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

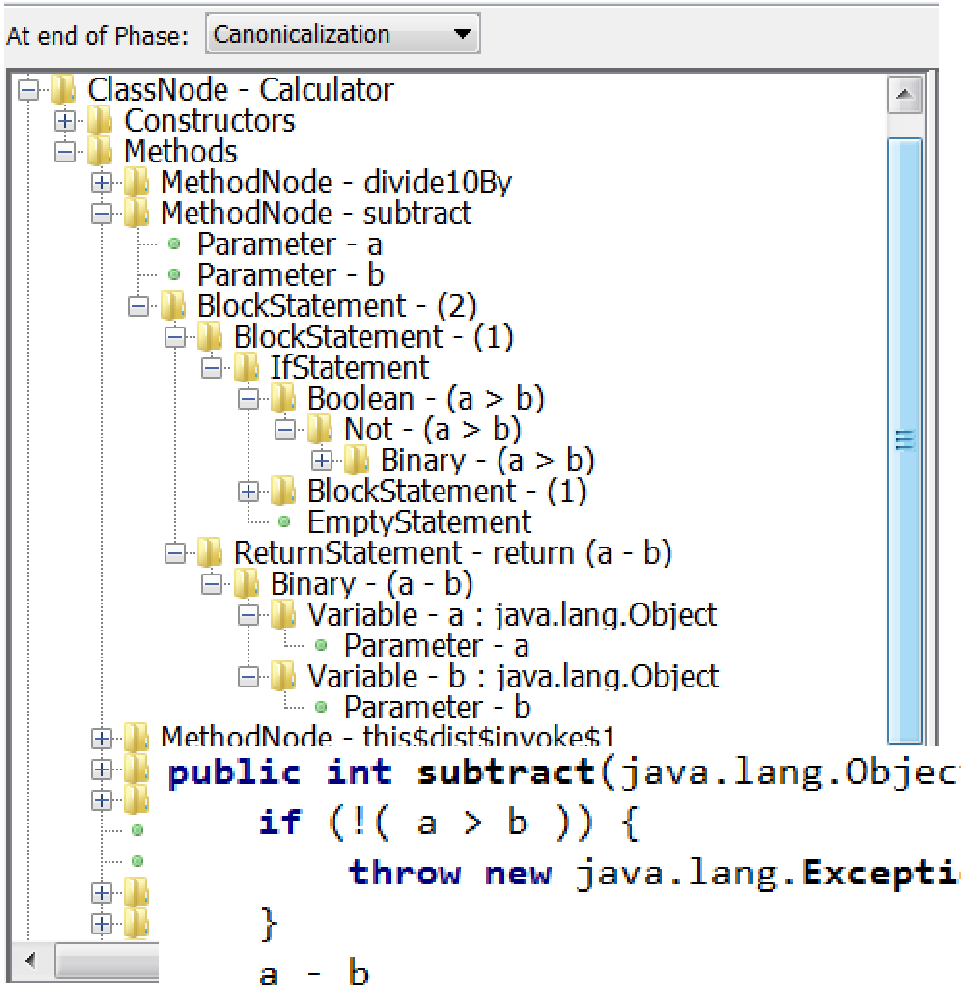
Agenda for today

- AST transformations
- Static meta-programming
- Concurrency primer

Part 5

Static meta-programming

AST



At end of Phase: Canonicalization

```
ClassNode - Calculator
  Constructors
  Methods
    MethodNode - divide10By
    MethodNode - subtract
      Parameter - a
      Parameter - b
      BlockStatement - (2)
        BlockStatement - (1)
          IfStatement
            Boolean - (a > b)
              Not - (a > b)
                Binary - (a > b)
            BlockStatement - (1)
            EmptyStatement
        ReturnStatement - return (a - b)
          Binary - (a - b)
            Variable - a : java.lang.Object
              Parameter - a
            Variable - 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}')
    }
    a - b
}
```

AST Transformations

```
class Registrations {
    @Delegate List items = []
}

def people = new Registrations()
people.addAll(["Joe", "Dave"])
assert ["Dave", "Joe"] == people.reverse()
```

