canoo
› your provider for business web solutions ›

AST Transformations

Hamlet D'Arcy, Canoo Engineering AG
@HamletDRC
http://hamletdarcy.blogspot.com

Agenda

Lombok and Groovy AST Transformations
– *Abstract Syntax Trees*

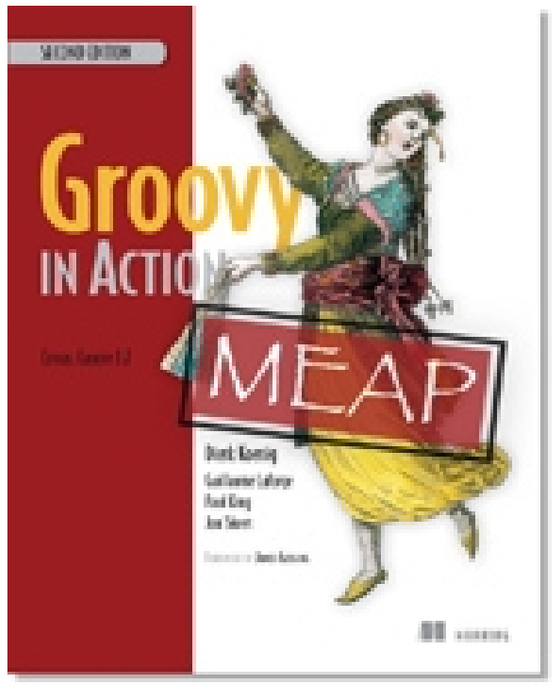
CodeNarc and Groovy 2.0
– *Static Analysis and Type Checking*

Mirah
– *Macros and Static Typing*

About Me


Groovy


CODENARC
Less Bugs Better Code


Groovy
IN ACTION
MEAP


JETBRAINS
DEVELOPMENT ACADEMY
MEMBER

I shot an elephant in my pajamas.

Subject:
I

Verb:
shot

Direct Object:
an elephant

Indirect Object:
in my pajamas

I shot an elephant in my pajamas.

How he got in my pajamas,

I'll never know.

Subject:
I

Verb:
shot

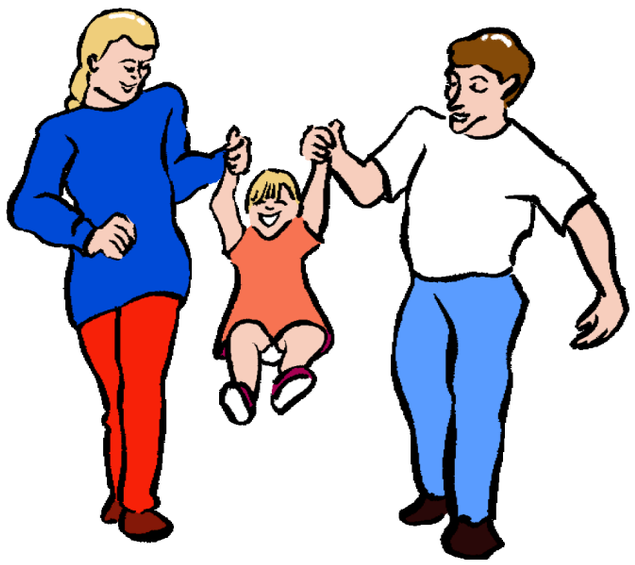
Participle Phrase

an elephant

in my pajamas

I want to thank my parents, Jesus and Oprah Winfrey

I want to thank my parents, Jesus and Oprah Winfrey



Subject:
I

Verb:
want

Infinitive:
to thank

Participle:

my parents

God

Oprah Winfrey

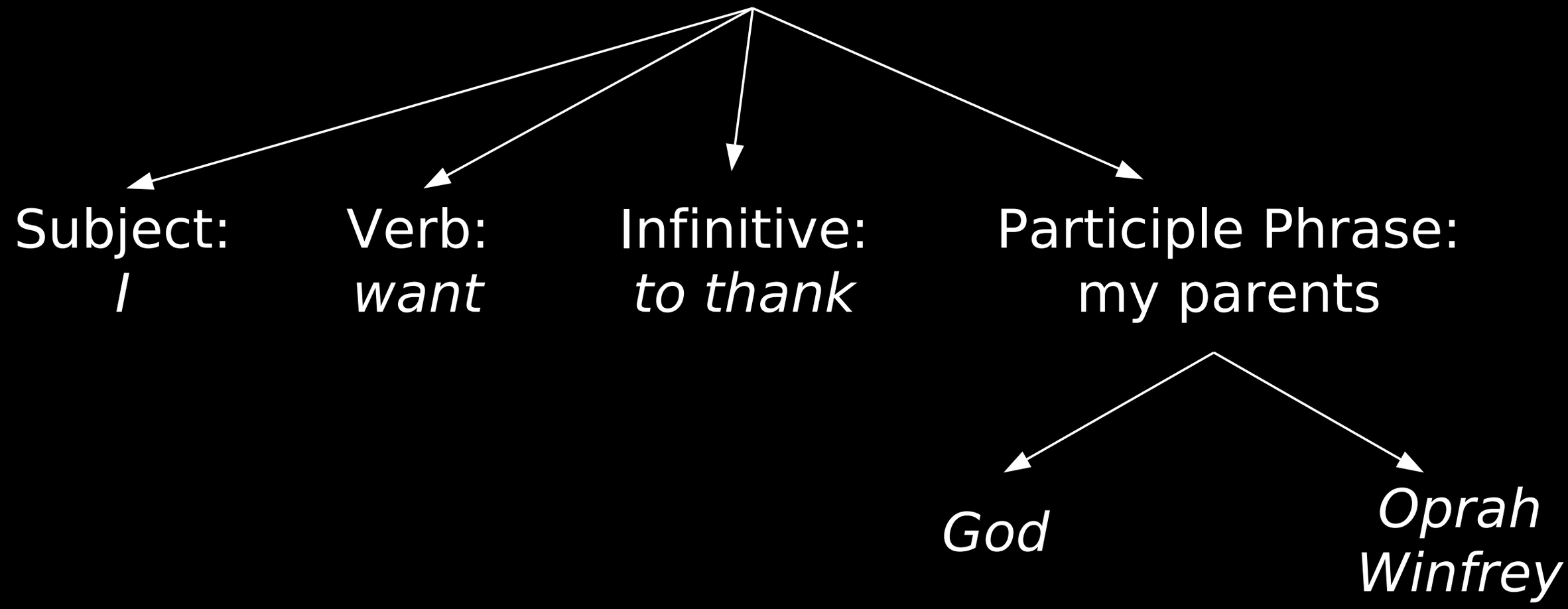
I want to thank my parents, Jesus and Oprah Winfrey

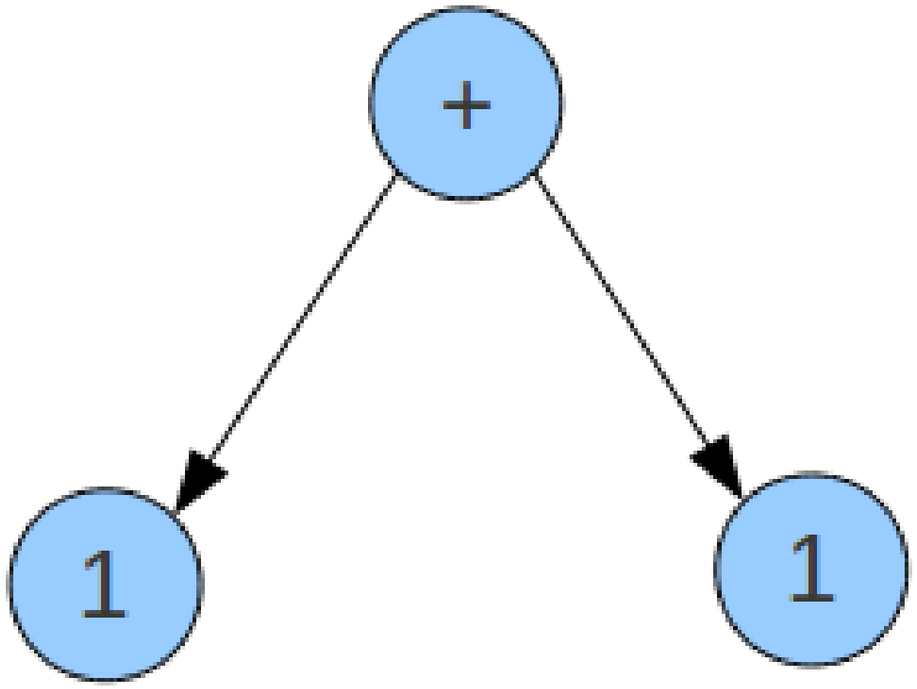
I want to thank my parents, Jesus and Oprah Winfrey

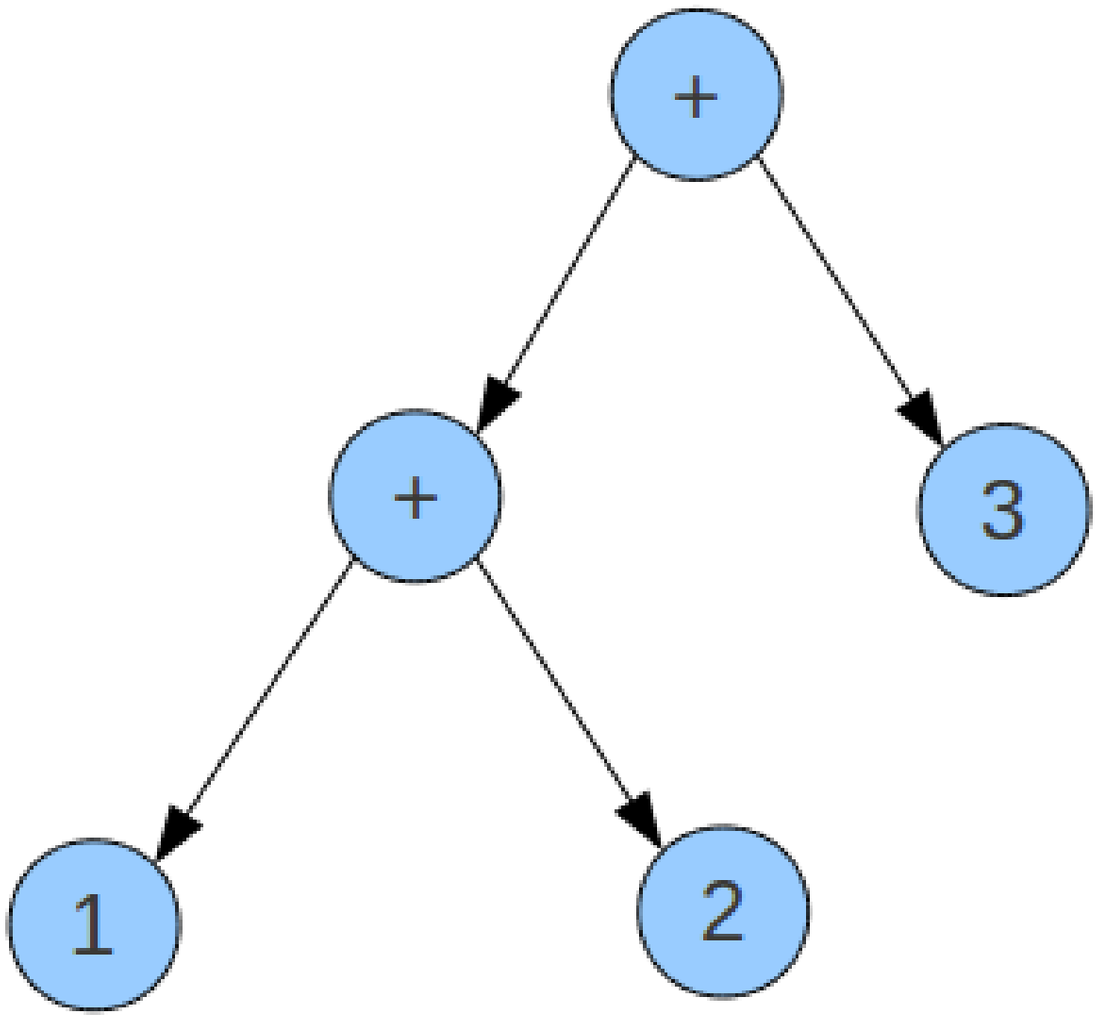
God
b. ?

