Contracts as a basis for stigating reusability of Smalltalk code

Koen De Hondt
Programming Technology Lab
Computer Science Department
Vrije Universiteit Brussel

@vub.ac.be http:/progwww.vub.ac.be/

view

- **lems with reuse**
- lems with evolution
- t are reuse contracts?
- e contracts at work
- nining class hierarchies based on
e contracts
- e contract research
- cises: introduction to the browser

do You Reuse a Class?

ing (copy and paste)

ritance / method overriding

position / delegation

e by Cloning

“components” are not easily
ole

pport is provided for adaptation/reuse

tion between original and result

ult to maintain since bug fixes and
des are not propagated to the derived
cation (proliferation of versions)

This kind of reuse should be avoided

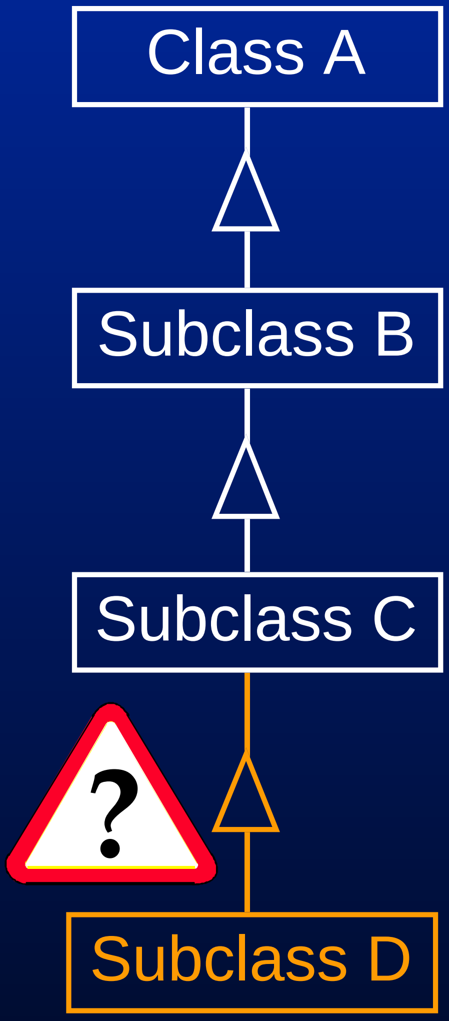
e by Inheritance

you determine

t to reuse (inherit)?

t to adapt (override)?

t to write from scratch?



nple: *Make a Subclass of*

k
dd:
ddAll:
gSet

What to override?

- □ #add: if #addAll: uses #add:
- □ #add <u>&</u> addAll: if #addAll: does not use #add:

A CountingSet is a Set that counts all added elements

e by Composition

you determine

to reuse (what to compose, what to
ate)?

to adapt (how to compose)?

to write from scratch?

Class A
?
Class B
Class C

ing a Class is Hard

OOA/OOD notations do not provide information to reuse a class

developers do not document how a class eused, they only document what each does

comment contains reuse information, it has the form of a cookbook

Reusers are compelled to inspect the source code

ecting the Source Code

use a class:

pect the class

pect all its superclasses

pect all the classes it co-operates with

ce code inspection is error-prone

urce code inspection doesn’t work:
o the developer (i.e. the expert)!

t are You Looking for?

