeinterface



# invokeinterface .. slower than invokevirtual

Invokestatic ....  $O(1 \text{ call})$

Invokevirtual ....  $O(1 \text{ array access} + \text{push } 1 \text{ extra param « this »} + 1 \text{ call})$

Invokeinterface ....  $O(1 \text{ pointer access}$   
+  $J$  linear OR  $\log(J)$  binary search interface tables  
+ 1 virtual table array access  
+ push 1 extra param « this » + 1 call )

# Remark: abstract class vs interface

For performance (with millions of calls...):

1/ Prefer declare virtual method in abstract class  
Rather than method in interface

2/ do not use too many interfaces

3/ sort « implements » interfaces by most frequent usage first  
(? .. If linear search )

Example: cf in JDK ...

```
public abstract class InputStream { public abstract int read(); }  
public abstract class OutputStream { public abstract void write(int data); }
```

# ArrayList

extends AbstractList (..AbstractCollection)

implement List

```
// 1:
List<Integer> ls1 = new ArrayList<Integer>();
ls1.add(e); // invokeinterface #41, 2 // InterfaceMethod java/util/List.add:(Ljava/lang/Object;)Z

// 2:
AbstractList<Integer> ls2 = new ArrayList<Integer>();
ls2.add(e); // invokevirtual #51 // Method java/util/AbstractList.add:(Ljava/lang/Object;)Z

// 3:
ArrayList<Integer> ls3 = new ArrayList<Integer>();
ls3.add(e); // invokevirtual #57 // Method java/util/ArrayList.add:(Ljava/lang/Object;)Z

// 4:
val ls4 = new ArrayList<Integer>(); // lombok val, or jdk>=15 var
ls4.add(e); // idem ArrayList .. invokevirtual
```

1: Slower than 2:,3:,4:

2: & 3: same perf (both invokevirtual, not same type)

4: idem 3:

HOTSPOT may optimize all as invokespecial !!

# Declarations List, ArrayList, ..

```
public interface Collection<E> extends Iterable<E> {
    boolean add(E e);
}

public abstract class AbstractCollection<E> implements Collection<E> {
    @Override
    public boolean add(E e) {
        throw new UnsupportedOperationException();
    }
}

public interface List<E> extends Collection<E> {
    boolean add(E e);
}

public abstract class AbstractList<E> extends AbstractCollection<E> implements List<E> {
    @Override
    public boolean add(E e) {
        throw new UnsupportedOperationException();
    }
}

public class ArrayList<E> extends AbstractList<E>
    implements List<E>, RandomAccess, Cloneable, java.io.Serializable {

    @Override
    public boolean add(E e) { ..
        ..
    }
```

# Miranda Methods

(implicit abstract methods from interface)

```
public interface IFoo {  
    void foo();  
}  
  
public abstract class AbstractFoo implements IFoo {  
    //... implicitly miranda method  
    // public @Override abstract void foo();  
}  
  
public final class Foo extends AbstractFoo {  
    @Override  
    public void foo() {  
    }  
}
```

```
IFoo f1 = new Foo();  
f1.foo(); // invoke interface
```

```
AbstractFoo f2 = new Foo();  
f2.foo(); // invoke interface ?? => invoke virtual !!  
           // invokevirtual #32                      // Method test/TestMiranda$AbstractFoo.foo:()V
```

```
Foo f3 = new Foo();  
f3.foo(); // invoke virtual
```



Questions ?