You
b. 1954

You
b. 1976

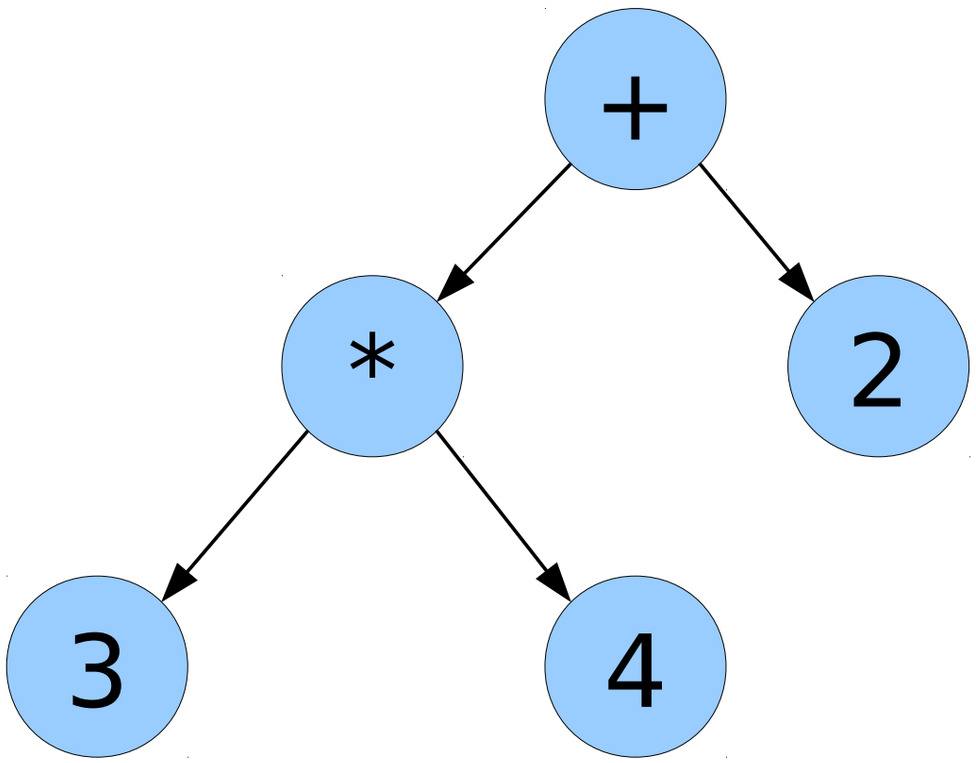

Subject:
I
Verb:
want
Infinitive:
to thank
Participle Phrase:
my parents
God
Oprah Winfrey