nds

ends

t methods

te methods

methods

s that are overridden frequently

s that are part of a design pattern

ration with other objects/classes

Reusers need the specialisation interface

Sends are Important

ds & template methods & abstract
s reify the design of a class

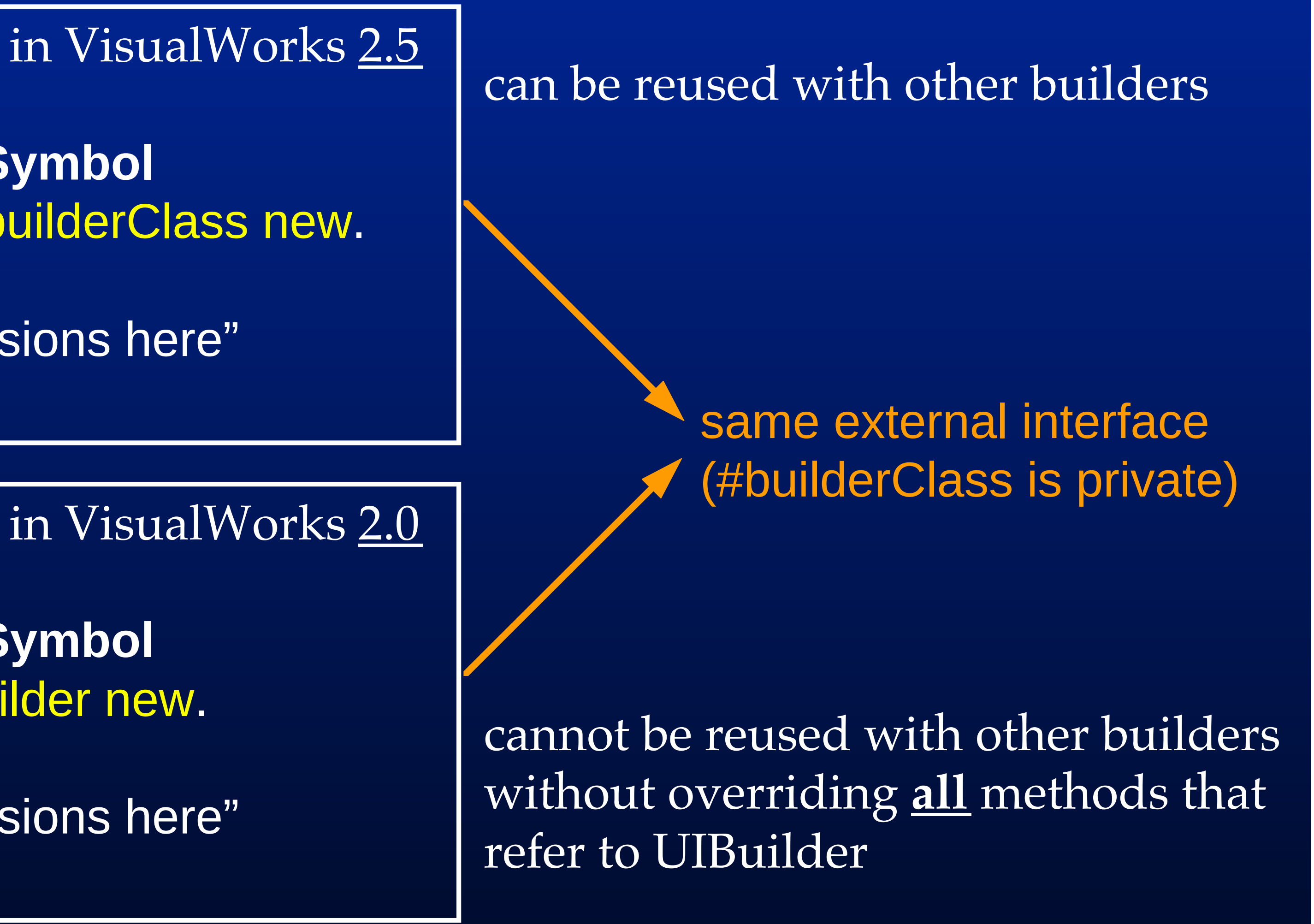
decomposition

uish “core” methods from “peripheral”
ds

lf sends = planning for reuse

ained overriding of methods

Sends: Planning for Reuse



in VisualWorks 2.5

Symbol
builderClass new.

sions here”

in VisualWorks 2.0

Symbol
ilder new.

sions here”

can be reused with other builders

same external interface
(#builderClass is private)

cannot be reused with other builders without overriding **all** methods that refer to UIBuilder

