Static meta-programming



Václav Pech

NPRG014 2017/2018

http://jroller.com/vaclav

http://www.vaclavpech.eu

@vaclav_pech



Last time agenda

Dynamic meta-programming



Domain-specific languages

Builders

Agenda for today

AST transformations

Static meta-programming

Part 5

Static meta-programming

AST

```
At end of Phase: Canonicalization
   ClassNode - Calculator
  MethodNode - divide10By

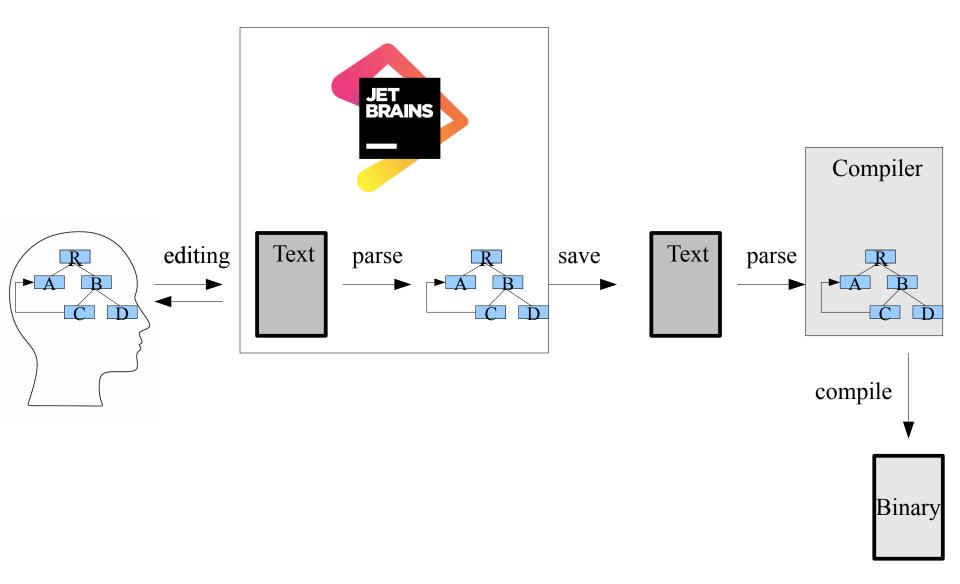
    Parameter - a

      -- 🏿 Parameter - b
     BlockStatement - (1)
         EmptyStatement

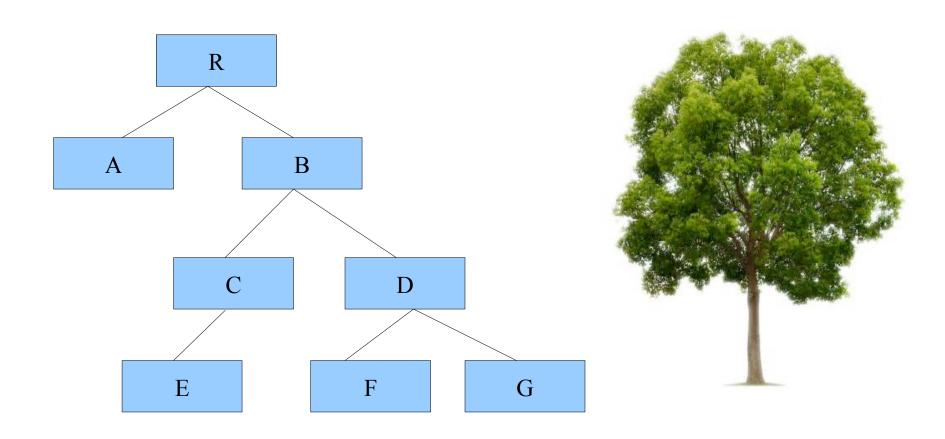
    □ ■ ReturnStatement - return (a - b)

         🖮 <table-cell-columns> Binary - (a - b)
           🖃 ル Variable - a : java.lang.Object
             🗓 🏿 Parameter - a
           □ Wariable - b: java.lang.Object
             🔙 🏿 Parameter - b
       MethodNode - this$dist$invoke$1
       public int subtract(java.lang.Object a, java.lang.Object b) {
            if (!( a > b )) {
                throw new java.lang.Exception('Precondition violated: {a > b}')
```

