Static meta-programming and a concurrency primer

Václav Pech

NPRG014 2016/2017

http://jroller.com/vaclav

http://www.vaclavpech.eu

@vaclav_pech

Last time agenda

- Dynamic meta-programming
- Domain-specific languages
- Builders
- DSL-based frameworks

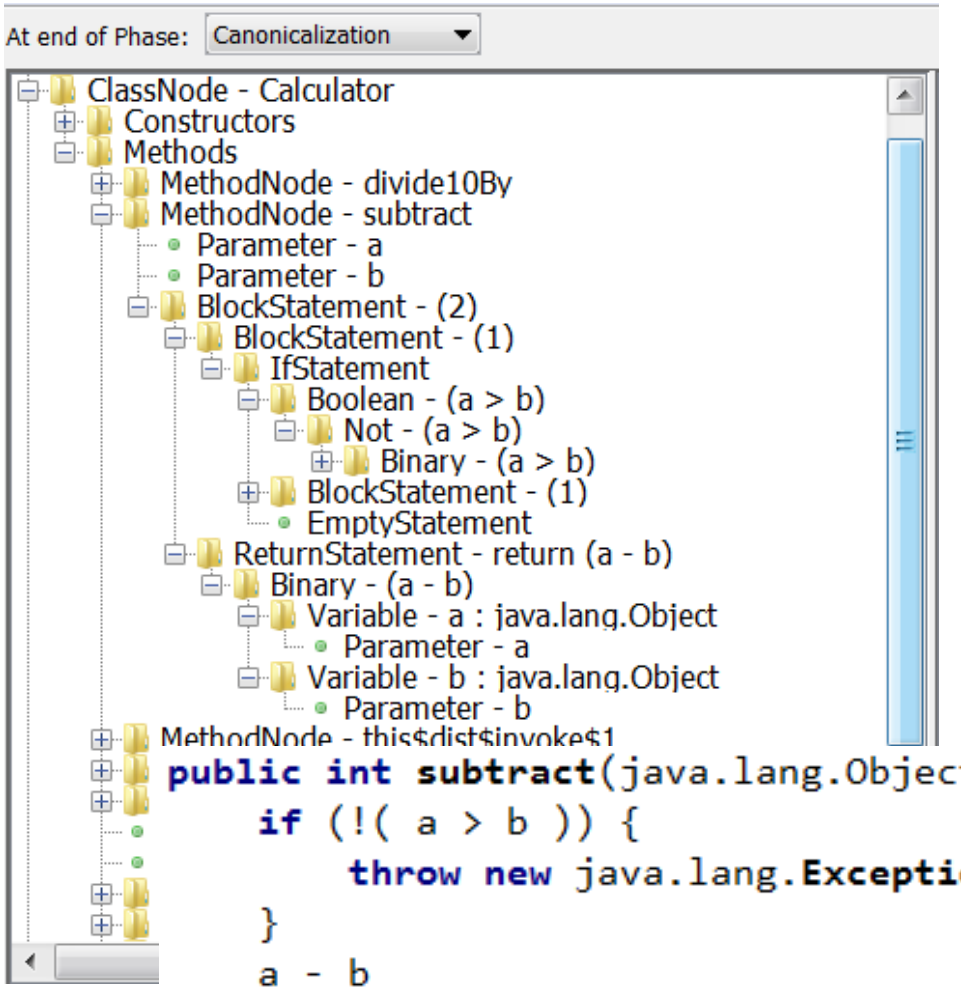
Agenda for today

- AST transformations