@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"
                     }
      }  }  }  }  }
```

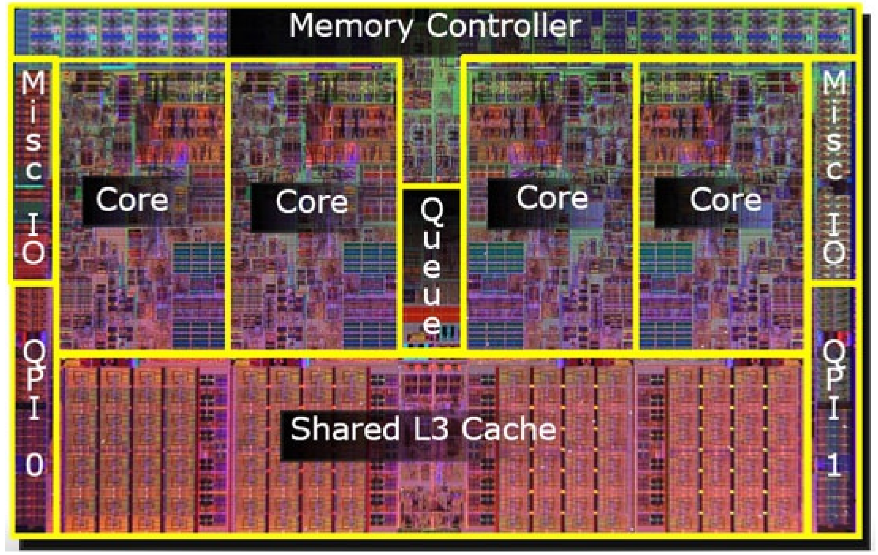
Type-checking/Static

@TypeChecked, @CompileStatic

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

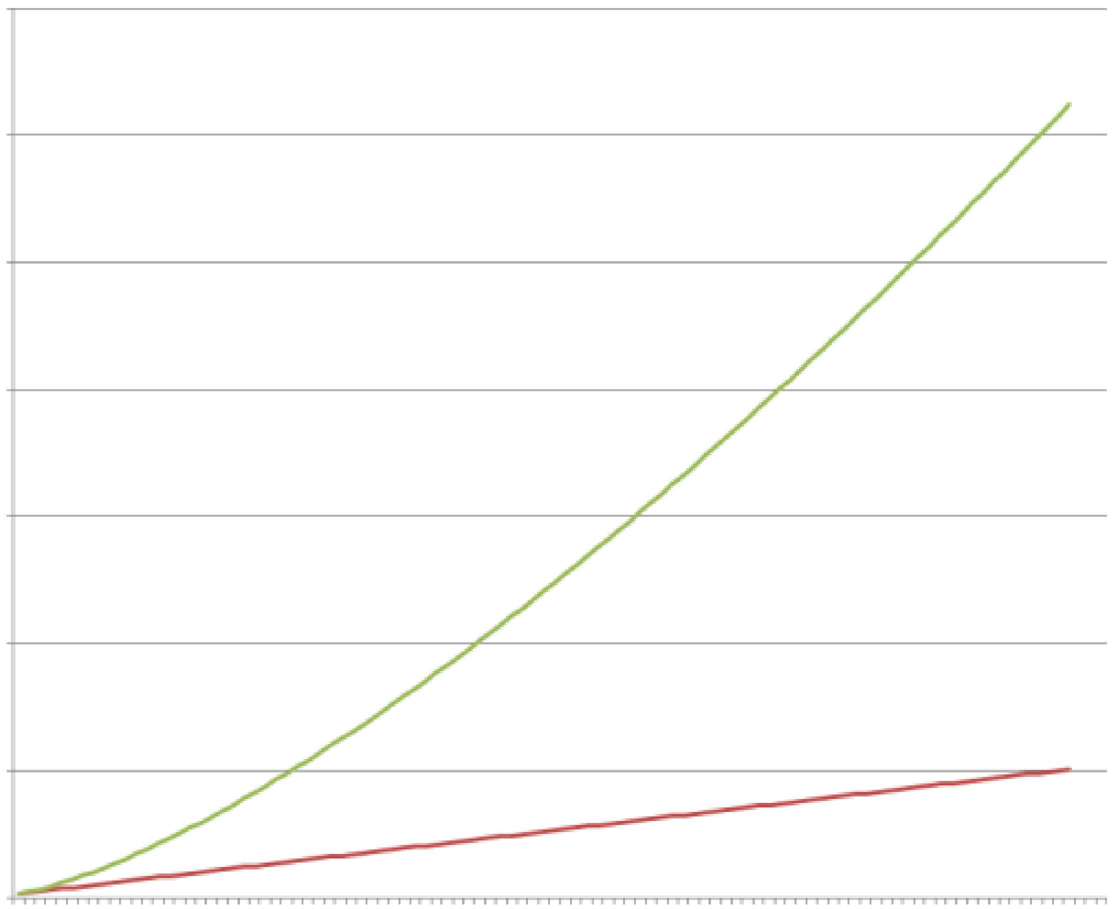
Concurrency

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

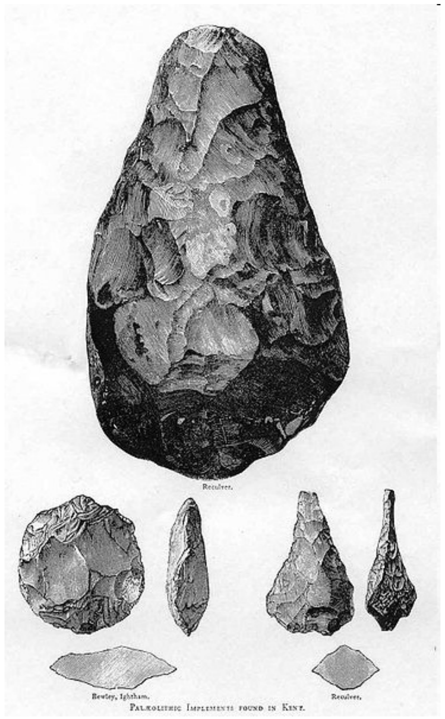
Dead-locks

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/