>CONFESS_2011
CONference For
Enterprise Software
Solutions_

GContracts

Programming by Contract with Groovy

Andre Steingress

Andre Steingress

- Independent Software Dev
- @sternegross, @gcontracts

- Groovy, Grails, JEE, Spring Portfolio, Android

- http://blog.andresteingress.com
- GroovyMag, JavaMagazin (German)

Motivation

Programming is about assumptions, mental models and knowledge.

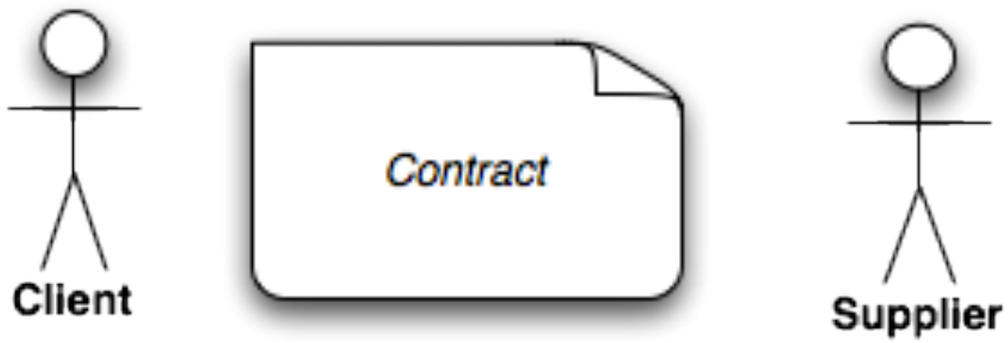
Motivation

„Contracts help to externalize mental models and formalize implicit assumptions."

- How to formalize assupmtions?
  - **Logic** is the formal study of **correct reasoning**
  - **Boolean Algebra** is the tool of choice

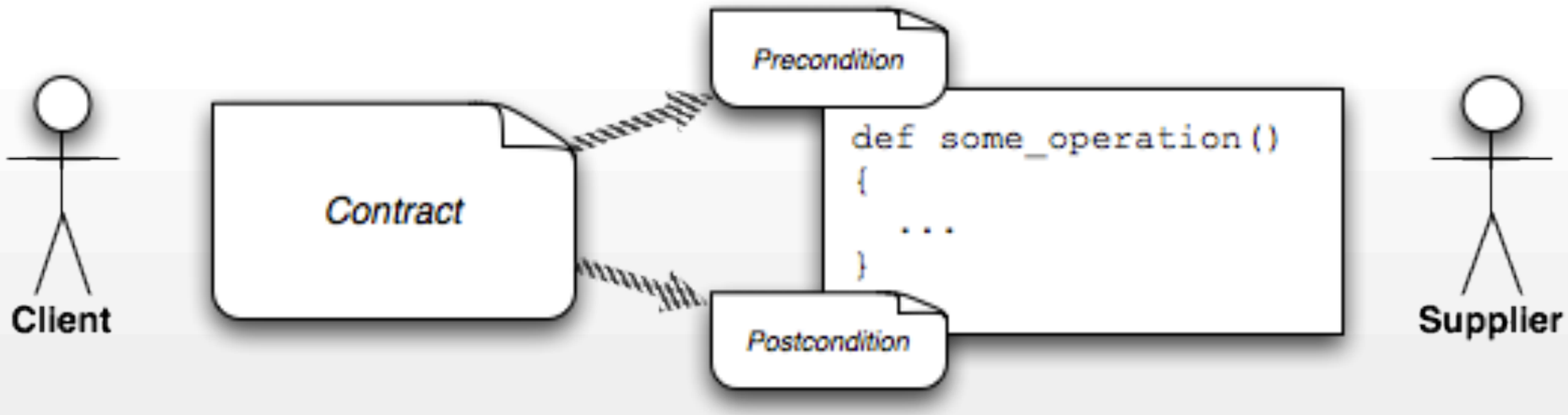
Basic Terms

- **Supplier** implements a software element (e.g. a class)
- **Client** references the software element