- Static meta-programming
- Concurrency primer

Part 5

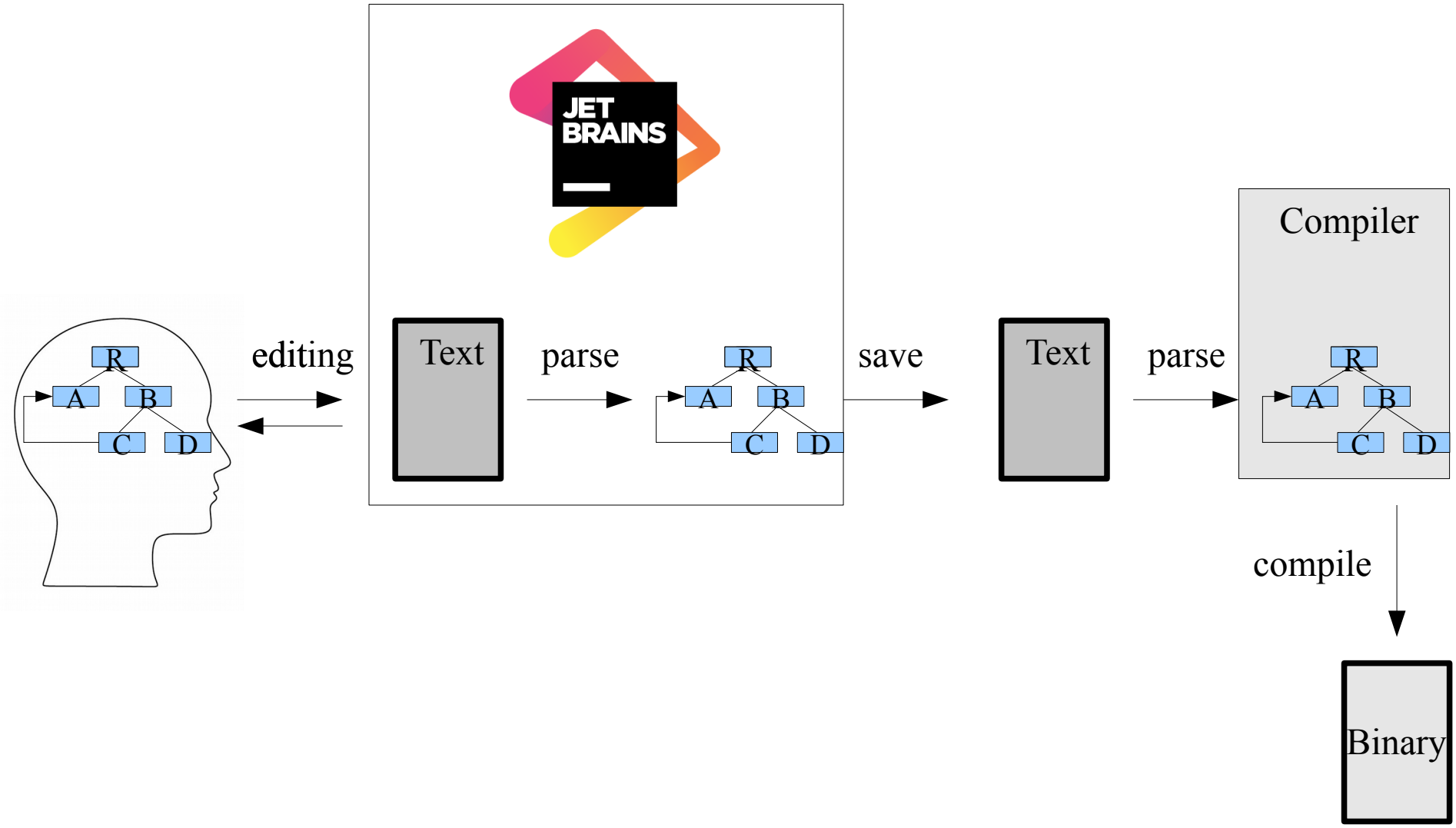
Static meta-programming

AST

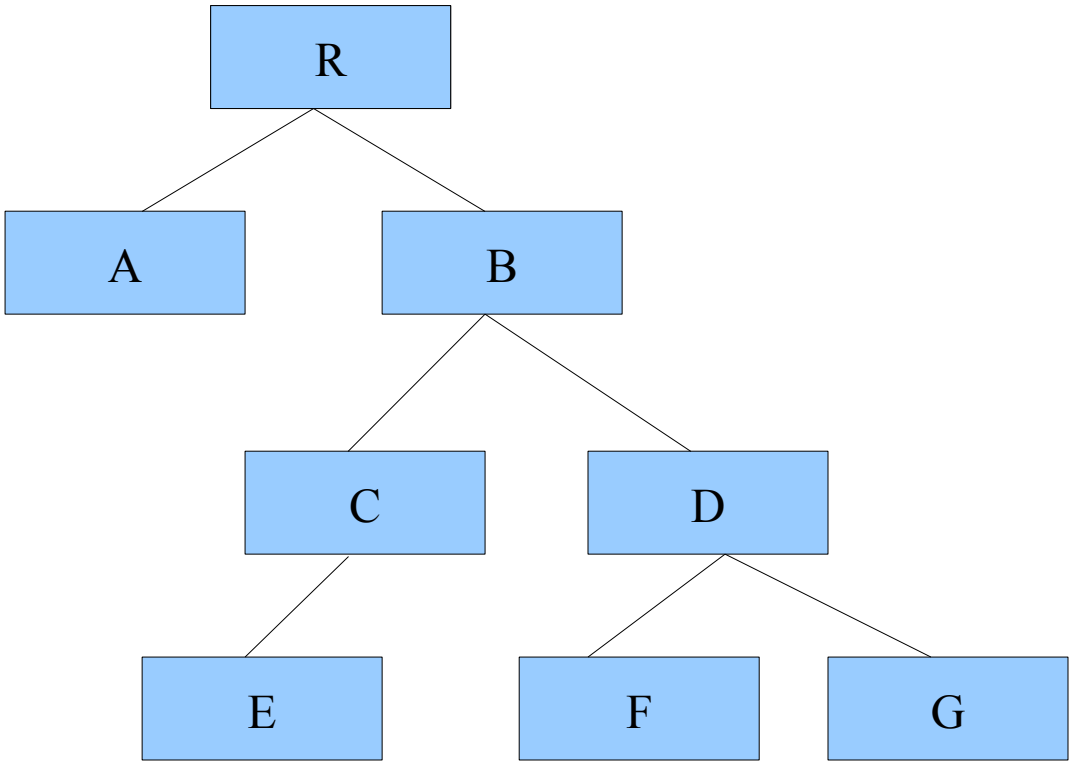



```
public int subtract(java.lang.Object a, java.lang.Object b) {
    if (!( a > b )) {
        throw new java.lang.Exception('Precondition violated: {a > b}')
    }
    a - b
}
```