+
1
1

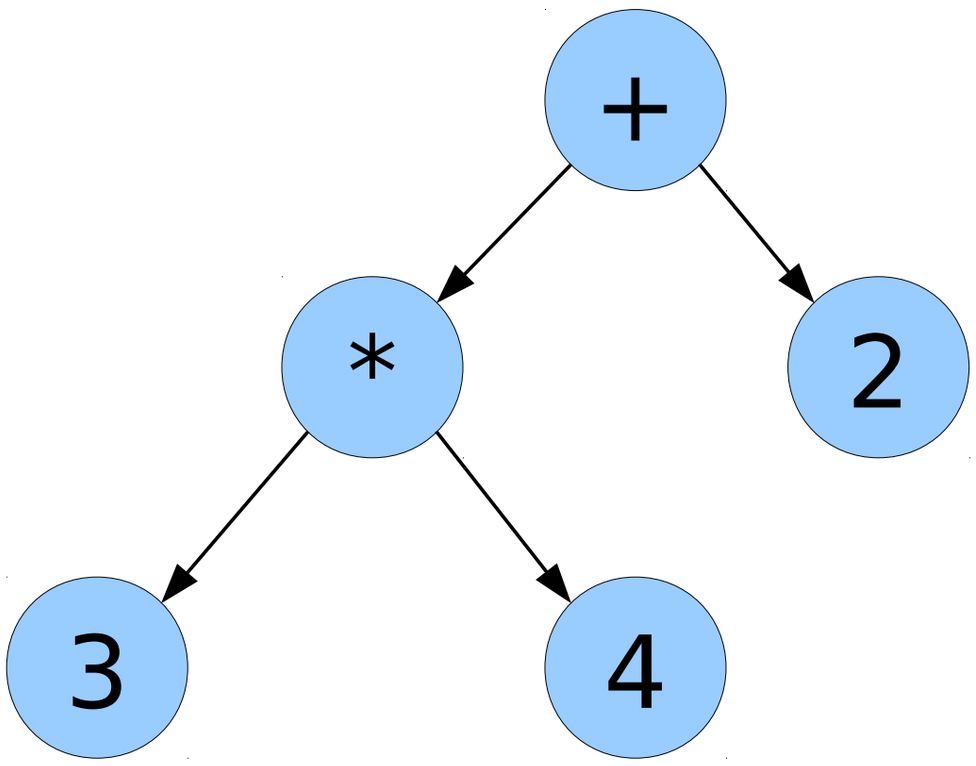

+
+
3
1
2

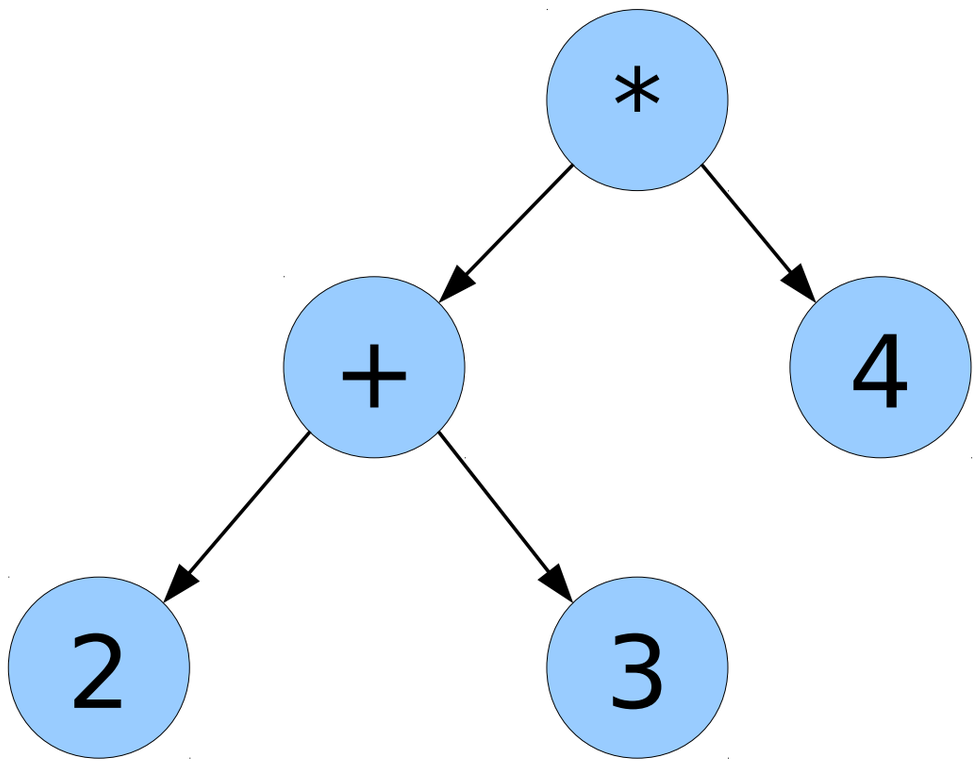
2 + 3 * 4

2 + 3 * 4

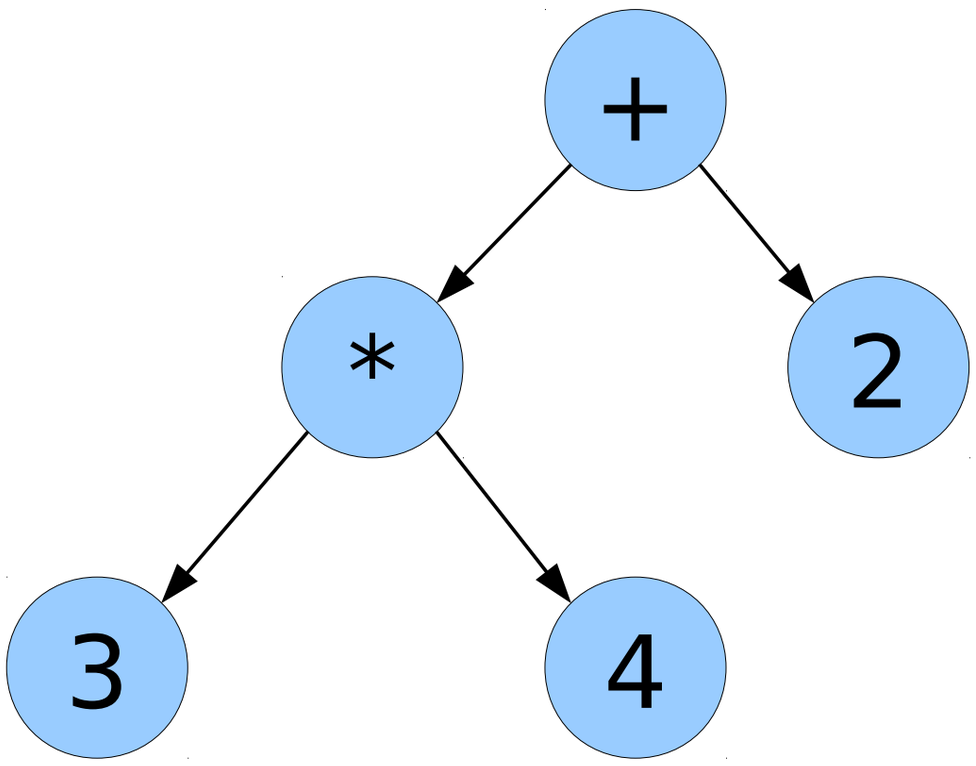


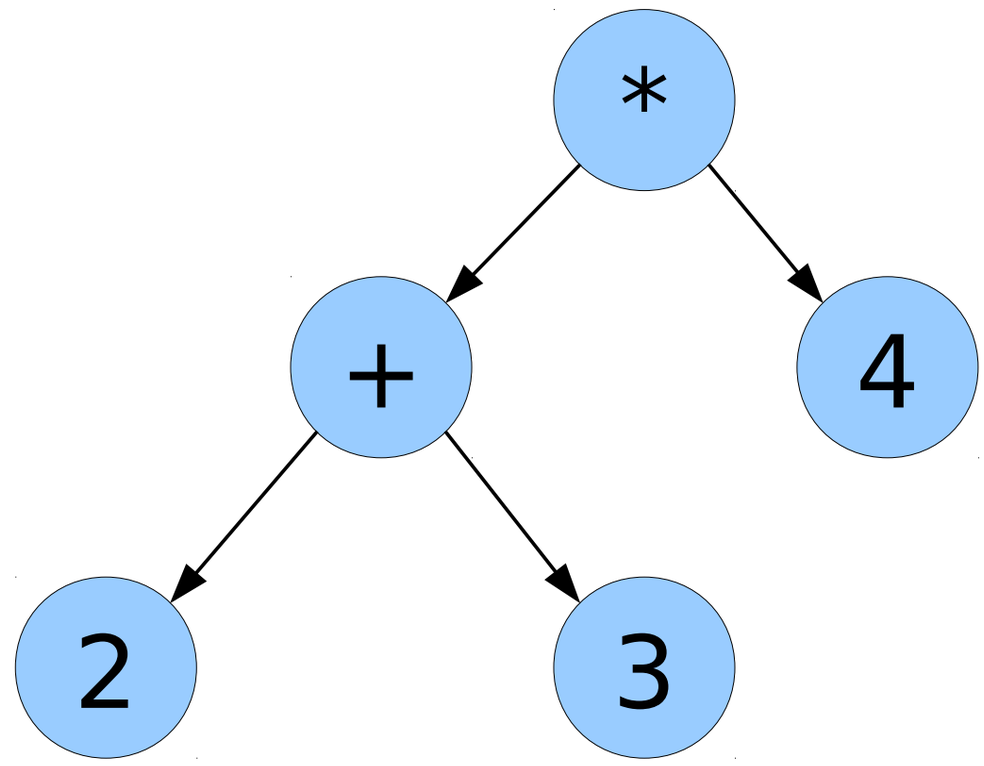
2 + 3 * 4

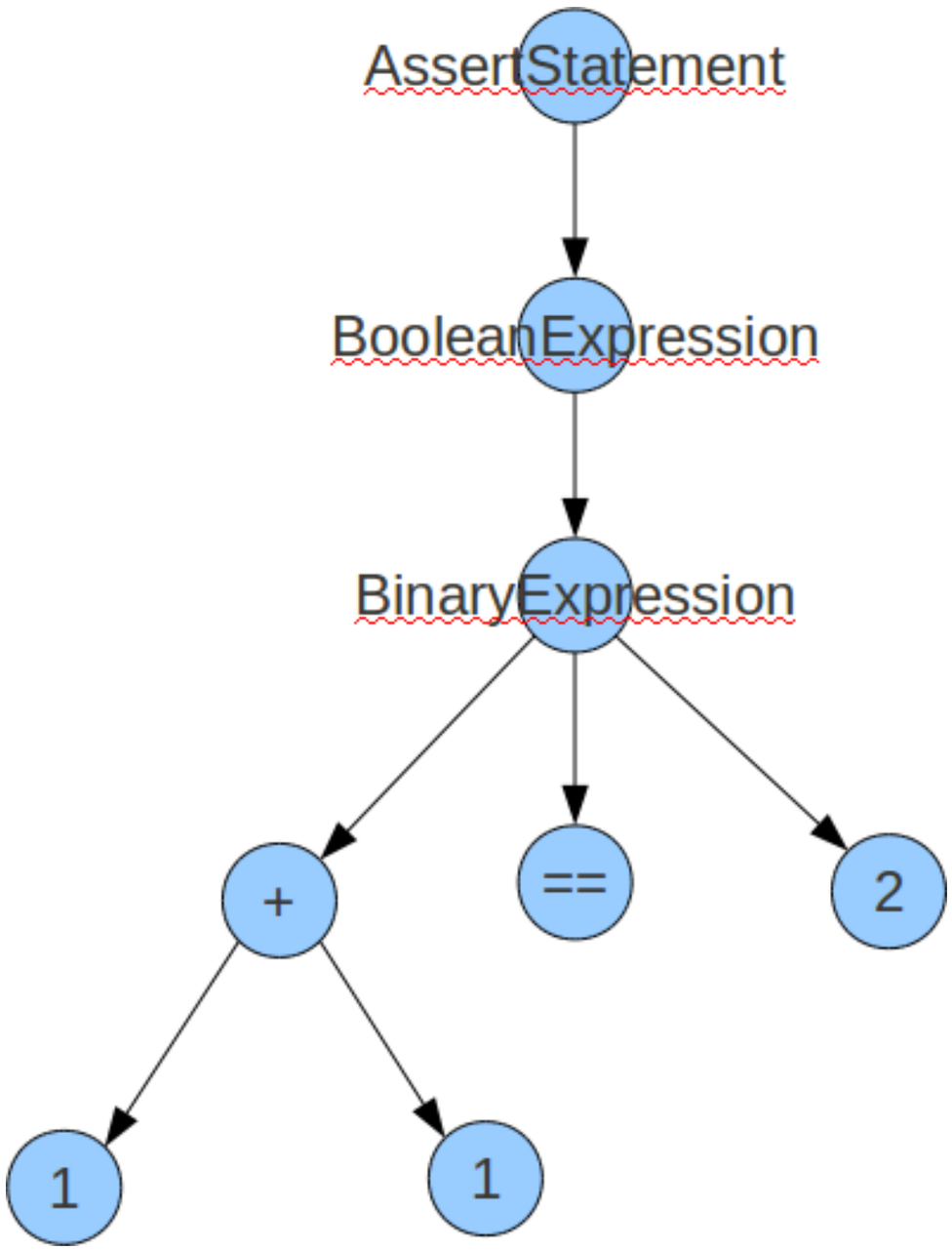




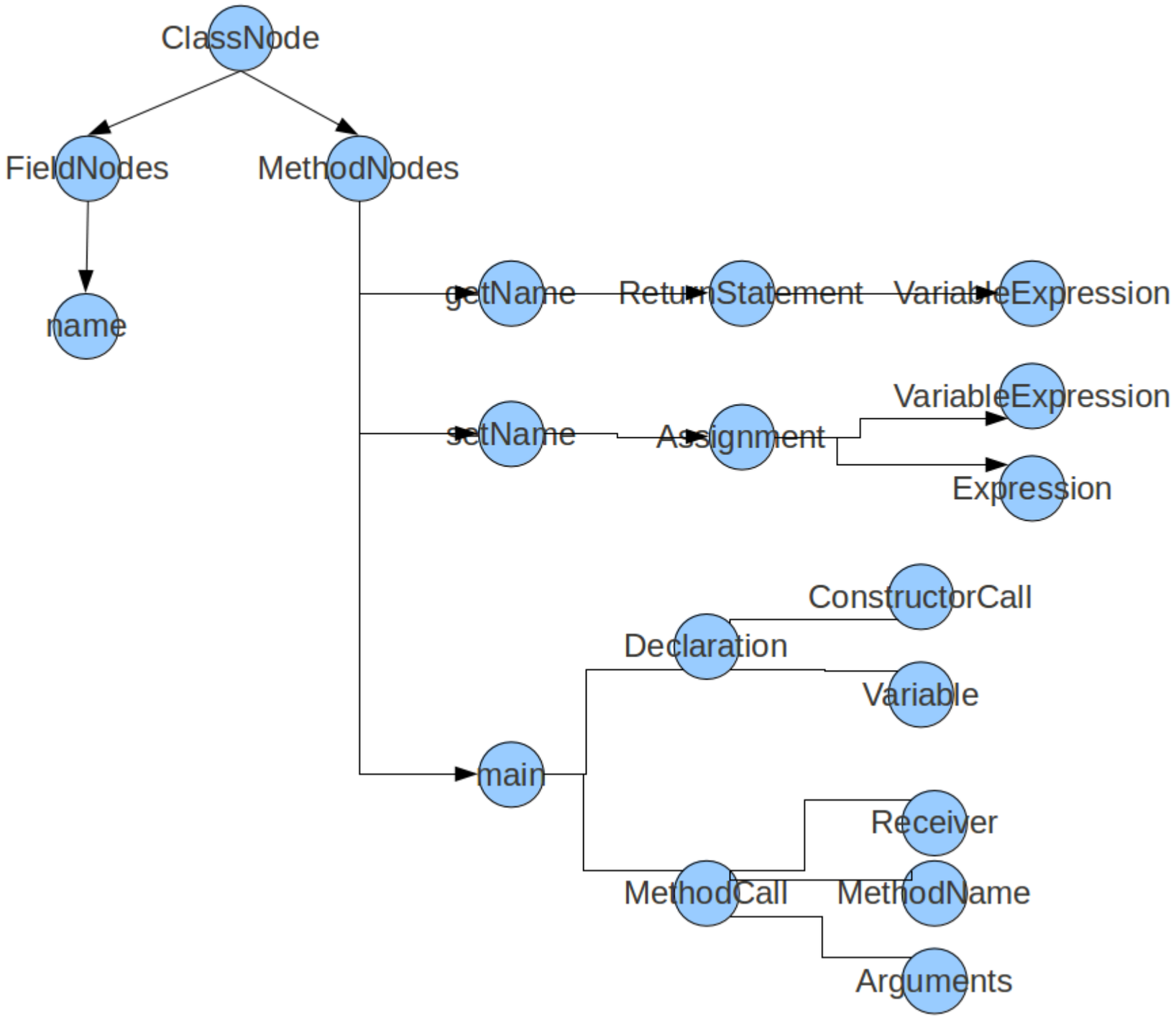
(+ 2 (* 3 4))



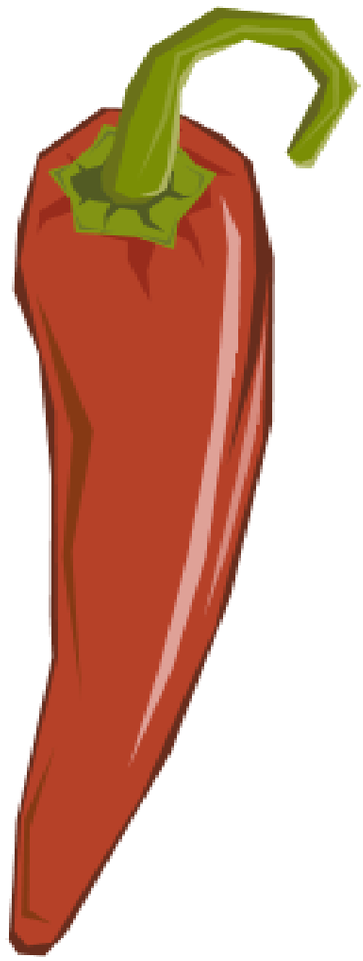



AssertStatement
BooleanExpression
BinaryExpression
+
==
2
1
1

```
public class Person {
    private String name;

    public void setName(String name) {
        this.name = name;}

    public String getNameName() {
        return name;
    }

    public static void main(String[] args) {
        Person p = new Person();
        p.setName(“Hamlet”);
        System.out.println(p);
    }
}
```


ClassNode
FieldNodes
MethodNodes
name
getName
ReturnStatement
VariableExpression
setName
Assignment
VariableExpression
Expression
main
Declaration
ConstructorCall
Variable
MethodCall
Receiver
MethodName
Arguments

Project Lombok



Person.java

```
1  package lombok;
2
3  public class Person {
4
5      private String firstName;
6      private String lastName;
7  }
8
9
```

Terminal

File Edit View Terminal Help

```
hamlet@fairytopia:~lombok $ javap Person
Compiled from "Person.java"
public class lombok.Person extends java.lang.Object{
    public lombok.Person();
}

hamlet@fairytopia:~lombok $
```

Person.class

```
package lombok;

public class Person
{
   private String firstName;
   private String lastName;
}
```

Person.java

```
1  package lombok;
2
3  public class Person {
4
5      @Getter @Setter private String firstName;
6      @Getter @Setter private String lastName;
7  }
8
9
```

Main.java

```
1   package lombok;
2
3   public class Main {
4
5       public static void main(String[] args) {
6
7           new Person().get*Na
8
9       }
10  }
11
```

getFirstName() String
getLastName() String
Use Ctrl+Shift+Enter to syntactically correct your code after completing (balance parentheses etc.)

Terminal

File Edit View Terminal Help

```
hamlet@fairytopia:~lombok $ javap Person
Compiled from "Person.java"
public class Person extends java.lang.Object{
    public Person();
    public java.lang.String getFirstName();
    public void setFirstName(java.lang.String);
    public java.lang.String getLastName();
    public void setLastName(java.lang.String);
}
```

Person.class

```
public class Person
{
  private String firstName;
  private String lastName;

  public String getFirstName()
  {
    return this.firstName;
  }

  public void setFirstName(String paramString)
  {
    this.firstName = paramString;
  }

  public String getLastName()
  {
    return this.lastName;
  }

  public void setLastName(String paramString)
  {
    this.lastName = paramString;
  }
}
```

Person.java

```
1  package lombok;
2
3  @EqualsAndHashCode
4  public class Person {
5
6      @Getter @Setter private String firstName;
7      @Getter @Setter private String lastName;
8  }
9
10
```

Terminal

File Edit View Terminal Help

```
hamlet@fairytopia:~lombok $ javap Person
Compiled from "Person.java"
public class Person extends java.lang.Object{
    public Person();
    public boolean equals(java.lang.Object);
    public int hashCode();
    public java.lang.String getFirstName();
    public void setFirstName(java.lang.String);
    public java.lang.String getLastName();
    public void setLastName(java.lang.String);
}
```

Person.class

```
public class Person
{
  private String firstName;
  private String lastName;

  public boolean equals(Object paramObject)
  {
    if (paramObject == this)
      return true;
    if (paramObject == null)
      return false;
    if (paramObject.getClass() != getClass())
      return false;
    Person localPerson = (Person)paramObject;
    if (this.firstName == null ? localPerson.firstName != null : !this.firstName.equals(localPerson.firstName))
      return false;
    return this.lastName == null ? localPerson.lastName == null : this.lastName.equals(localPerson.lastName);
  }

  public int hashCode()
  {
    int i = 1;
    i = i * 31 + (this.firstName == null ? 0 : this.firstName.hashCode());
    i = i * 31 + (this.lastName == null ? 0 : this.lastName.hashCode());
    return i;
  }
```

Person.java

```
1  package lombok;
2
3  @Data
4  public class Person {
5
6      private String firstName;
7      private String lastName;
8  }
```

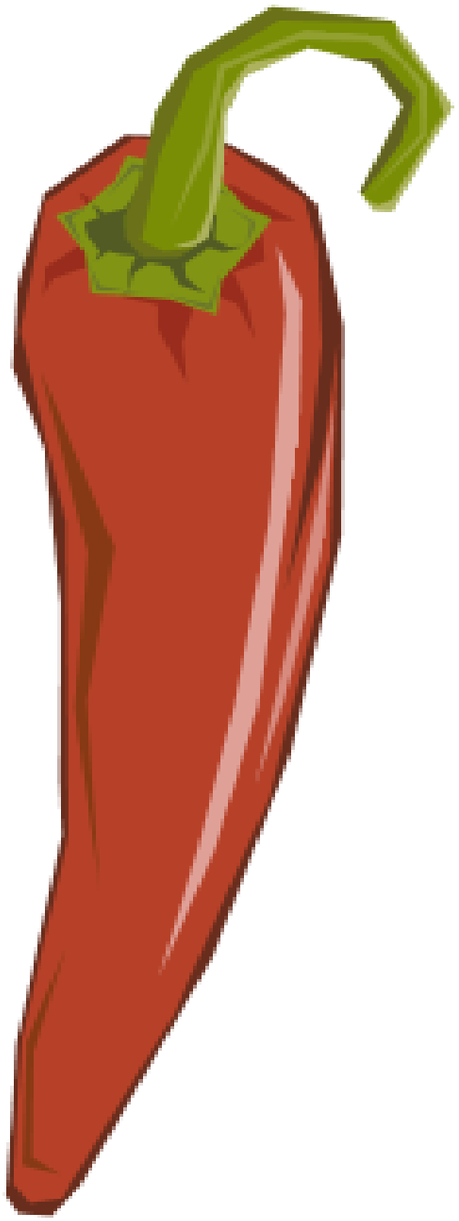
Terminal

File Edit View Terminal Help

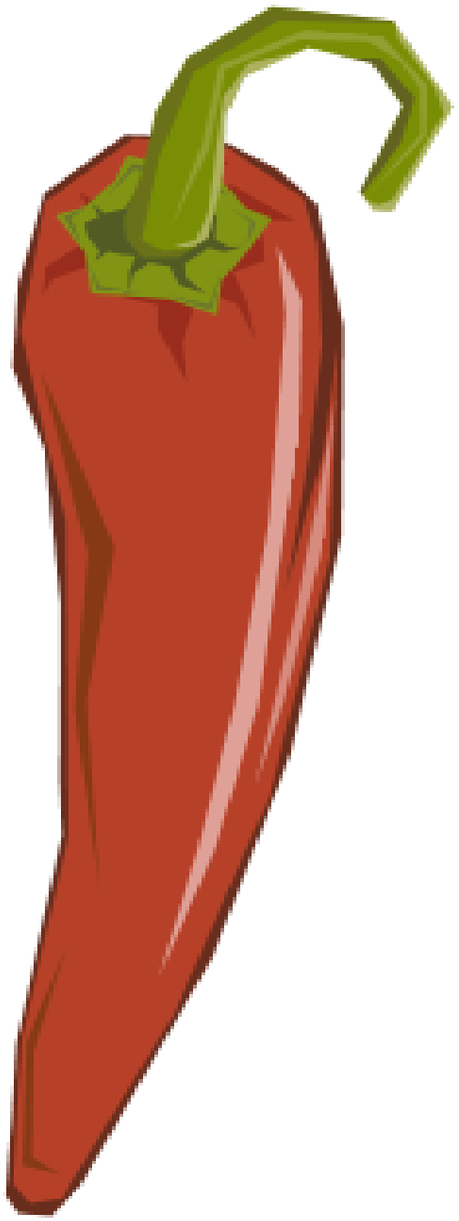
```
hamlet@fairytopia:~lombok $ javap Person
Compiled from "Person.java"
public class Person extends java.lang.Object{
    public Person();
    public java.lang.String getFirstName();
    public void setFirstName(java.lang.String);
    public java.lang.String getLastName();
    public void setLastName(java.lang.String);
    public boolean equals(java.lang.Object);
    public int hashCode();
    public java.lang.String toString();
}
```

```
@Data
public class Person {
  private String firstName;
  private String lastName;
}
```

```
public class Person {

  private String firstName;
  private String lastName;

  public String getFirstName() {
    return this.firstName;
  }

  public void setFirstName(String paramString) {
    this.firstName = paramString;
  }

  public String getLastName() {
    return this.lastName;
  }

  public void setLastName(String paramString) {
    this.lastName = paramString;
  }

  public boolean equals(Object paramObject) {
    if (paramObject == this)
      return true;
    if (paramObject == null)
      return false;
    if (paramObject.getClass() != getClass())
      return false;
    Person localPerson = (Person) paramObject;
    if (this.firstName == null ? localPerson.firstName != null
        : !this.firstName.equals(localPerson.firstName))
      return false;
    return this.lastName == null ? localPerson.lastName == null
        : this.lastName.equals(localPerson.lastName);
  }

  public int hashCode() {
    int i = 1;
    i = i * 31 + (this.firstName == null ? 0 : this.firstName.hashCode());
    i = i * 31 + (this.lastName == null ? 0 : this.lastName.hashCode());
    return i;
  }

  public String toString() {
    return "Person(firstName="+this.firstName+", lastName="+this.lastName + ")";
  }
}
```