peration with Other
cts/Classes is Important

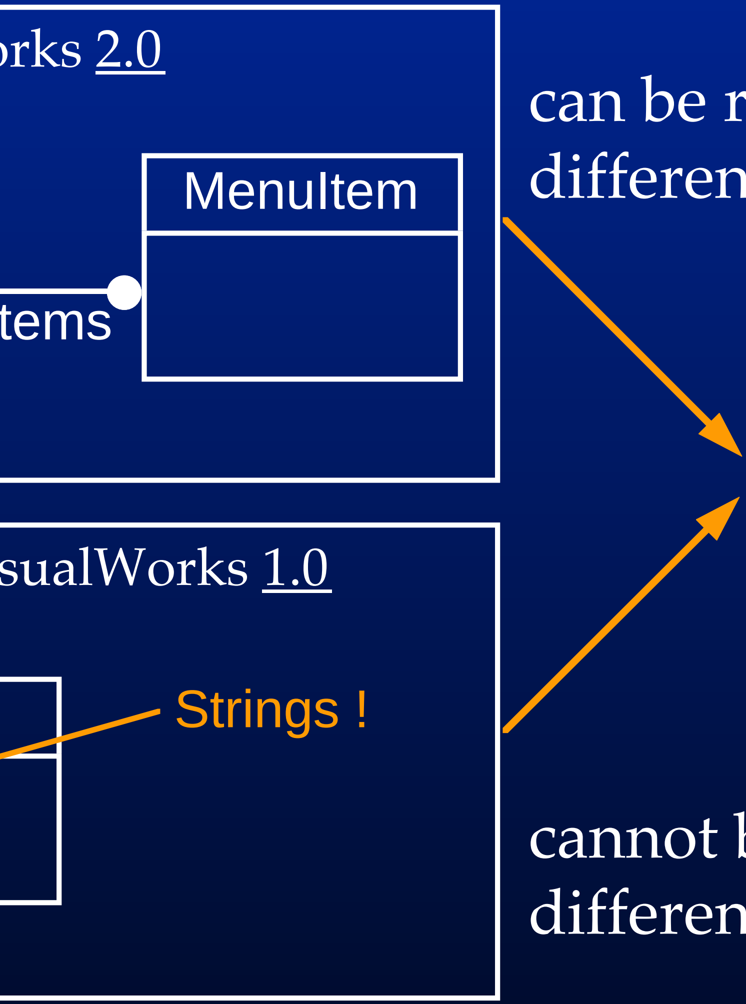
gation of responsibilities principle

g delegation= planning for reuse

ystem can easily be extended by adding
v classes

ects with “the same interface” can be
bstituted for each other

ation: Planning for Reuse



can be reused for
different menu items

same external behaviour
same interface
for instance creation

cannot be reused for
different menu items

view

lems with reuse

lems with evolution

t are reuse contracts?