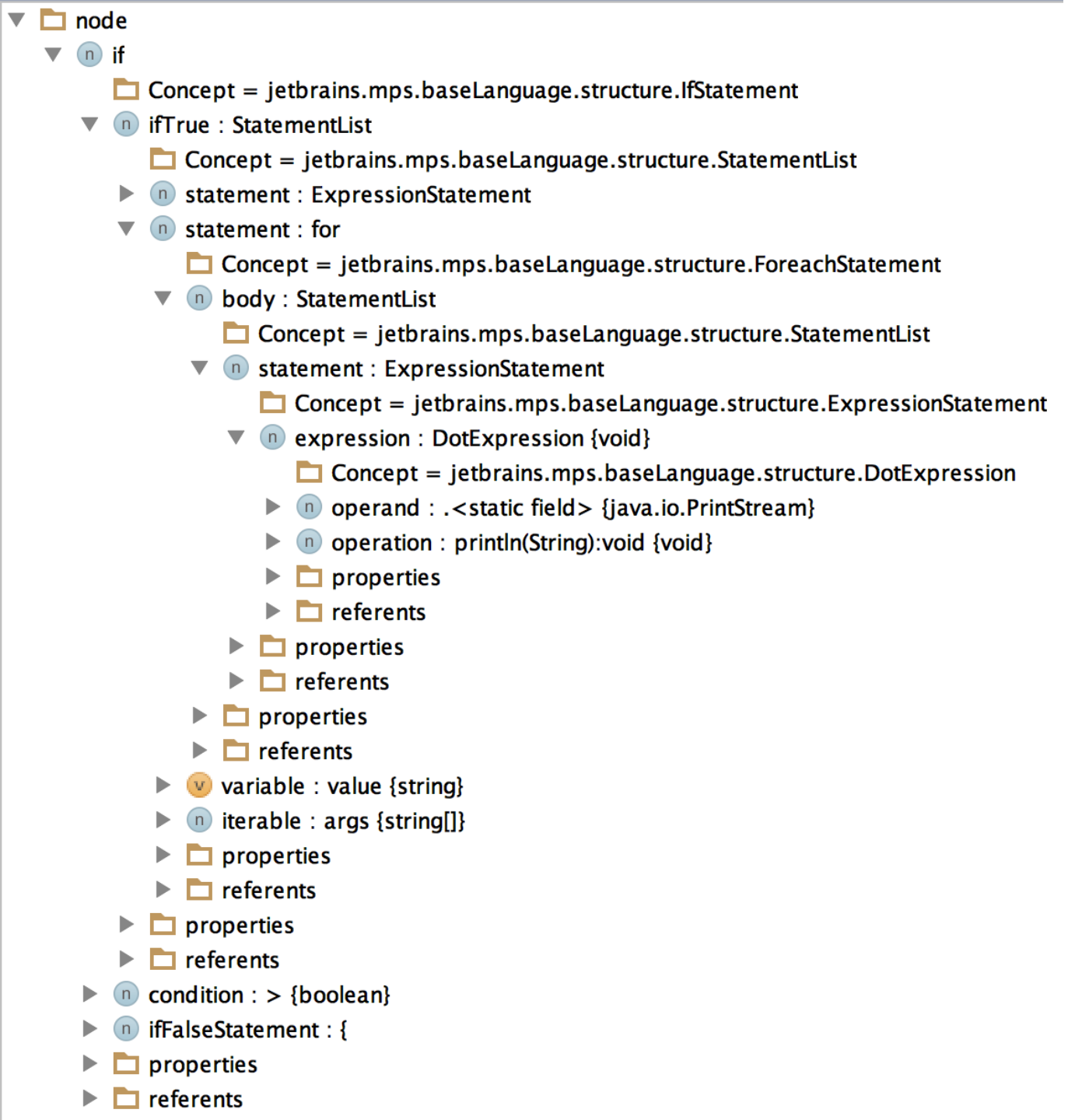
Programming



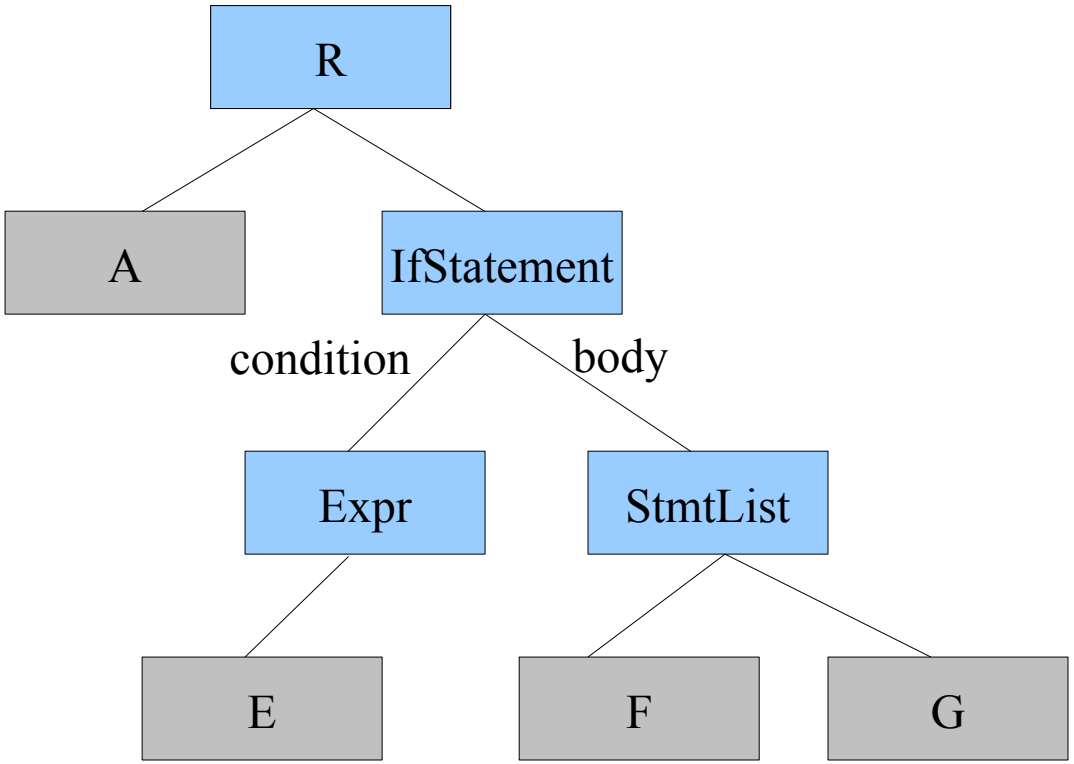
Programs are trees



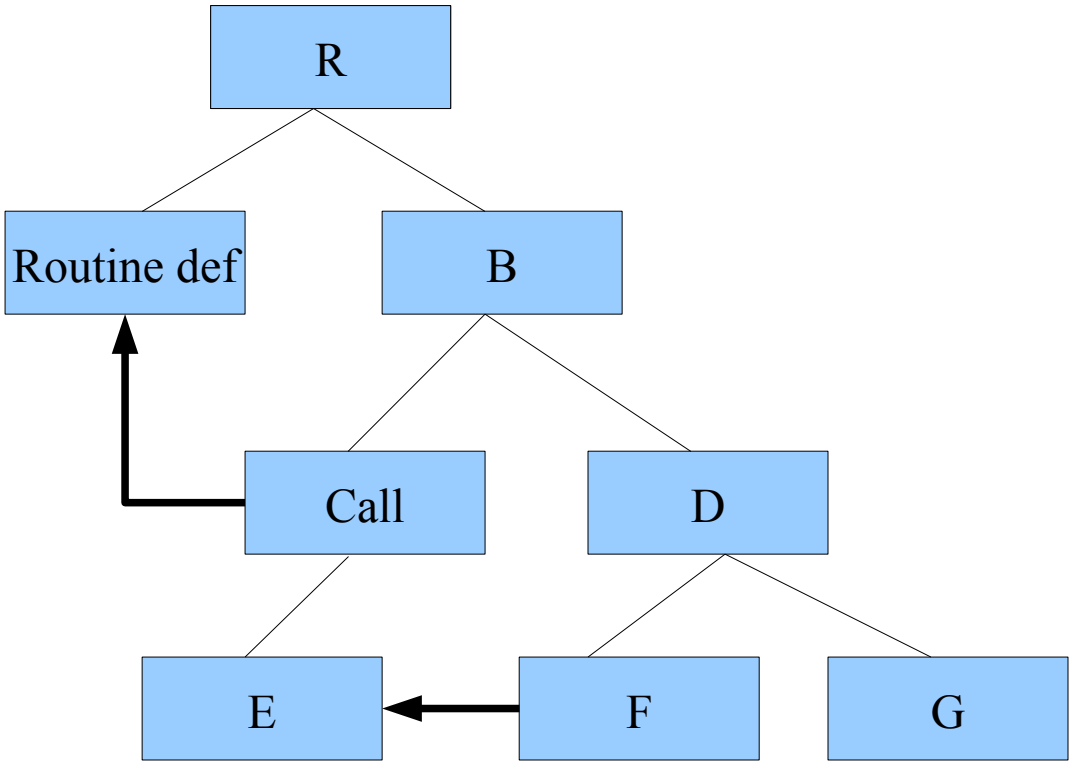
```
public class Demo {

  private static void foo() {
    System.out.println("Foo called");
  }

  public static void main(string[] args) {

    System.out.println("Application started");

    if (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");
  }

}
```