@Getter / @Setter
@ToString
@EqualsAndHashCode
@NoArgsConstructor
@RequiredArgsConstructor
@AllArgsConstructor
@Data
@Cleanup
@Synchronized
@SneakyThrows
@Log
@Delegate
val



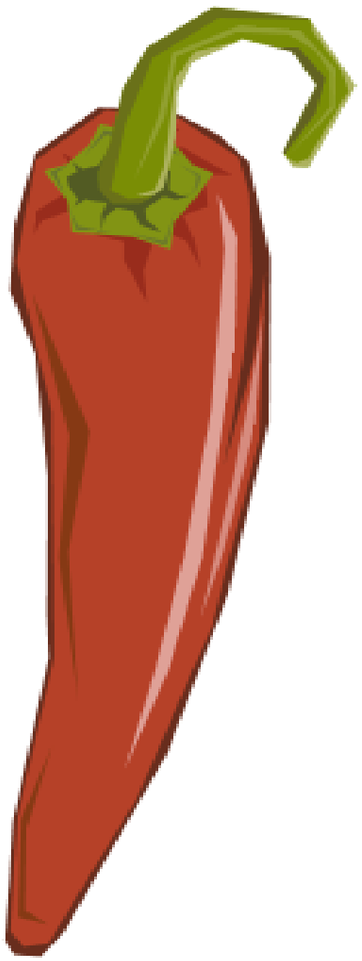
Generates Java Boilerplate

Compile Time Only
- For Eclipse and javac
- IDEA & NetBeans too

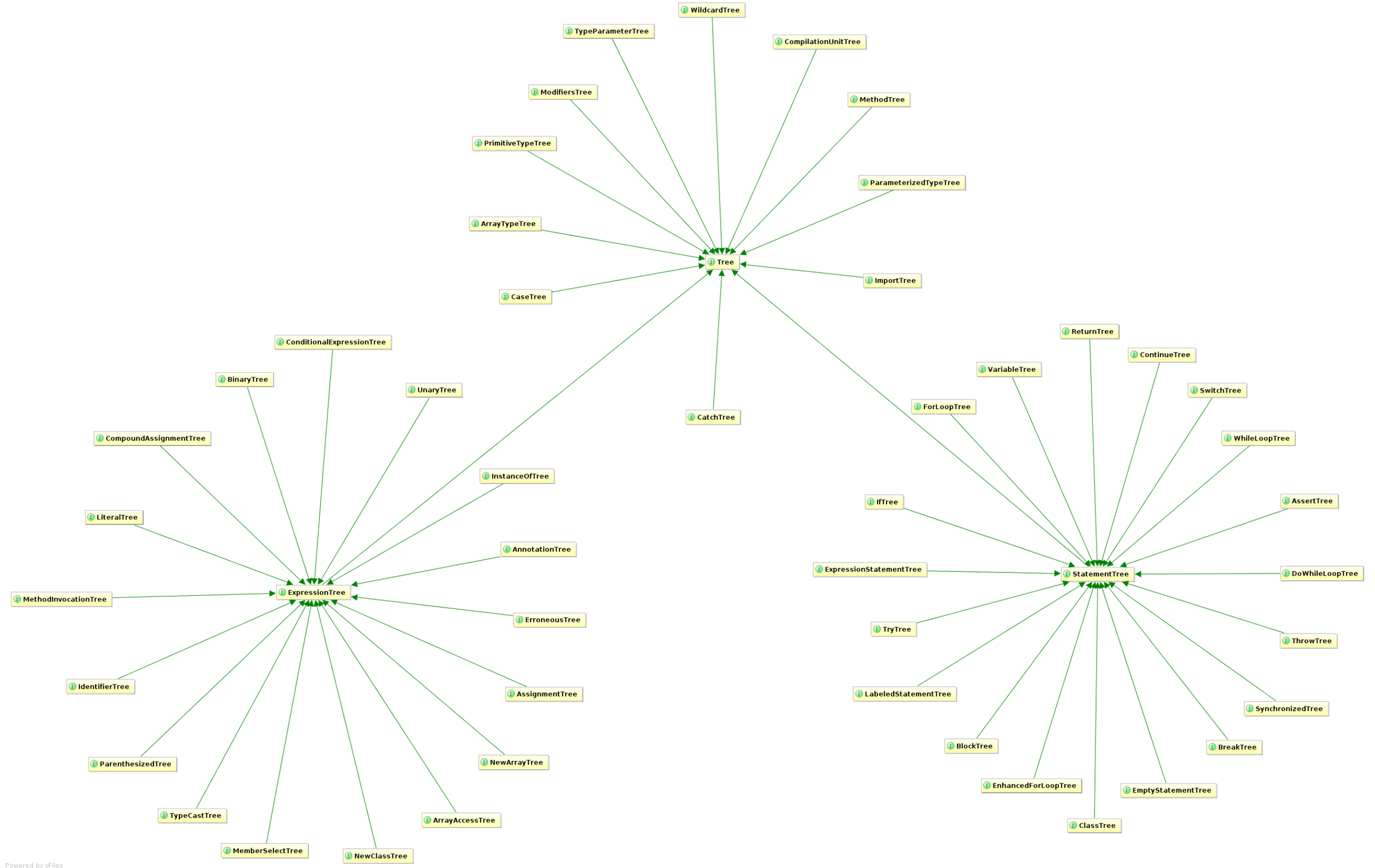
Removable with delombok
- Javadoc
- GWT?

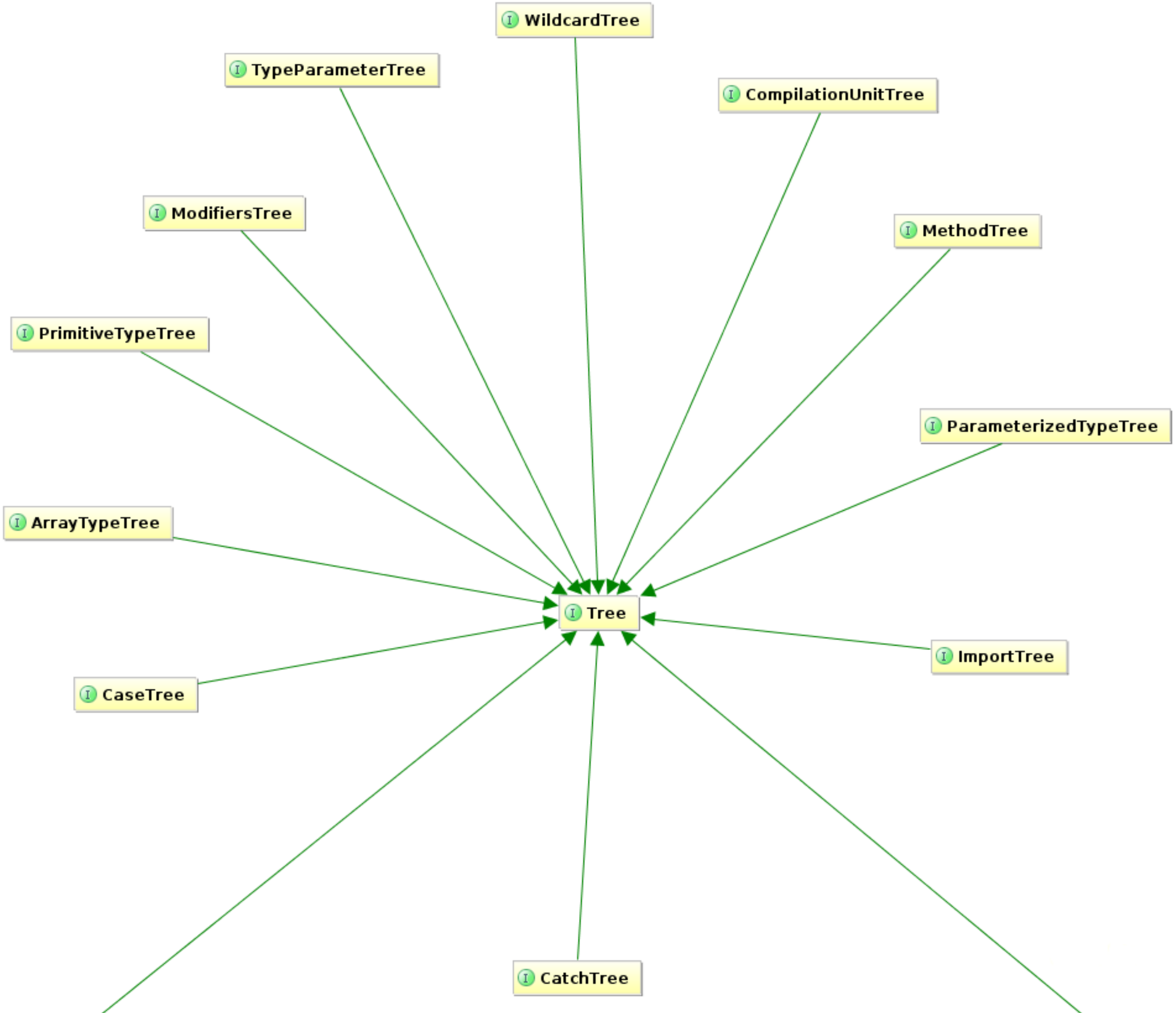
Read the fine print
- Know what is generated
- Slows compilation times?