Basic Terms

- A **contract** consists of pre- and postconditions for each public method
  - **Precondition** expresses the constraints under which a method call will function properly
  - **Postcondition** expresses properties of the state resultung from a method‘s execution
- **ContractViolation** is thrown if the contract is broken



GContracts

- **Design by Contract (tm)** for Groovy
- Version **1.2.1**

- **Compile-Time** Contract Injection
- **Power Assert** Support

- **Maven** Central Repository

- **pure Java library, no dependencies**
  - gcontracts-core-1.2.1.jar

- **dynamic jvm language**
- **power features** from Ruby, Smalltalk, Python et. al.
- **reducing Java „boilerplate“ code**

- **DSL friendliness** with **closures**
- **seamless integration** with **Java libraries** and **classes**

Prerequisites

- contracts are specified **applying annotations**
- Groovy supports **closures**

```
def c = {
   println "hello world"
}
c.call()
```

- **Annotation Closures**
  - only syntactically supported in Groovy 1.7
  - officially supported in Groovy 1.8

```
@RunIf({ jdkVersion >= 6 })
```

Demo

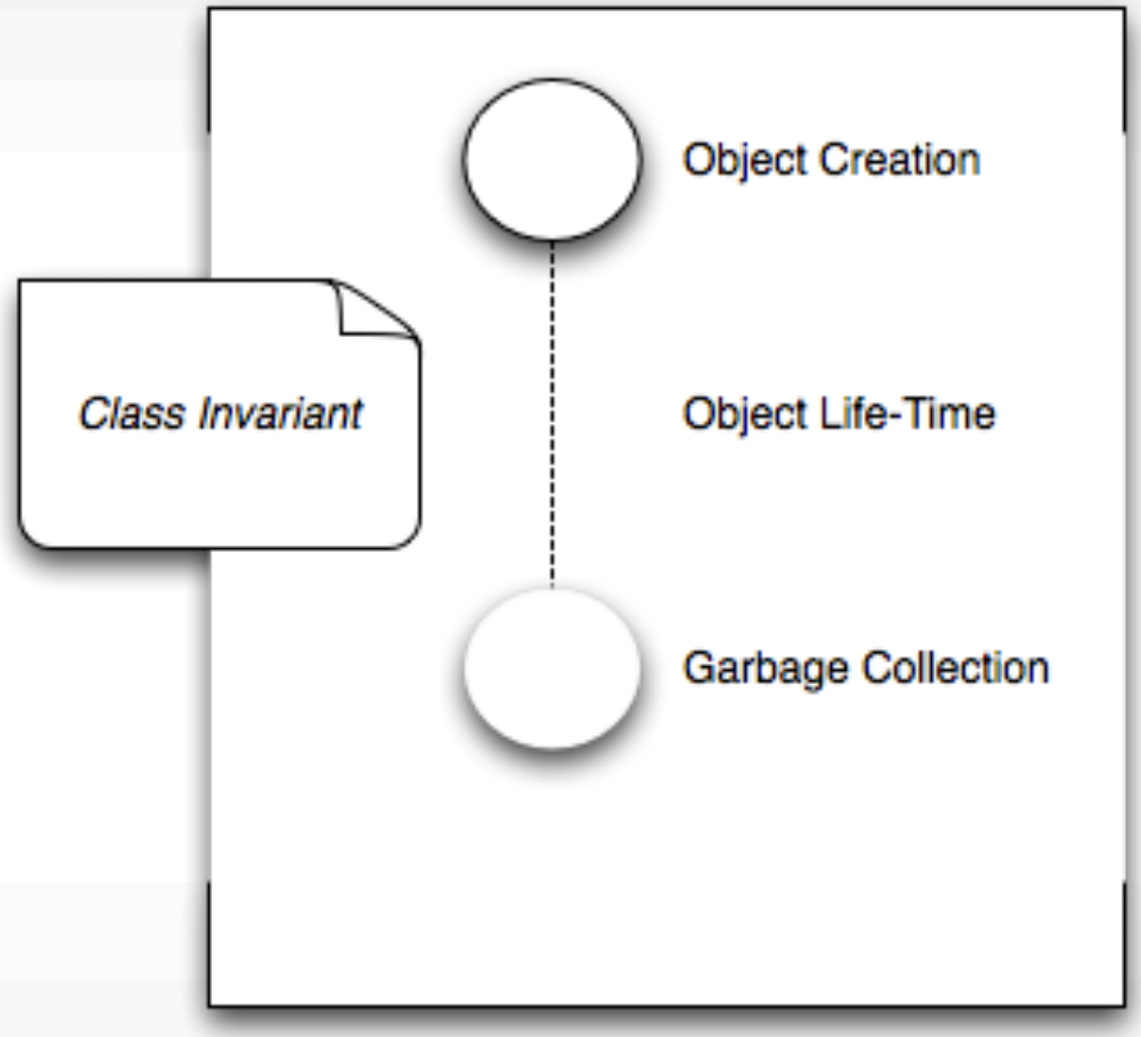
Internals

- Groovy AST Transformation
  - modifies the abstract syntax tree (AST) during compilation runs

- Uses closures as annotation arguments
  - only syntactically supported in Groovy 1.7
  - officially supported in Groovy 1.8
    - Compiles annotation closures to closure classes
    - References compiled closure classes in annotation args

- **Groovy AST Browser** shows the outcome!