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

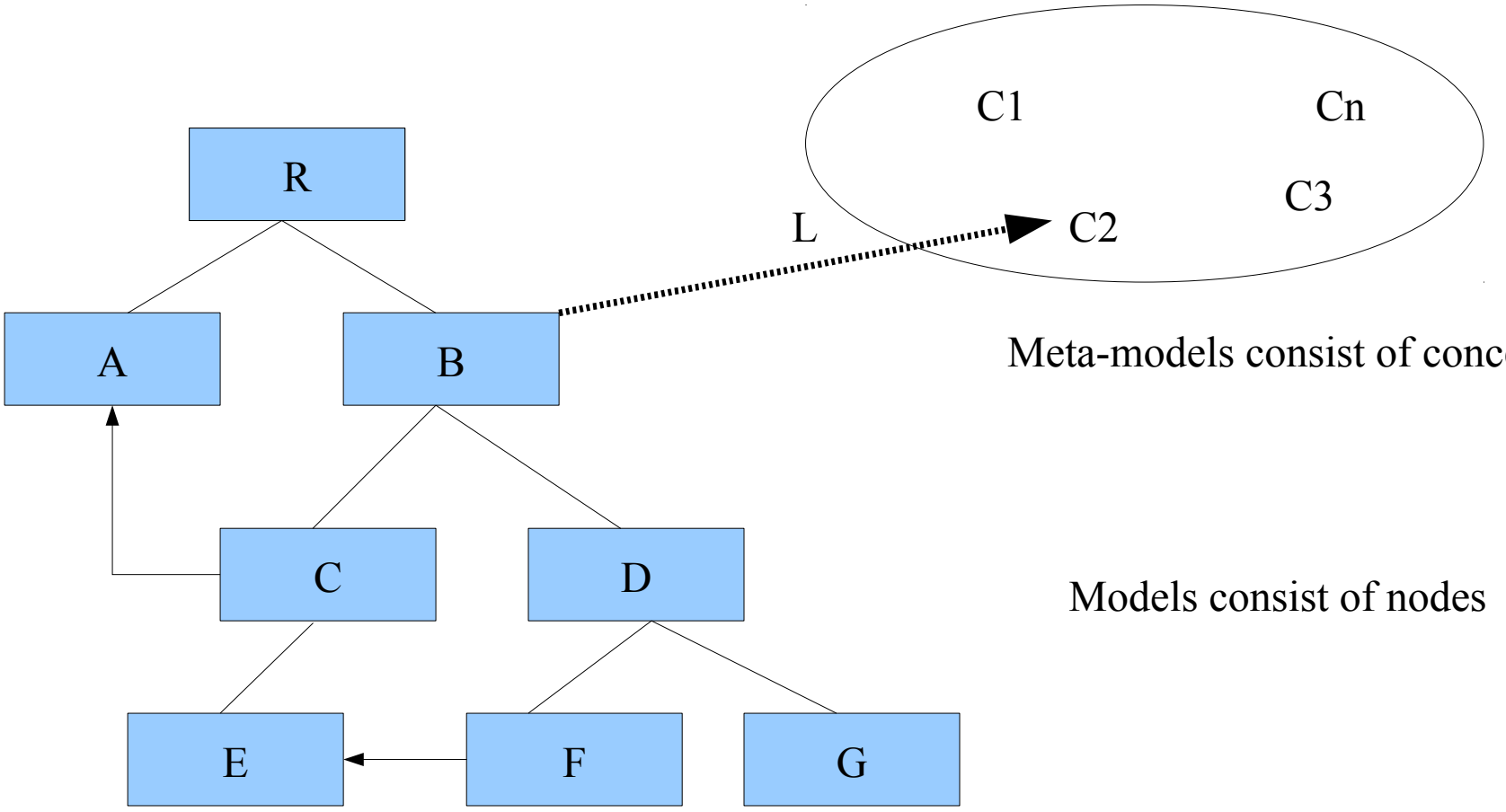
Children have roles



References cross the tree



Programs and Languages



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: "
    }
```

```
public class TernaryExpression extends Expression {

    private BooleanExpression booleanExpression;
    private Expression trueExpression;
    private Expression falseExpression;
```

```
public class ElvisOperatorExpression extends TernaryExpression {

    public ElvisOperatorExpression(Expression base, Expression falseExpression) {
        super(getBool(base), base, falseExpression);
    }
```

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

    private Parameter 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;
```