Programming

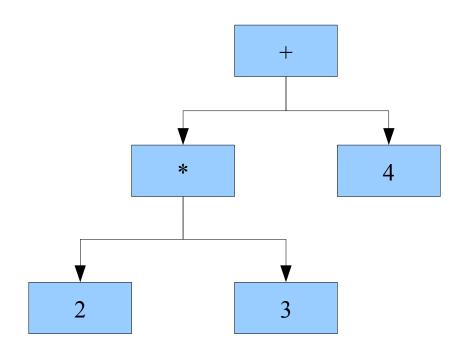


Programs are trees



Lisp (1958)

$$(+(*23)4)$$





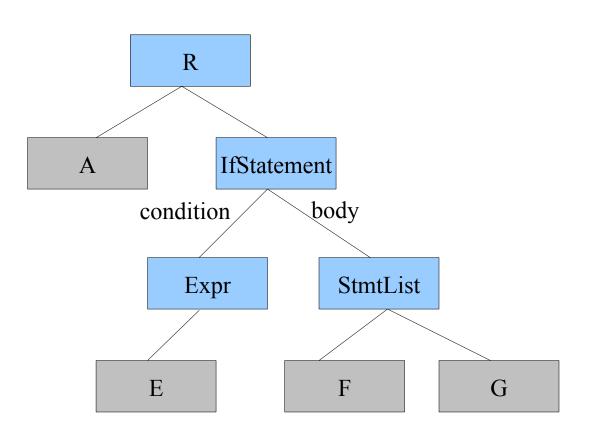
```
when Clicked
say Hello! for 2 secs
repeat until distance = 10
  move 10 steps
  play sound meow v
                                               and
                                                  then
                                                          move 10 steps
                                                           turn 🖍 15 degrees
                                                           pen down
```

```
public class Demo {
  private static void foo() {
    System.out.println("Foo called");
  public static void main(string[] args) {
    System.out.println("Application started");
      (args.length > 0) {
      System.out.println("Supplied arguments");
      for (string value : args) {
        System.out.println("Argument: " + value);
    } else {
      System.out.println("No arguments provided");
    foo();
    System.out.println("Application completed");
```

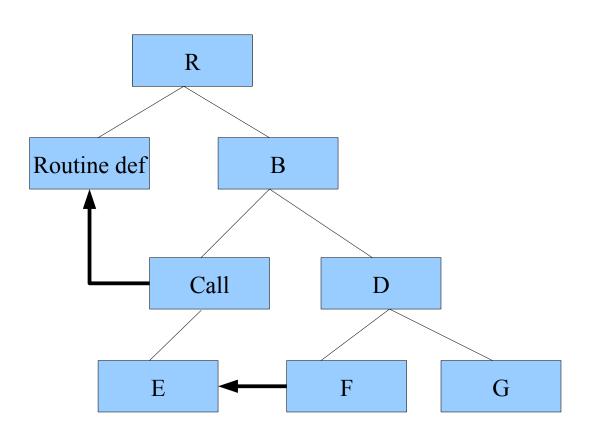
```
Concept = jetbrains.mps.baseLanguage.structure.lfStatement
ifTrue: StatementList
   Concept = jetbrains.mps.baseLanguage.structure.StatementList
   n statement : ExpressionStatement
  n statement : for
      Concept = jetbrains.mps.baseLanguage.structure.ForeachStatement
   ▼ n body : StatementList
         Concept = jetbrains.mps.baseLanguage.structure.StatementList
      statement : ExpressionStatement
            Concept = jetbrains.mps.baseLanguage.structure.ExpressionStatement
         • n expression : DotExpression {void}
               Concept = jetbrains.mps.baseLanguage.structure.DotExpression
            n operand : .<static field> {java.io.PrintStream}
            n operation : println(String):void {void}
               properties
            referents
           properties
         referents
        properties
      referents
   variable : value {string}
     iterable : args {string[]}
      properties
   referents
  properties
referents
n condition : > {boolean}
  ifFalseStatement : {
properties
referents
```