- **Class Invariants**

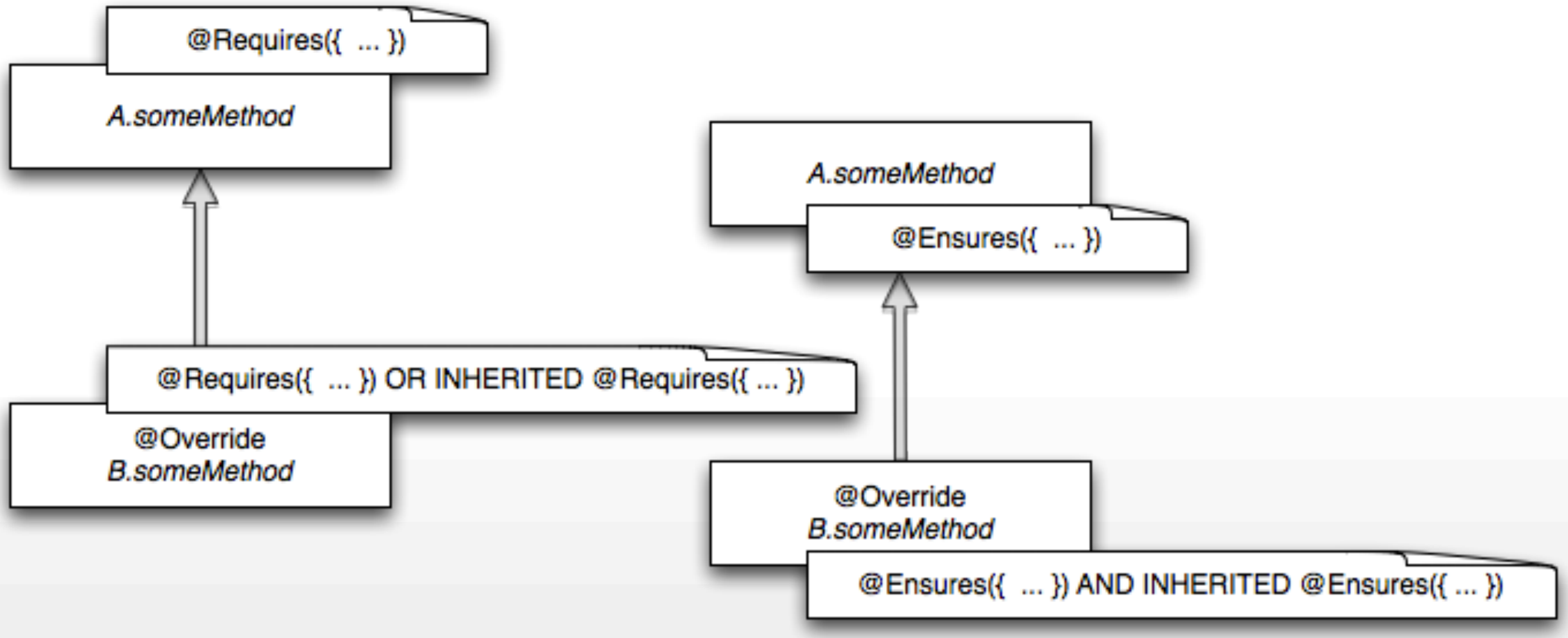


- **Contract Inheritance**

```
import org.gcontracts.annotations.*

abstract class Stack {

  protected internal = []

  @Requires({ item != null && !contains(item) })
  @Ensures({ last() == item })
  def push(def item)  {
    internal << item
  }
  // ...
}

class StackImpl extends Stack {
  // ...
}
```

GContracts 1.2.1

- **Contract Inheritance**
  - Precondition gets **weakened**
  - Postcondition gets **strengthened**



- **Interface Contracts**

```
import org.gcontracts.annotations.*

interface Stack {

  @Requires({ item != null && !contains(item) })
  @Ensures({ last() == item })
  def push(def item)

  @Requires({ !isEmpty() })
  def pop()

  def last()
  boolean isEmpty()
  boolean contains(def item)
}
```

GContracts 1.2.1

- **Labels**

```
@Requires({
    not_empty: !isEmpty()
})

@Ensures({
    not_full:           !isFull()
    descrease_counter: old.count == count - 1
})
```

GContracts 1.2.1

- Module **gcontracts-grails**
  - Spring Integration

```
import org.gcontracts.annotations.*
import org.springframework.stereotype.*

@Component
@Invariant({ property?.size() > 0 })
class MySpringBean {

  protected property

  def MySpringBean(def someValue)  {
    property = someValue
  }
}
```

GContracts 1.2.1

- Module **gcontracts-doc**
  - GroovyDoc extension

Constructor Summary

EiffelStack()
EiffelStack(List preElements)

Method Summary

def	count()
boolean	@Ensures({ result == true ? count() > 0 : count() >= 0 }) has(def item)
boolean	is_empty()
def	@Requires({ !is_empty() }) last_item()
def	@Ensures({ last_item() == item }) put(def item)