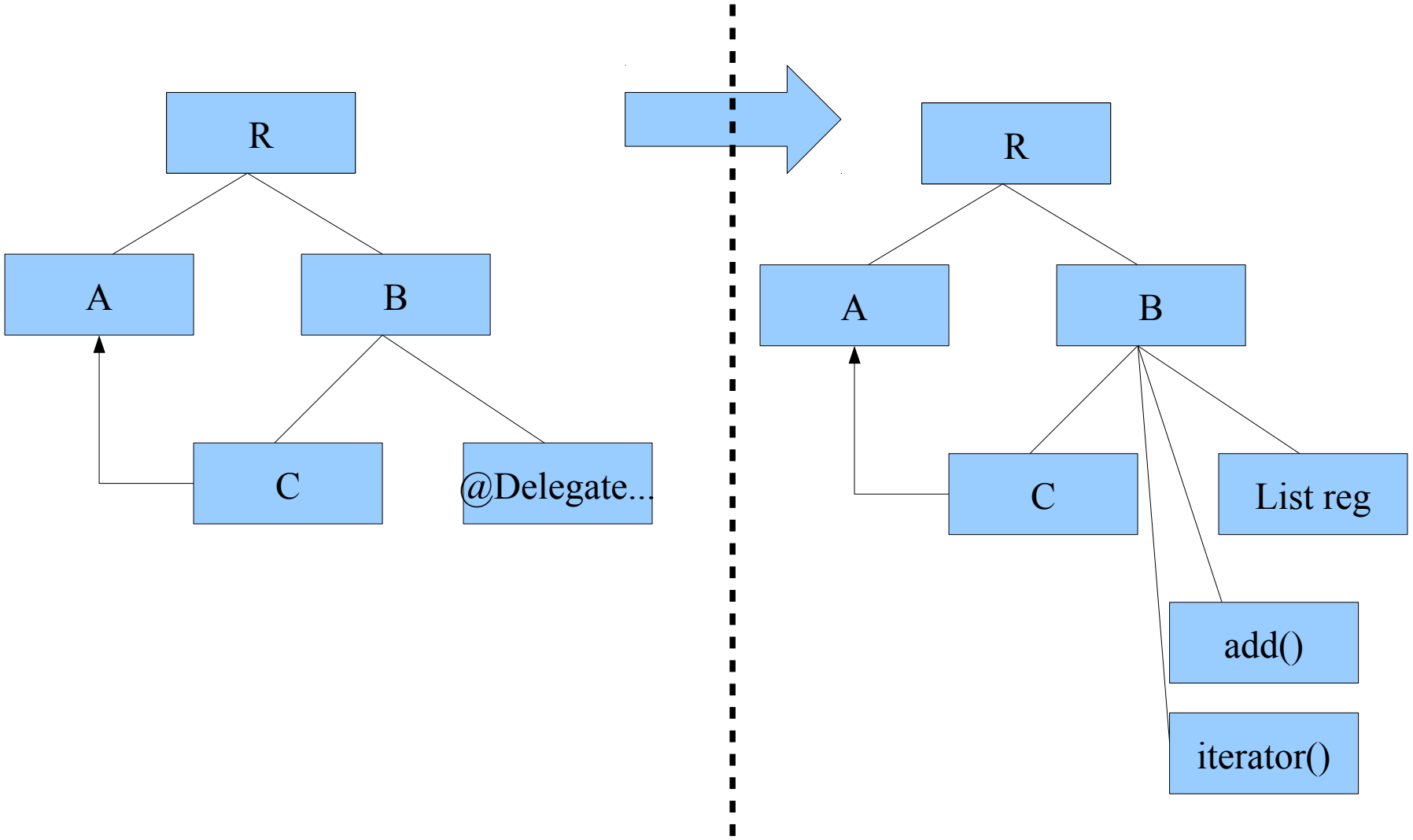
There's no life without trees

- 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")
''')
```

```
.buildFromCode (
                    Integer.parseInt("$param")
)
```

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

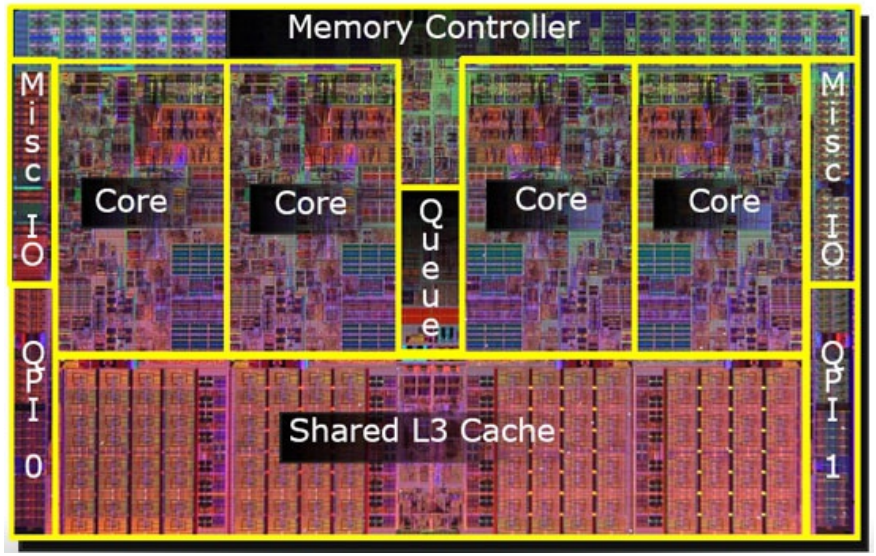
Type-checking/Static

@TypeChecked, @CompileStatic