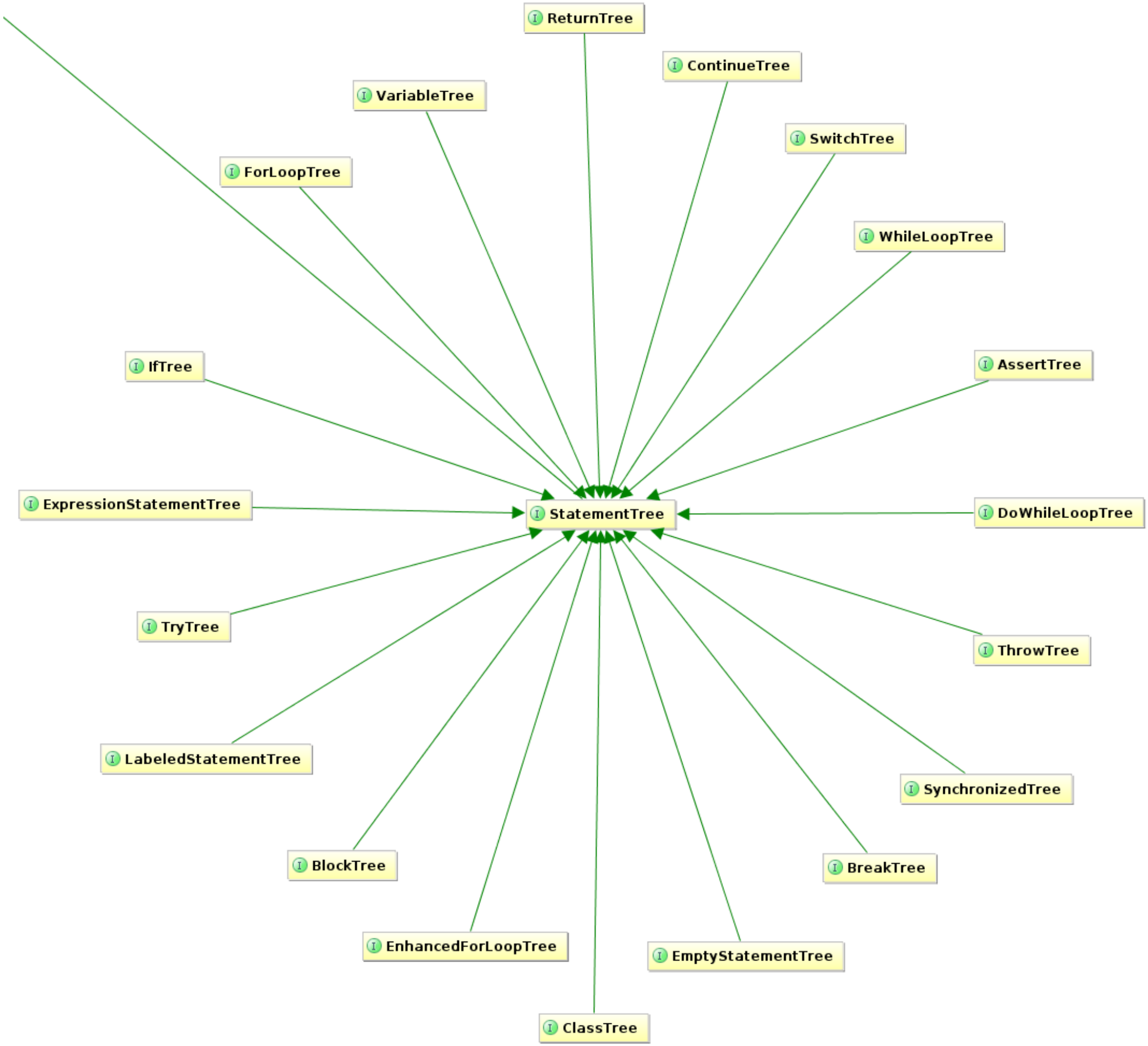
How it Works

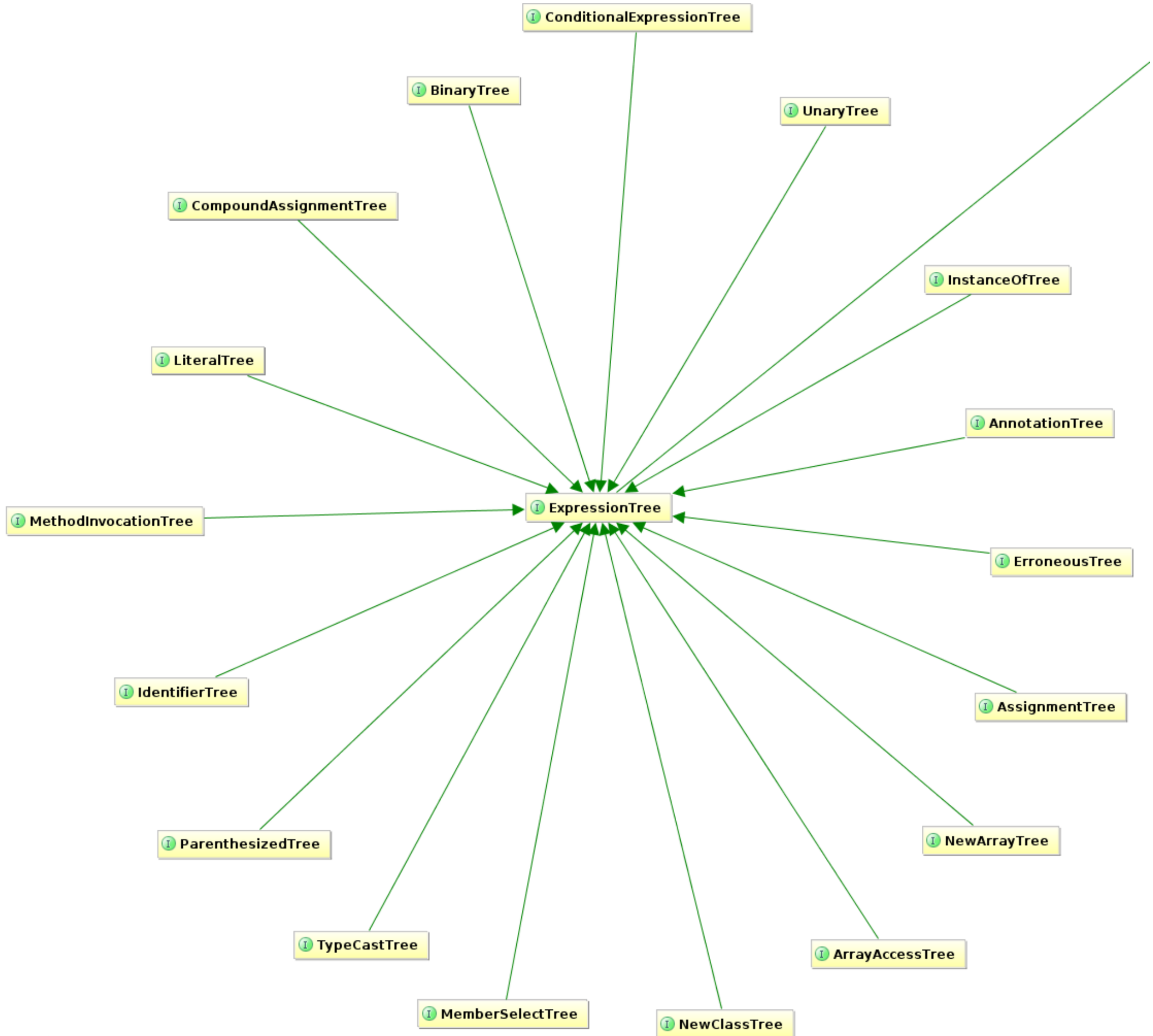


Annotation Processor
Javac Handler
Eclipse Handler




WildcardTree
TypeParameterTree
CompilationUnitTree
ModifiersTree
MethodTree
PrimitiveTypeTree
ParameterizedTypeTree
ArrayTypeTree
Tree
ImportTree
CaseTree
CatchTree


ReturnTree
ContinueTree
VariableTree
SwitchTree
ForLoopTree
WhileLoopTree
IfTree
AssertTree
ExpressionStatementTree
StatementTree
DoWhileLoopTree
TryTree
ThrowTree
LabeledStatementTree
SynchronizedTree
BlockTree
BreakTree
EnhancedForLoopTree
EmptyStatementTree
ClassTree



```
// javac
private void generatePropertyChangeSupportField(JavacNode node) {
	if (fieldAlreadyExists(PROPERTY_SUPPORT_FIELD_NAME, node)) return;
	JCExpression exp = chainDots(node.getTreeMaker(), node, "this");
	JCVariableDecl fieldDecl = newField()
			.ofType(PropertyChangeSupport.class)
			.withName(PROPERTY_SUPPORT_FIELD_NAME)
			.withModifiers(PRIVATE | FINAL)
			.withArgs(exp)
			.buildWith(node);
	injectField(node, fieldDecl);
}

// ECJ
private void generatePropertyChangeSupportField(EclipseNode node) {
	if (fieldAlreadyExists(PROPERTY_SUPPORT_FIELD_NAME, node)) return;
	Expression exp = referenceForThis(node.get());
	FieldDeclaration fieldDecl = newField()
			.ofType(PropertyChangeSupport.class)
			.withName(PROPERTY_SUPPORT_FIELD_NAME)
			.withModifiers(PRIVATE | FINAL)
			.withArgs(exp)
			.buildWith(node);
	injectField(node, fieldDecl);
}
```

Alex Ruiz – Custom AST Transformations with Project Lombok

http://www.ibm.com/developerworks/java/library/j-lombok/


Groovy

```
class Event {
    String title
}
```

```
public class Event {
  String title

  public void getTitle() {
    title
  }

  public String setTitle(String t) {
    this.title = t
  }
}
```

```
class Event {
    @Delegate Date when
}
```

```
class Event implements Comparable, Clonable {
    Date when
    boolean after(Date when) { this.when.after(when) }
    boolean before(Date when)  { this.when.before(when) }
    Object clone() { this.when.clone() }
    int getDate()  { this.when.date }
    int getDay()  { this.when.day }
    int getHours()  { this.when.hours }
    int getMinutes()  { this.when.minutes }
    int getMonth()  { this.when.month }
    int getSeconds()  { this.when.seconds }
    long getTime()  { this.when.time }
    int getTimezoneOffset() { this.when.timezoneOffset }
    int getYear() { this.when.year }
    void setDate(int date) { this.when.date = date }
    void setHours(int hours)  { this.when.hours = hours }
    void setMonth(int month)  { this.when.month = month }
    void setTime(long time) { this.when.time = time }
    void setYear(int year)  { this.when.year = year }
    String toGMTString()  { this.when.toGMTString() }
    String toLocaleString() { this.when.toLocaleString() }
    void setSeconds(int seconds) {
        this.when.seconds = seconds
    }
    void setMinutes(int minutes)  {
        this.when.minutes = minutes
    }
    int compareTo(Date anotherDate)  {
        this.when.compareTo(otherDate)
    }
}
```

```
class Event {
  @Lazy ArrayList speakers
}
```

```
class Event {
  ArrayList speakers

  def getSpeakers() {
    if (speakers != null) {
      return speakers
    } else {
      synchronized(this) {
        if (speakers == null) {
          speakers = []
        }
        return speakers
      }
    }
  }
}
```

```
@Immutable
class Event {
   String title
}
```

– Class is final
– Properties must be @Immutable or effectively immutable
– Properties are private
– Mutatators throw ReadOnlyPropertyException
– Map constructor created
– Tuple constructor created
– Equals(), hashCode() & toString() created
– Dates, Clonables, & arrays are defensively copied on way in & out (but not deeply cloned)
– Collections & Maps are wrapped in Immutable variants
– Non-immutable fields force an error
– Special handling for Date, Color, etc
– Many generated methods configurable