node

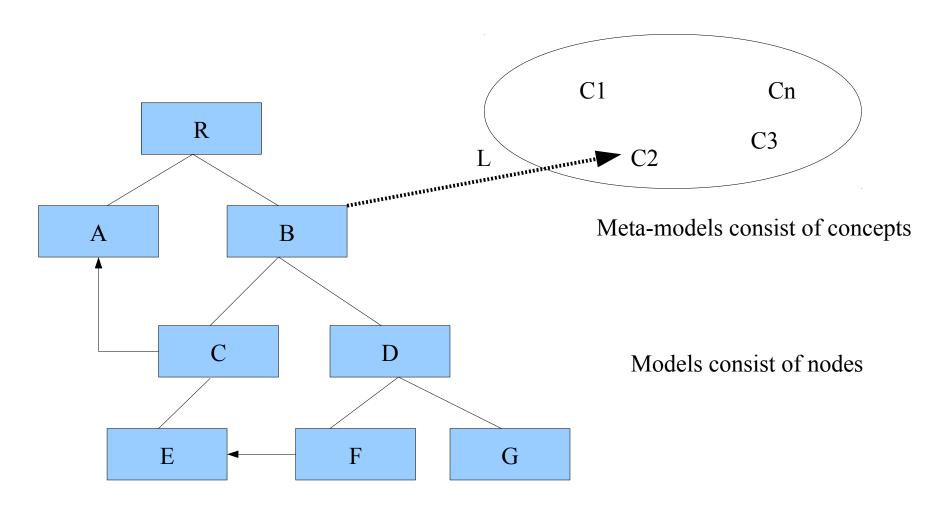
Children have roles



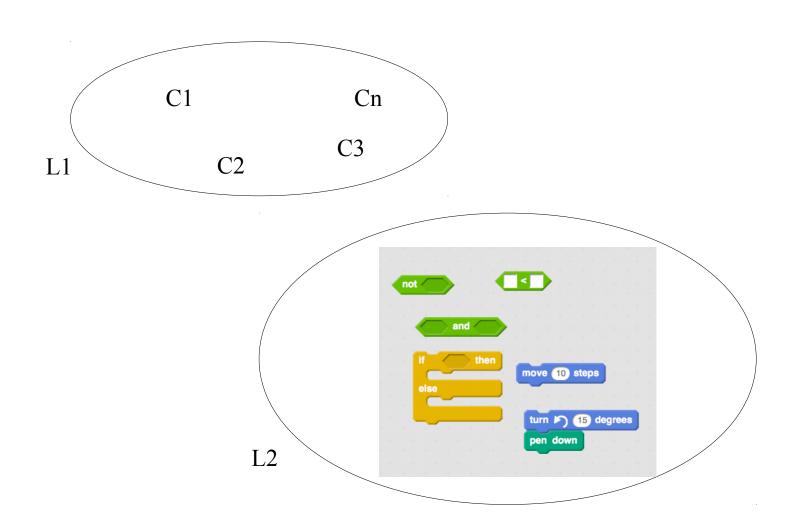
References cross the tree



Programs and Languages



Languages are sets of concepts



```
touching edge v ?
                                move 10 steps
                    then
                                turn (15) degrees
           else
                                point in direction 90
                                                         us.groovy.ast.stmt;
                                move 10 steps
                                        * Represents an if (condition) { ... } else { ... } statement in Groovy
                                       * @version $Revision$
                                       public class IfStatement extends Statement {
```

The Node

```
public class ASTNode {
    private int lineNumber = -1;
    private int columnNumber = -1;
    private int lastLineNumber = -1;
    private int lastColumnNumber = -1;
    private ListHashMap metaDataMap = null;

public void visit(GroovyCodeVisitor visitor) {
        throw new RuntimeException("No visit() method implemented for class: "
    }
}
```

```
private Expression falseExpression;

public class ElvisOperatorExpression extends TernaryExpression {
```

public ElvisOperatorExpression(Expression base, Expression falseExpression) {

public class TernaryExpression extends Expression {

private BooleanExpression booleanExpression;

super(getBool(base), base, falseExpression);

private Expression trueExpression;

```
public class ForStatement extends Statement implements LoopingStatement {
    public static final Parameter FOR_LOOP_DUMMY = new Parameter(ClassHelp
    private Parameter variable;
```

```
private rarameter variable;
private Expression collectionExpression;
private Statement loopBlock;
private VariableScope scope;
```

```
public class MethodNode extends AnnotatedNode implements Opcodes {
    public static final String SCRIPT_BODY_METHOD_KEY = "org.codeha
```

```
private final String name;
private int modifiers;
private boolean syntheticPublic;
private ClassNode returnType;
private Parameter[] parameters;
private boolean hasDefaultValue = false;
private Statement code;
private boolean dynamicReturnType;
private VariableScope variableScope;
private final ClassNode[] exceptions;
private final boolean staticConstructor;
// type spec for generics
private GenericsType[] genericsTypes = null;
private boolean hasDefault;
// cached data
```