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

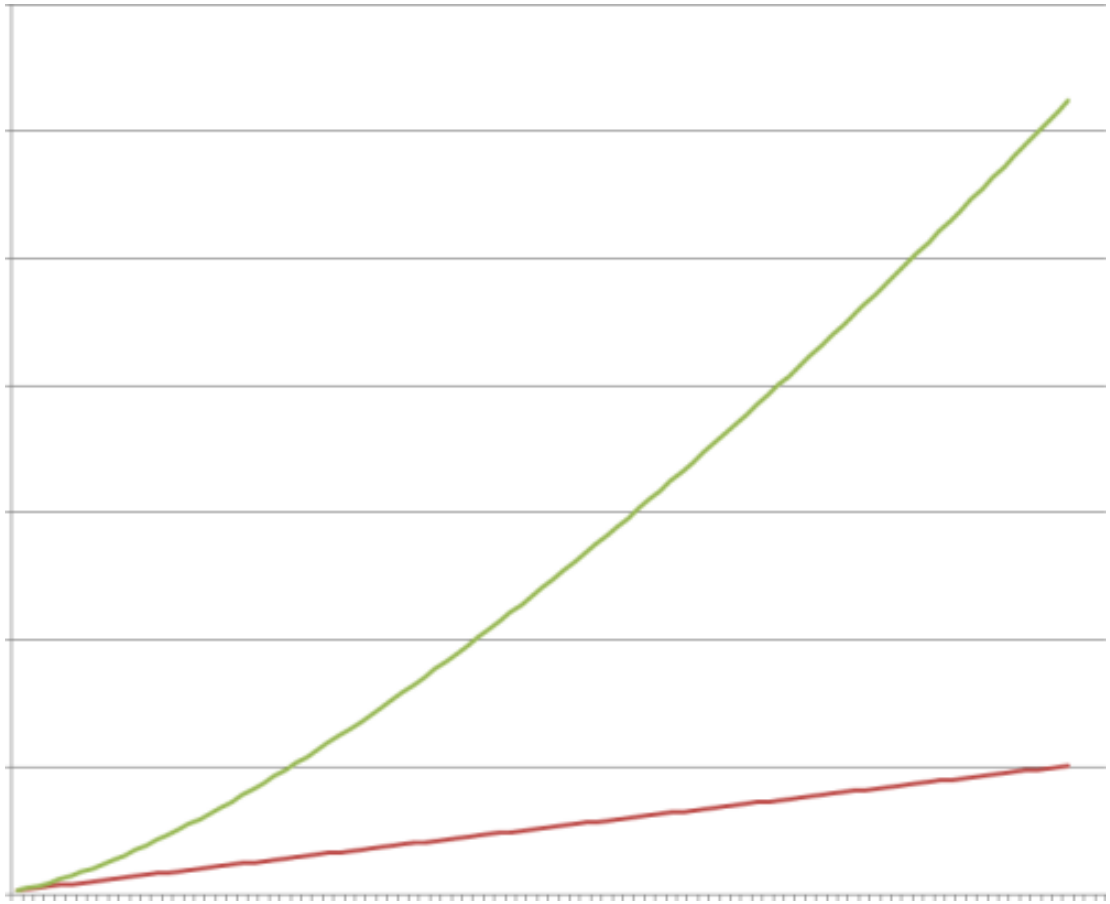
Concurrency

Why concurrency?



We’re all in the parallel computing business!

of cores



JVM machinery

Thread, Runnable, Thread Pools

JVM machinery

Thread, Runnable, Thread Pools

Synchronized blocks

Volatile

Locks

Atomic

Dealing with threads sucks!