Code Generation Transformations

@ToString

@EqualsAndHashCode

@Canonical

@Lazy

@InheritConstructors

@TupleConstructor

Class Design Annotations

@Delegate

@Singleton

@groovy.transform.Immutable

Logging Improvements

@Log

@Log4j

@Slf4j

@Commons

Declarative Concurrency

@Synchronized

@WithReadLock

@WithWriteLock

Easier Cloning and Externalizing

@AutoClone

@AutoExternalize

Safer Scripting

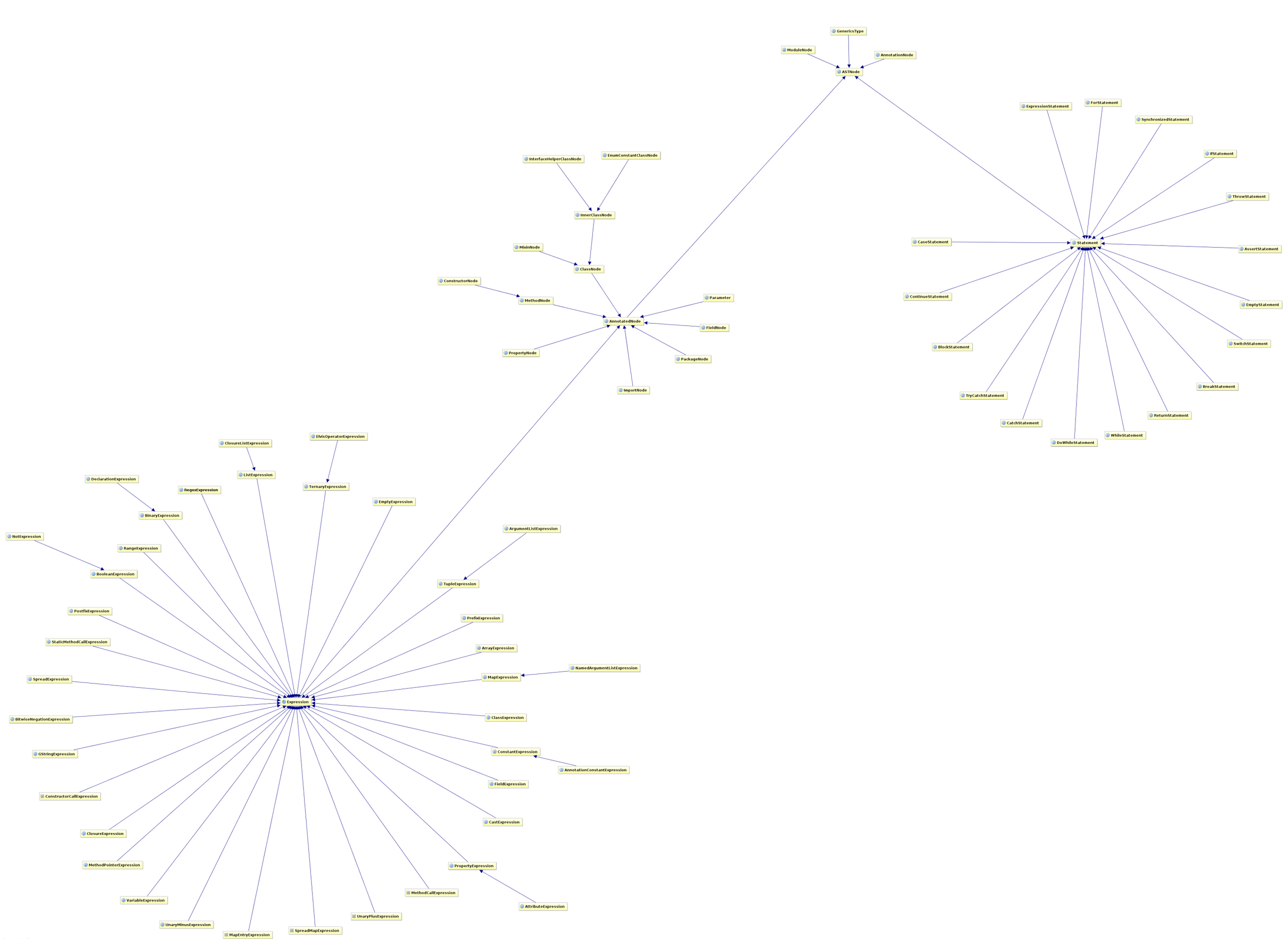
@TimedInterrupt

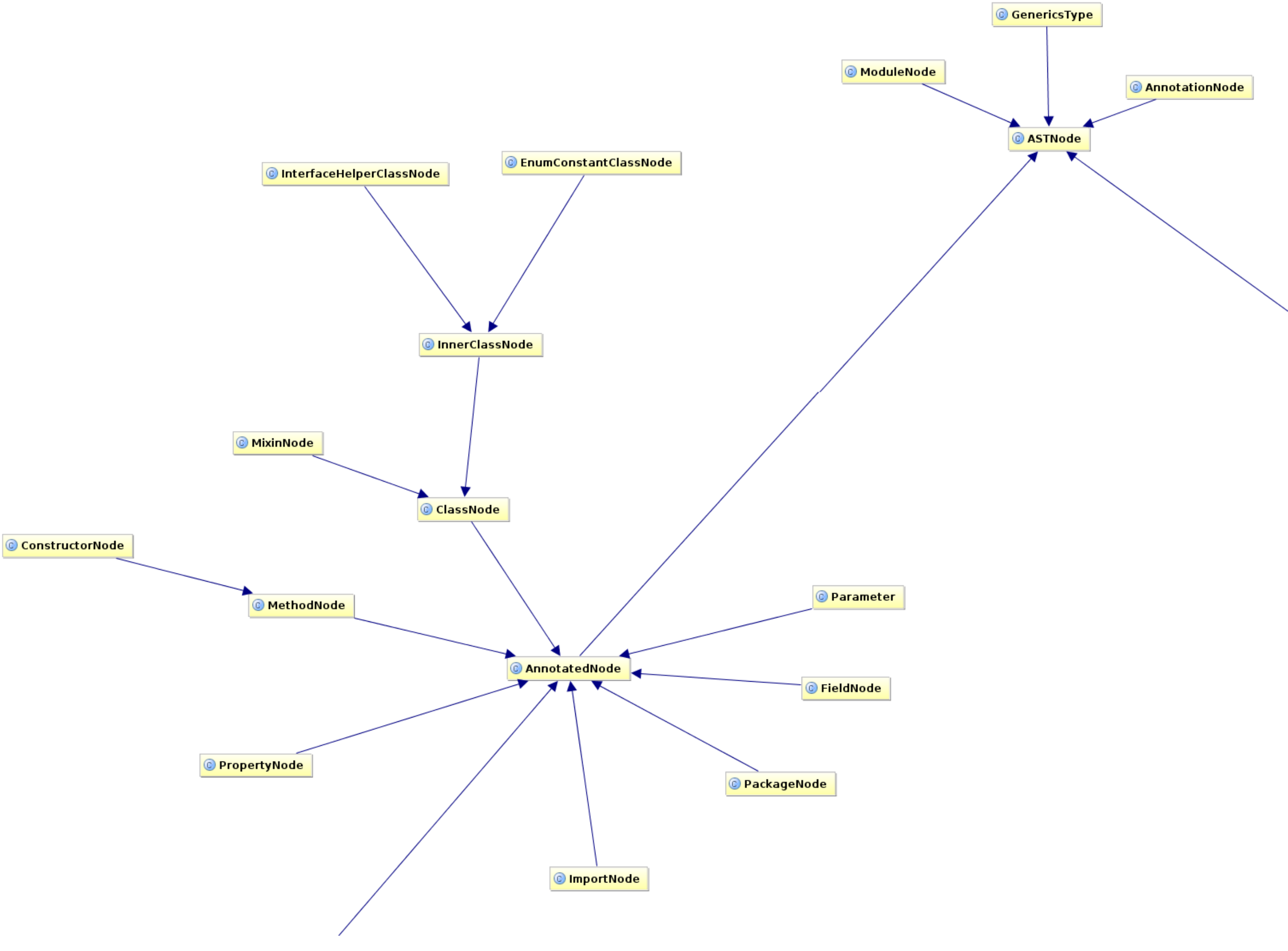
@ThreadInterrupt

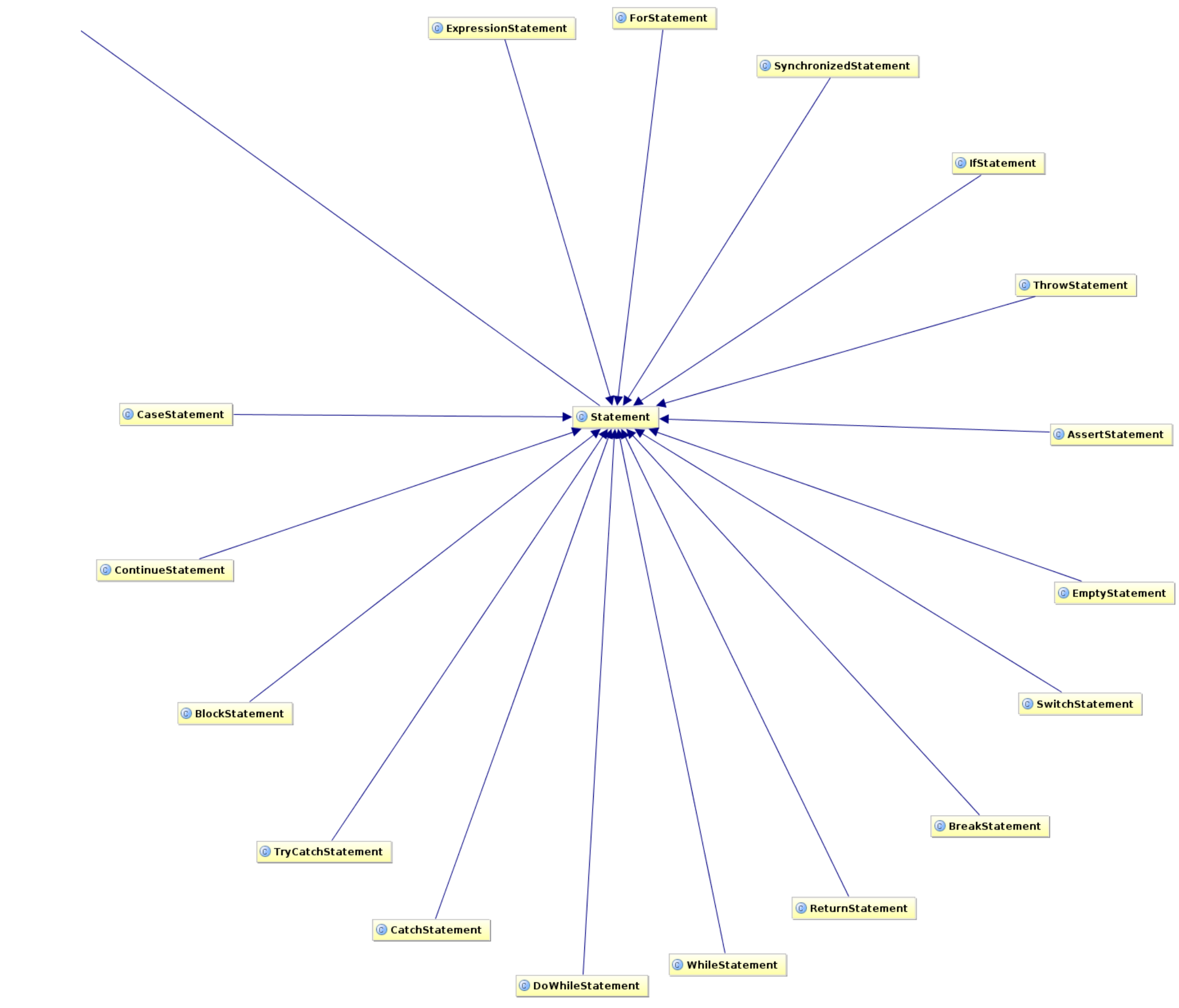
@ConditionalInterrupt

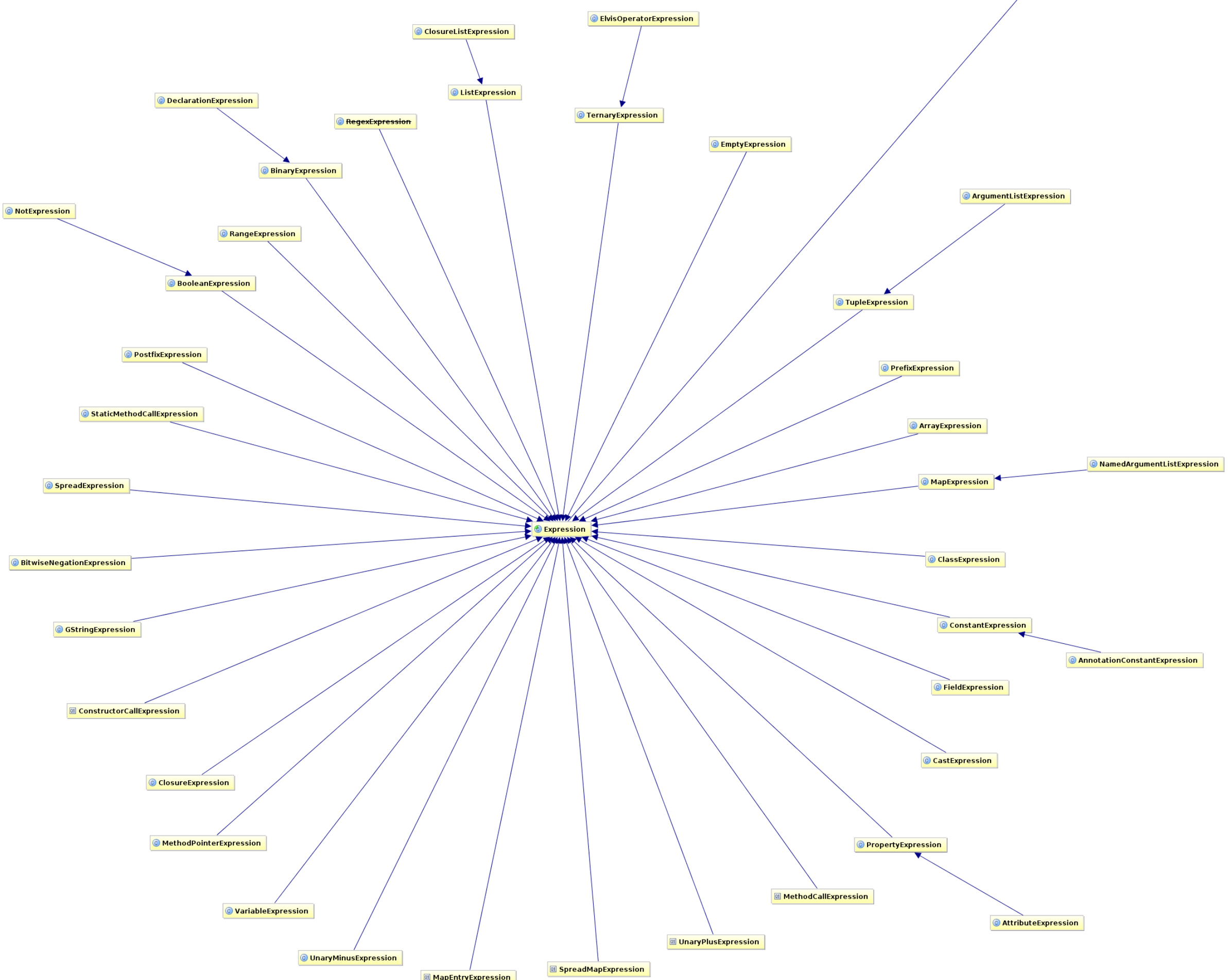
How it Works

Local AST Transformations


GenericsType
ModuleNode
AnnotationNode
ASTNode
ExpressionStatement
ForStatement
SynchronizedStatement
IfStatement
ThrowStatement
CaseStatement
Statement
AssertStatement
ContinueStatement
EmptyStatement
SwitchStatement
BlockStatement
BreakStatement
TryCatchStatement
ReturnStatement
CatchStatement
WhileStatement
DoWhileStatement
InterfaceHelperClassNode
EnumConstantClassNode
InnerClassNode
MixinNode
ClassNode
ConstructorNode
MethodNode
Parameter
AnnotatedNode
FieldNode
PropertyNode
PackageNode
ImportNode
ClosureListExpression
ElvisOperatorExpression
ListExpression
TernaryExpression
DeclarationExpression
RegexExpression
EmptyExpression
BinaryExpression
ArgumentListExpression
NotExpression
RangeExpression
TupleExpression
BooleanExpression
PostfixExpression
PrefixExpression
StaticMethodCallExpression
ArrayExpression
NamedArgumentListExpression
SpreadExpression
MapExpression
Expression
BitwiseNegationExpression
ClassExpression
ConstantExpression
GStringExpression
AnnotationConstantExpression
FieldExpression
ConstructorCallExpression
CastExpression
ClosureExpression
MethodPointerExpression
PropertyExpression
MethodCallExpression
VariableExpression
AttributeExpression
UnaryPlusExpression
UnaryMinusExpression
MapEntryExpression
SpreadMapExpression
Powered by yFiles


GenericsType
ModuleNode
AnnotationNode
ASTNode
InterfaceHelperClassNode
EnumConstantClassNode
InnerClassNode
MixinNode
ClassNode
ConstructorNode
MethodNode
Parameter
AnnotatedNode
FieldNode
PropertyNode
PackageNode
ImportNode


ExpressionStatement
ForStatement
SynchronizedStatement
IfStatement
ThrowStatement
CaseStatement
Statement
AssertStatement
ContinueStatement
EmptyStatement
BlockStatement
SwitchStatement
TryCatchStatement
BreakStatement
CatchStatement
ReturnStatement
DoWhileStatement
WhileStatement