def	@Requires({ !is_empty() }) @Ensures({ result != null }) remove()
def	@Requires({ !is_empty() }) @Ensures({ last_item() == item }) replace(def item)

- Reusing Contracts: **Annotation Contracts**

```
import org.gcontracts.annotations.meta.*
import java.lang.annotation.*

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.PARAMETER)

@Precondition
@AnnotationContract({ it != null })
public @interface NotNull {}
```

GContracts 1.2.1

- Reusing Contracts: **Annotation Contracts**

```
class BankAccount {

    @Requires({ amount > 0.0 })
    BigDecimal withdraw(@NotNull BigDecimal amount)
    {
        if (balance >= amount)
        {
            balance -= amount;
            return amount;
        }
        else
            // Withdrawal not allowed
            return 0.0;
    }
    // ...
```

GContracts 1.2.1

- **More ...**
  - VM Args to Enable/Disable Contract Checking
  - Domain Model
  - Custom Annotation Processors
  - Cyclic Call Detection
  - Gradle Multi-Build Project
  - ...

IDE Support

- ✓ IntelliJ
- Groovy Eclipse Plugin
  - Snapshot for Groovy 1.8
    - includes support for annotation closures

MythBusters

- Don't know how to apply contracts in my application!

- My Methods are too complicated for contracts!

- I write unit tests, let's go bowling!

Where to get it?

- BSD license
- Github project (source/binaries)

- Available in the Central Maven Repo
  - `@Grab(`
    `group='org.gcontracts',`
    `module='gcontracts-core',`
    `version='[1.2.1,)'`
    `)`
  - Ivy
  - Maven
  - Gradle

- Lighthouse Issue Tracking

Any Questions?

Thank you!

me@andresteingress.com

@sternegross

@gcontracts