```
public class Counter {
    private static long count = 0;

    public Counter() {

        count++;

    }
}
```

Dealing with threads sucks!

```
public class Counter {
    private volatile static long count = 0;

    public Counter() {

        count++;

    }
}
```

Dealing with threads sucks!

```
public class Counter {
    private volatile static long count = 0;

    public Counter() {

        count = count + 1;

    }
}
```

Dealing with threads sucks!

```
public class Counter {
    private static long count = 0;

    public Counter() {
        synchronized (this) {
            count++;
        }
    }
}
```

Dealing with threads sucks!

```
public class Counter {
    private static long count = 0;

    public Counter() {
        synchronized (this.getClass()) {
            count++;
        }
    }
}
```

Dealing with threads sucks!

```
public class Counter {
    private Long count = 0;

    public doSomething() {
        synchronized (count) {
            count++;
        }
    }
}
```

Dealing with threads sucks!

```
public class Counter {
    private Long count = 0;

    public doSomething() {
        synchronized (count) {
            count = new Long(count.longValue() + 1);
        }
    }
}
```

Dealing with threads sucks!

```
public class ClickCounter implements ActionListener {
    public ClickCounter(JButton button) {
        button.addActionListener(this);
    }

    public void actionPerformed(final ActionEvent e) {
        ...
    }
}
```

Stone age of parallel SW

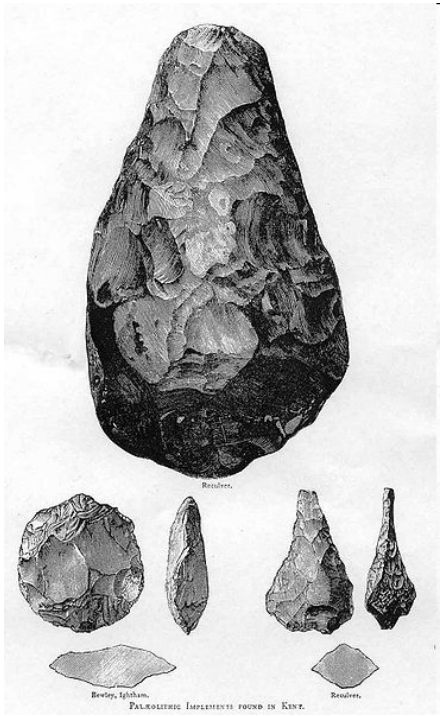
Dead-locks

Live-locks

Race conditions

Starvation

Shared Mutable State



Why high-level concurrency?

Multithreaded programs today work mostly by accident!

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/