ClosureListExpression
ElvisOperatorExpression
ListExpression
DeclarationExpression
RegexExpression
TernaryExpression
EmptyExpression
BinaryExpression
ArgumentListExpression
NotExpression
RangeExpression
BooleanExpression
TupleExpression
PostfixExpression
PrefixExpression
StaticMethodCallExpression
ArrayExpression
NamedArgumentListExpression
SpreadExpression
MapExpression
Expression
BitwiseNegationExpression
ClassExpression
ConstantExpression
GStringExpression
AnnotationConstantExpression
FieldExpression
ConstructorCallExpression
CastExpression
ClosureExpression
MethodPointerExpression
PropertyExpression
MethodCallExpression
VariableExpression
AttributeExpression
UnaryPlusExpression
UnaryMinusExpression
SpreadMapExpression
MapEntryExpression
Powered by yFiles

```groovy
class Event {
    @Delegate Date when
}
```

```groovy
@GroovyASTTransformationClass("org.pkg.DelegateTransform")
public @interface Delegate {
    ...
}
```

```groovy
@GroovyASTTransformation(phase = CANONICALIZATION)
public class DelegateTransform implements ASTTransformation {
    public void visit(ASTNode[] nodes, SourceUnit source) {
        ...
    }
}
```

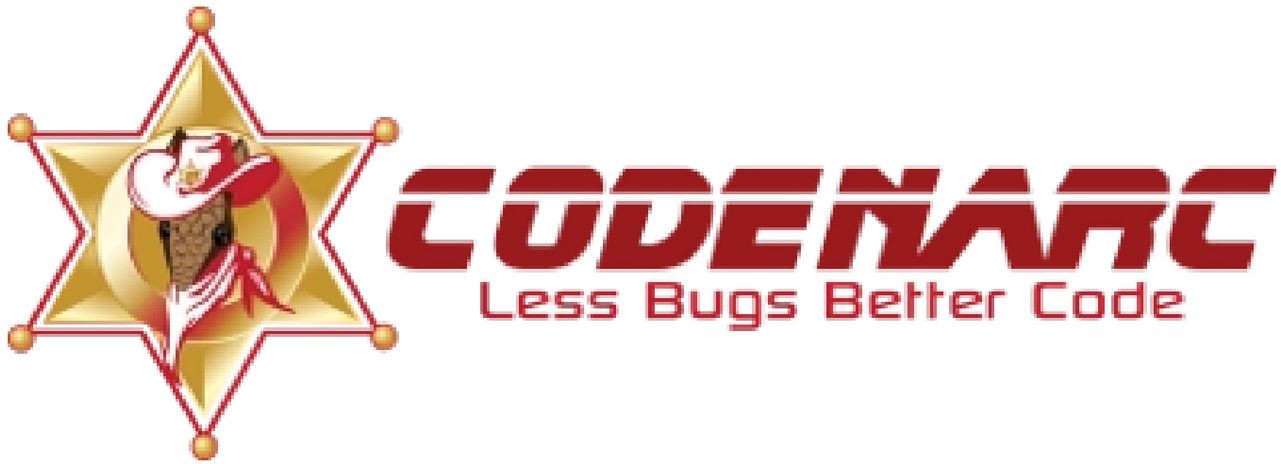
Generating a main(...) Method

```
class MainExample {

    @Main
    public void greet() {
        println "Hello from greet()!"
    }
}
```

```
$ groovy MainExample.groovy
Hello from greet()!
```

```
MethodNode makeMainMethod(MethodNode source) {
    def className = source.declaringClass.name
    def methodName = source.name

    def ast = new AstBuilder().buildFromString(
                        INSTRUCTION_SELECTION, false, """
        package $source.declaringClass.packageName

        class $source.declaringClass.nameWithoutPackage {
            public static void main(String[] args) {
                new $className().$methodName()
            }
        }
    """)
    ast[1].methods.find { it.name == 'main' }
}
```

Static Analysis with CodeNarc



CODENARC
Less Bugs Better Code

– Dead Code

– Defects like Gstring as Map Key, Duplicate Map Key

– Return path analysis (null returns)

– Type inference (and improving)

– Concurrency Problems (busy wait, etc)

– Poor Testing Practices

– Un-Groovy code

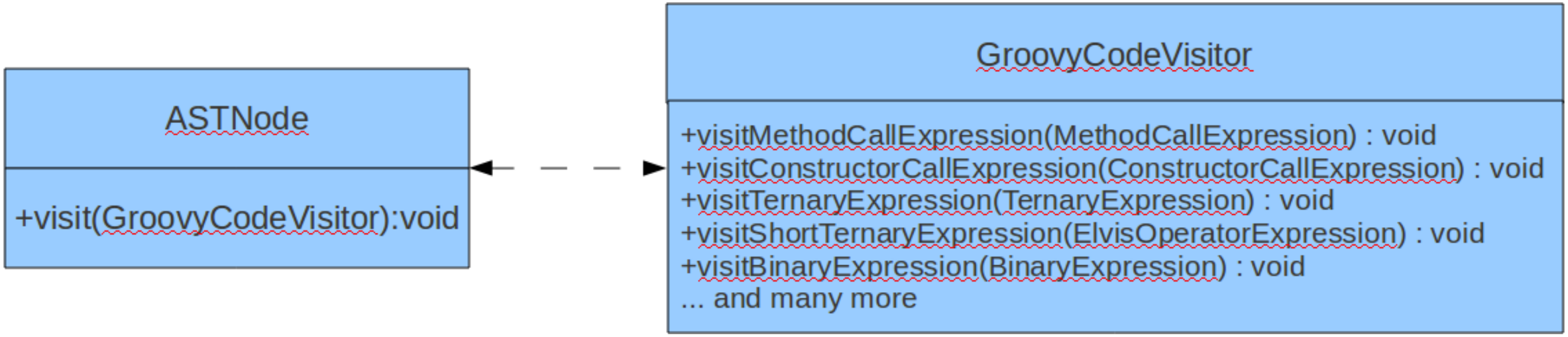
… and 260+ more rules

How it works

CodeNarc Rule:
    Ban new java.util.Random() calls

```
@Override
void visitConstructorCallExpression(
                        ConstructorCallExpression call) {

    if (AstUtil.classNodeImplementsType(call.type, Random)) {
        addViolation(call, 'Using Random is insecure...')
    }
    super.visitConstructorCallExpression(call)
}
```

How it works


ASTNode
+visit(GroovyCodeVisitor):void
GroovyCodeVisitor
+visitMethodCallExpression(MethodCallExpression) : void
+visitConstructorCallExpression(ConstructorCallExpression) : void
+visitTernaryExpression(TernaryExpression) : void
+visitShortTernaryExpression(ElvisOperatorExpression) : void
+visitBinaryExpression(BinaryExpression) : void
... and many more

Embedded Languages

```
def s = new ArithmeticShell()
assert 2 == s.evaluate(' 1+1 ')
assert 1.0 == s.evaluate('cos(2*PI)')
```

```
public interface GroovyCodeVisitor {
    void visitBlockStatement(BlockStatement statement);
    void visitForLoop(ForStatement forLoop);
    void visitWhileLoop(WhileStatement loop);
    void visitDoWhileLoop(DoWhileStatement loop);
    ...
}
```

Tree Pattern Matcher – PMD

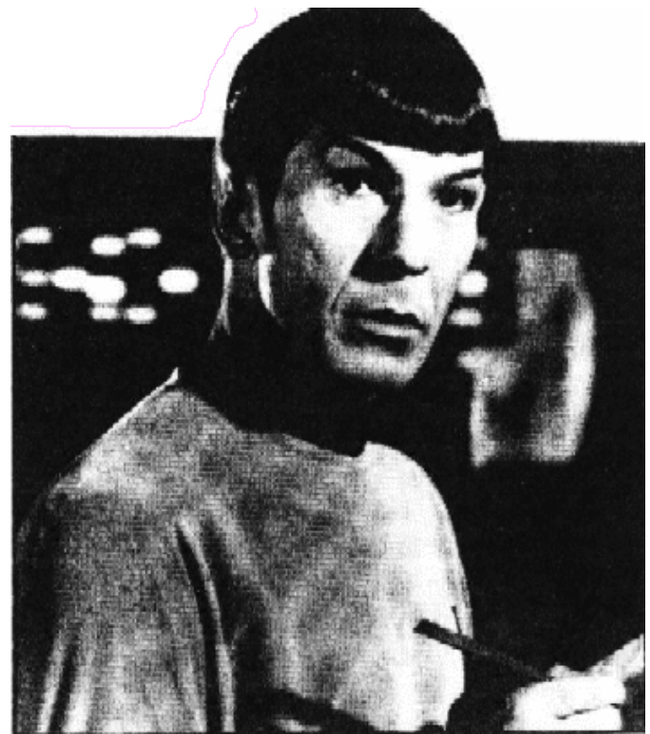
```
//FinallyStatement//ReturnStatement


//SynchronizedStatement/Block[1][count(*) = 0]


//AllocationExpression/ClassOrInterfaceType
    [contains(@Image,'ThreadGroup')] |
//PrimarySuffix
    [contains(@Image, 'getThreadGroup')]
```

How it Works

Compiles Groovy to AST
Analyzes AST Nodes
– Not well typed
– Not many tokens
– unreliable with other AST Transforms

Java Perversions

```
def "Does simple math work?"() {
    expect:
    def s = new ArithmeticShell()
    s.evaluate(input) == output

    where:
    input           | output
    '1 + 1'         | 2
    'cos(2*PI)'     | 1.0
}
```



2.0

```
void method(String message) {
   if (message != null) {
       log.info("Received input: ${message.toUppercase()}")
   }
}
```

```groovy
@groovy.transform.TypeChecked
void method(String message) {
   if (message != null) {
       log.info("Received input: ${message.toUppercase()}")
   }
}
```

```groovy
@groovy.transform.TypeChecked
void method(String message) {
   if (message != null) {
       log.info("Received input: ${message.toUppercase()}")
   }
}
```

```
1 compilation error:

[Static type checking] - Cannot find matching
method java.lang.String#toUppercase()
 at line: 4, column: 43
```

```
void method(Object message) {
   if (message instanceof String) {
       log.info("Received input: " + message.toUpperCase());
   }
}
```

```groovy
@groovy.transform.TypeChecked
void method(Object message) {
   if (message instanceof String) {
       log.info("Received input: ${message.toUpperCase()}")
   }
}
```

```groovy
@Log
class MyClass {

  def message

  @groovy.transform.TypeChecked
  void method() {
    if (message instanceof String) {
      doSomething()
      log.info("Received input: ${message.toUpperCase()}")
    }
  }

  def doSomething() {
    // ...
  }
}
```

```groovy
@Log
class MyClass {

  def message

  @groovy.transform.TypeChecked
  void method() {
    if (message instanceof String) {
      doSomething()
      log.info("Received input: ${message.toUpperCase()}")
    }
  }

  def doSomething() {
    message = 5
  }
}
```

```
def map = [x:1,y:2,z:3]
def keys = map*.key
def values = map*.value

keys*.toUpperCase()
```

```
def map = [x:1,y:2,z:3]
def keys = map*.key
def values = map*.value

keys*.toUpperCase()
values*.toUpperCase()
```

```
Dimension d1 = [100]

Dimension d2 = ['100','200']

Dimension d3 = new Dimension(
                    width: 100,
                    height: 100,
                    depth: 100)
```

@groovy.transform.TypeChecked

- Local AST ransformation
  - Not a static compiler
  - No new syntax
  - No new semantics
  - Bytecode not changed
  - Targeted at Java developers

```
def v = 1
v = v.toString()
println v.toUpperCase()
```

Flow Sensitive Typing

```
def v = 1
v = v.toString()
println v.toUpperCase()
```

- Under Discussion

```
int fib(int i) {
    i < 2 ? 1 : fib(i - 2) + fib(i - 1)
}
```

```
@groovy.transform.CompileStatic
int fib(int i) {
    i < 2 ? 1 : fib(i - 2) + fib(i - 1)
}
```

@groovy.transform.CompileStatic

- Is a static compiler
- No new syntax
- Requires new semantics
- Bytecode is changed

Mirah

Ruby FizzBuzz

```
1.upto(100) do |n|
    print "Fizz" if a = ((n % 3) == 0)
    print "Buzz" if b = ((n % 5) == 0)
    print n unless (a || b)
    print "\n"
end
```

Mirah FizzBuzz

```
1.upto(100) do |n|
    print "Fizz" if a = ((n % 3) == 0)
    print "Buzz" if b = ((n % 5) == 0)
    print n unless (a || b)
    print "\n"
end
```

Mirah: Pure JVM Class Output

```
public class Fizz-buzz {
  public static void main(String[] argv) {
    ...

    do { n = __xform_tmp_4;
      ...
      if (n % 15 == 0)
        System.out.println("FizzBuzz");
      else if (n % 5 == 0)
        System.out.println("Buzz");
      else if (n % 3 == 0)
        System.out.println("Fizz");
      else
        System.out.println(n);
      ...
    } while (__xform_tmp_4 <= __xform_tmp_5);
  }
}
```