String typeDescriptor;

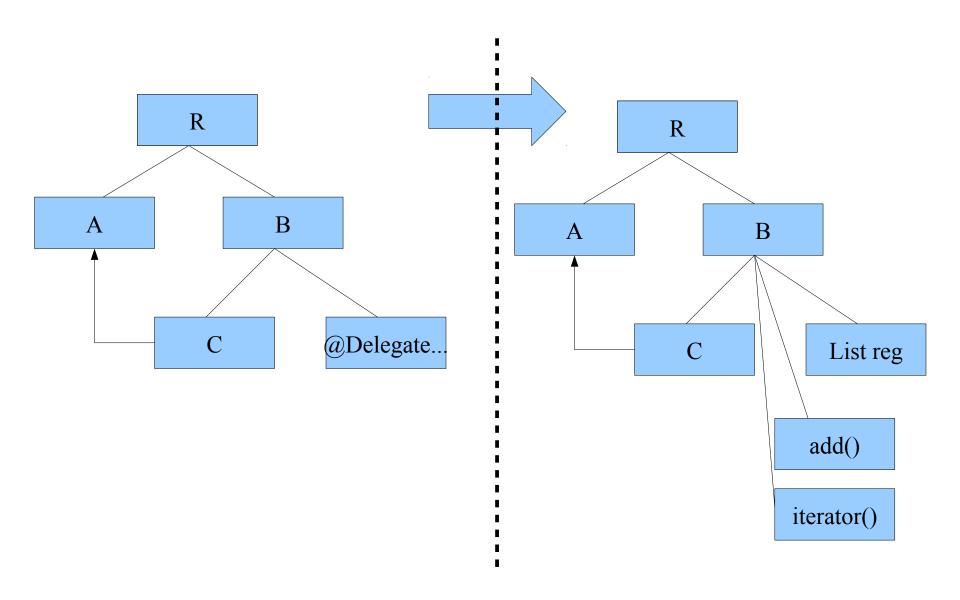
There's no life without trees

- Extending IDEs
- External DSLs
- Internal DSLs in modern languages
 - Groovy AST transformations
 - Scala macros
 - Lisp, Clojure macros
 - Lombok

AST Transformations

```
class Registrations {
  @Delegate List items = []
def people = new Registrations()
people.addAll(["Joe", "Dave"])
assert ["Dave", "Joe"] == people.reverse()
```

Ast transformation



- @Delegate, @Immutable, @Singleton
- @Lazy
- @TupleConstructor
- @InheritConstructors
- @Canonical
- @ToString
- @EqualsAndHashCode

- @Log, @Log4j, @Commons
- @Synchronized
- @WithReadLock
- @WithWriteLock
- @AutoClone, @AutoExternalize

. . .

Creating AST Transformations

new AstBuilder()

.buildFromString()

.buildFromCode()

.buildFromSpec()

```
.buildFromString ("'
Integer.parseInt("$param")
"'')
```

```
.buildFromSpec {
  method('convertToNumber', ACC_PUBLIC, Integer) {
         parameters { parameter 'parameter': String.class }
         exceptions {}
         block {
            returnStatement {
              staticMethodCall(Integer, "parseInt") {
                 argumentList {
                   variable "parameter"
```

Type-checking/Static

@TypeChecked, @CompileStatic

```
@TypeChecked
String test(Object val) {
  if (val instanceof String) {
     val.toUpperCase()
  } else if (val instanceof Number) {
     val.intValue() * 2
```

Summary



AST transforms for Java programmers





http://jroller.com/vaclav vaclav@vaclavpech.eu

References

http://www.groovy.cz

http://groovy.codehaus.org

http://grails.org

http://groovyconsole.appspot.com/

http://www.manning.com/koenig2/