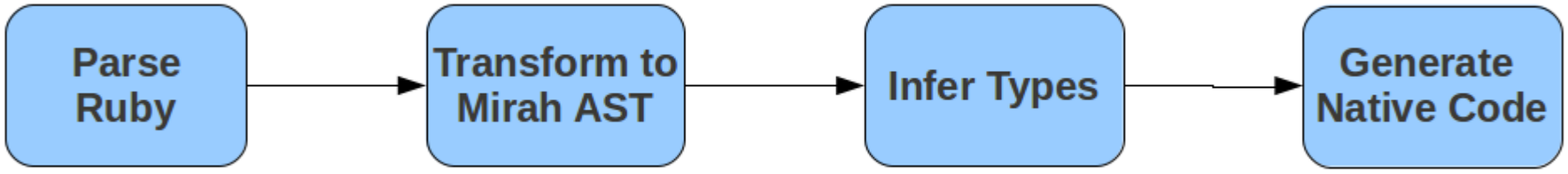
Mirah: .java File Output

```
if (((n % 15) == 0)) {
  PrintStream temp$10 = System.out;
  temp$10.println("FizzBuzz");
} else {
  if (((n % 5) == 0)) {
    PrintStream temp$11 = System.out;
    temp$11.println("Buzz");
  } else {
    if (((n % 3) == 0)) {
      PrintStream temp$12 = System.out;
      temp$12.println("Fizz");
    } else {
      PrintStream temp$13 = System.out;
      temp$13.println(n);
    }
  }
}
```

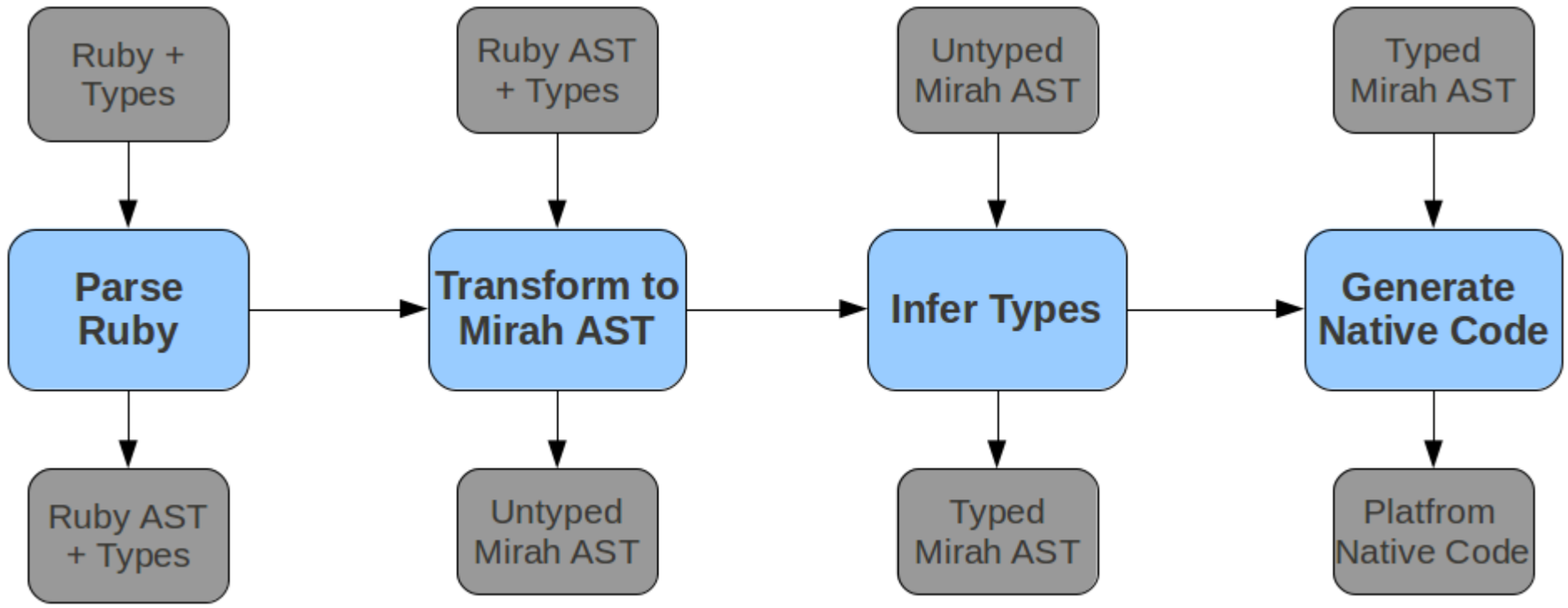
What it Means

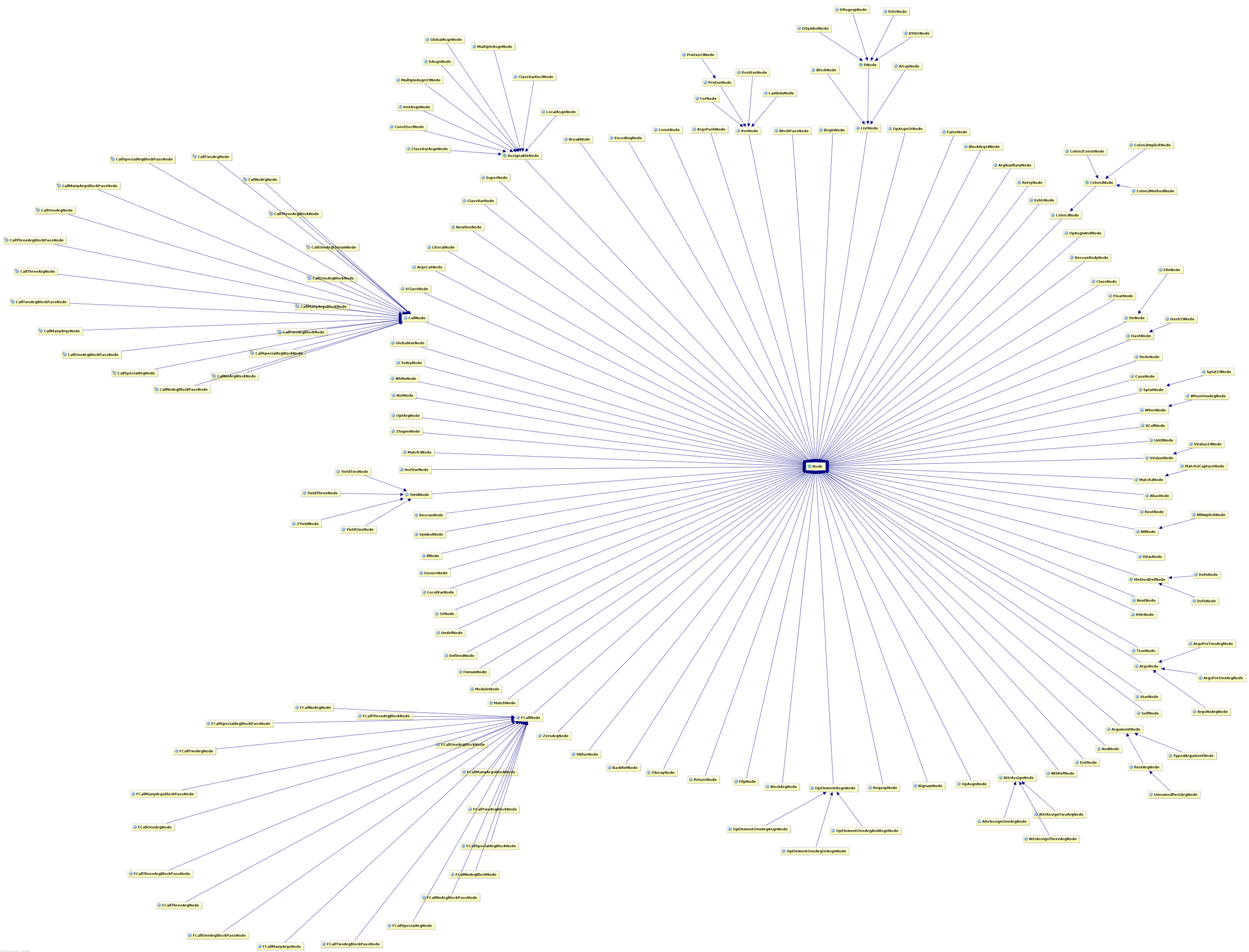
- Fast
- Lightweight
- Uses JDK
- Android?
- GWT?

How it Works


Parse Ruby
Transform to Mirah AST
Infer Types
Generate Native Code

How it Works


Ruby + Types
Ruby AST + Types
Untyped Mirah AST
Typed Mirah AST
Parse Ruby
Transform to Mirah AST
Infer Types
Generate Native Code
Ruby AST + Types
Untyped Mirah AST
Typed Mirah AST
Platfrom Native Code



Mirah Macros

```
macro def eachChar(value, &block)
    quote {
        `value`.toCharArray.each do | my_char |
            `block.body`
        end
    }
end

eachChar('laat de leeeuw niet ...') do | my_char |
    puts my_char
end
```

Mirah Macros

```
class Person
    make_attr :firstName, :string
end
```

```
p = Person.new
p.firstName = 'hamlet'
puts p.firstName
```

Mirah Macros

```
macro def make_attr(name, type)
    attribute_name = name.string_value()
    quote {
        def `name`
            @`name`
        end
        def `"#{attribute_name}_set"`(value:`type`)
            @`name` = value
        end
    }
end
```

Summary

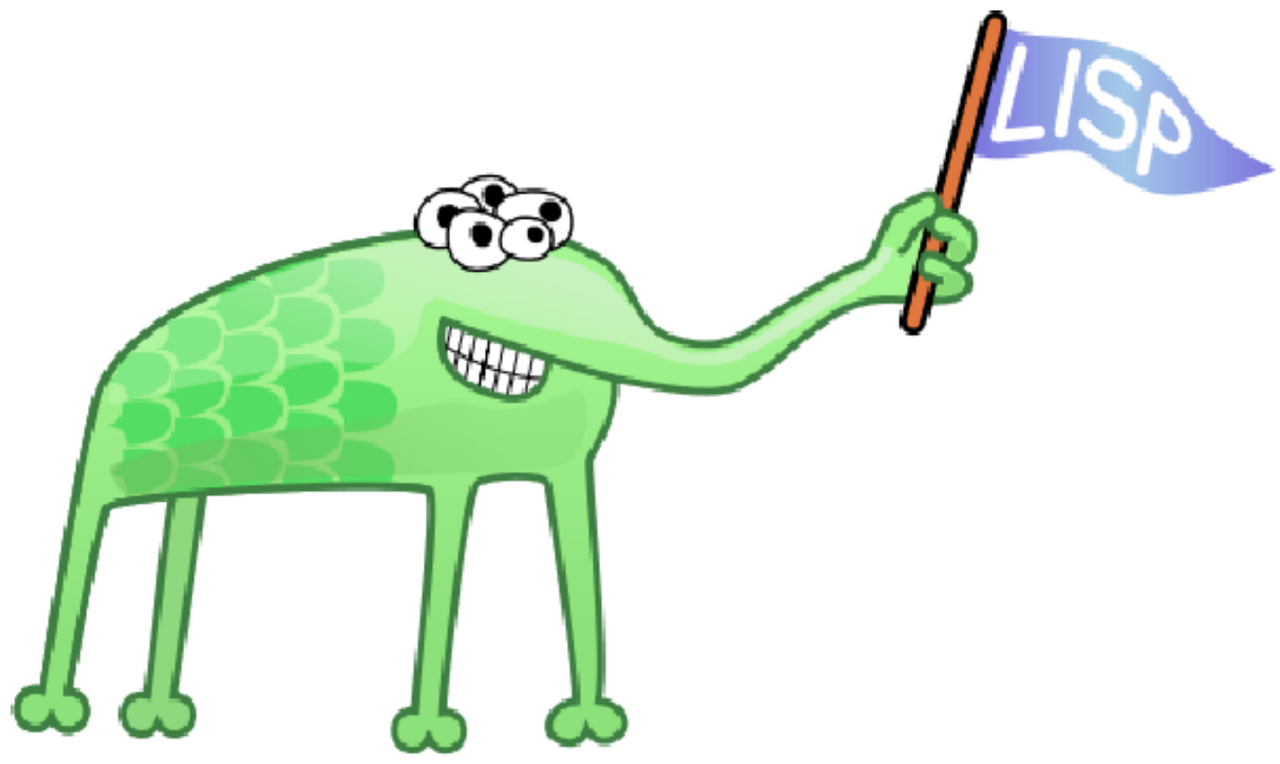
AST Transformations are Framework oriented

- *Easily transform classes and fields*
- *Difficult to find call-sites*
- *Still Difficult to generate AST*

Macros are more general

- *Compiler handles invocation*
- *Easily quote ASTs into source*
- *Static types are a big win for transformations*

Not Covered Here...



Agenda

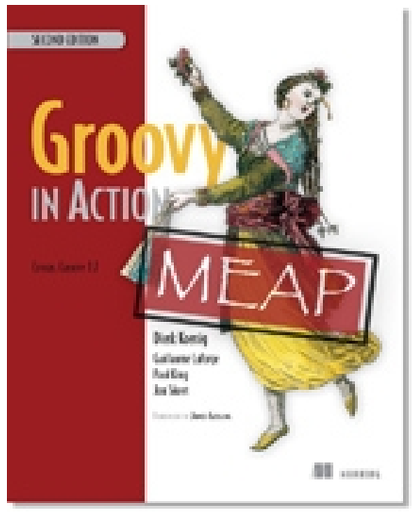
Lombok and Groovy AST Transformations
– *Abstract Syntax Trees*

CodeNarc and Groovy 2.0
– *Static Analysis and Type Checking*

Mirah
– *Macros and Static Typing*

Thanks!

What to do next:
– Groovy Wiki and Mailing List is amazingly helpful
– Use your creativity and patience
– http://hamletdarcy.blogspot.com & @HamletDRC



Groovy, Grails, Griffon, and Agile Consulting

info@canoo.com or hamlet.darcy@canoo.com