e contracts at work

nining class hierarchies based on
e contracts

e contract research

cises: introduction to the browser

ution is Important

tive development

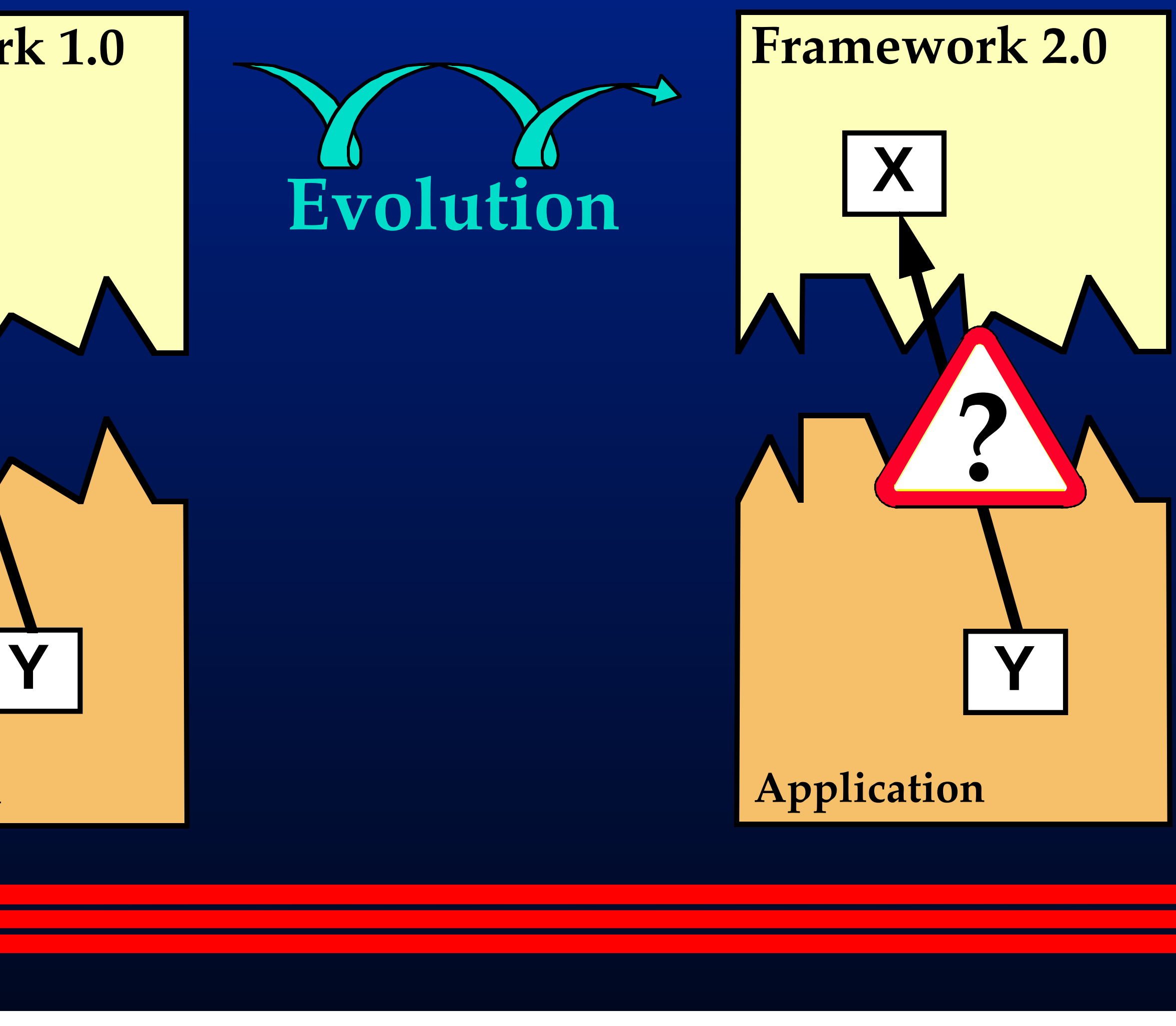
amework is never finished

ging requirements

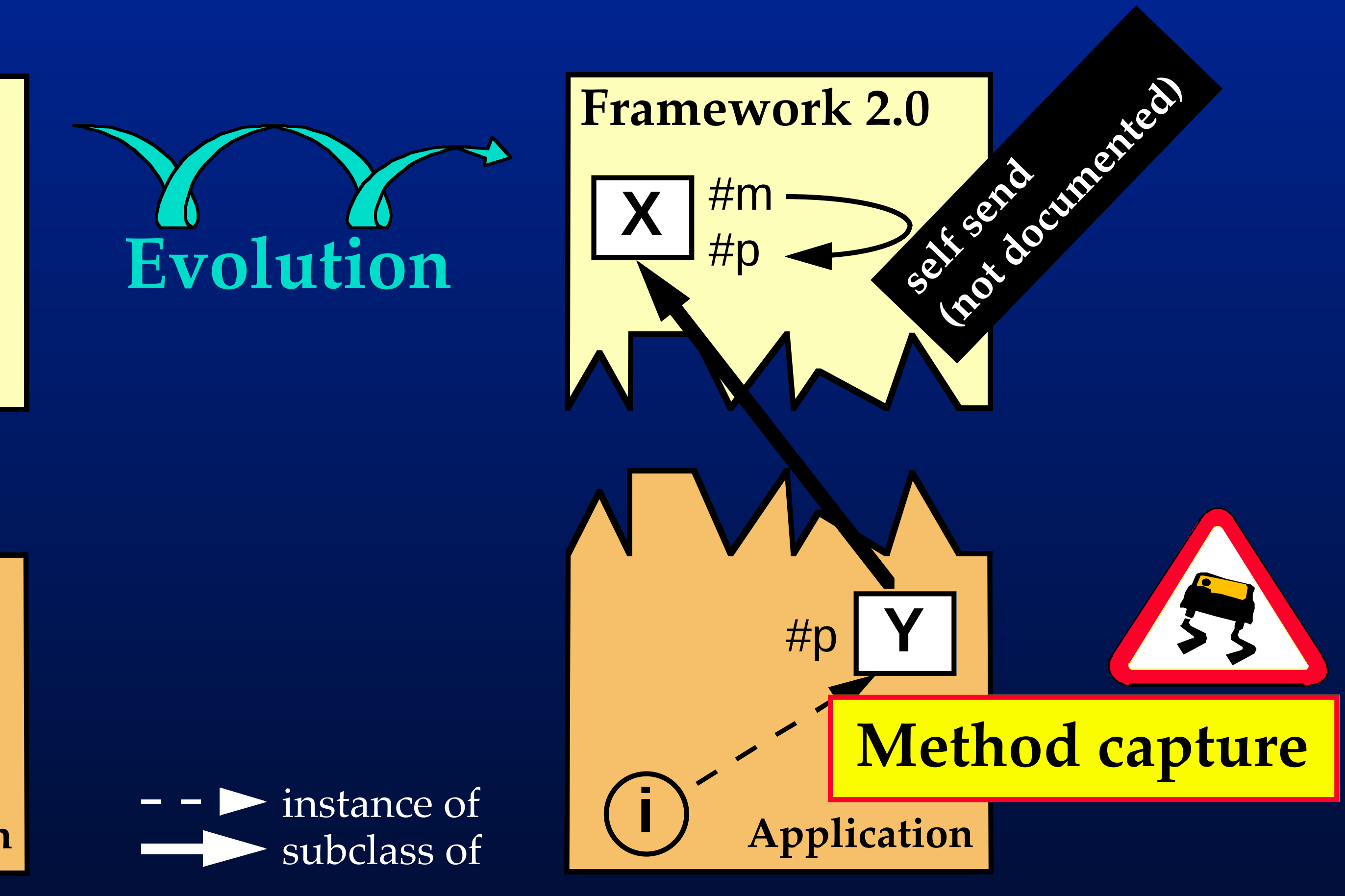
ctional: user requirements

n-functional: maintainability, ptibility, reusability, customisability, ...

t to do When the ework Changes?

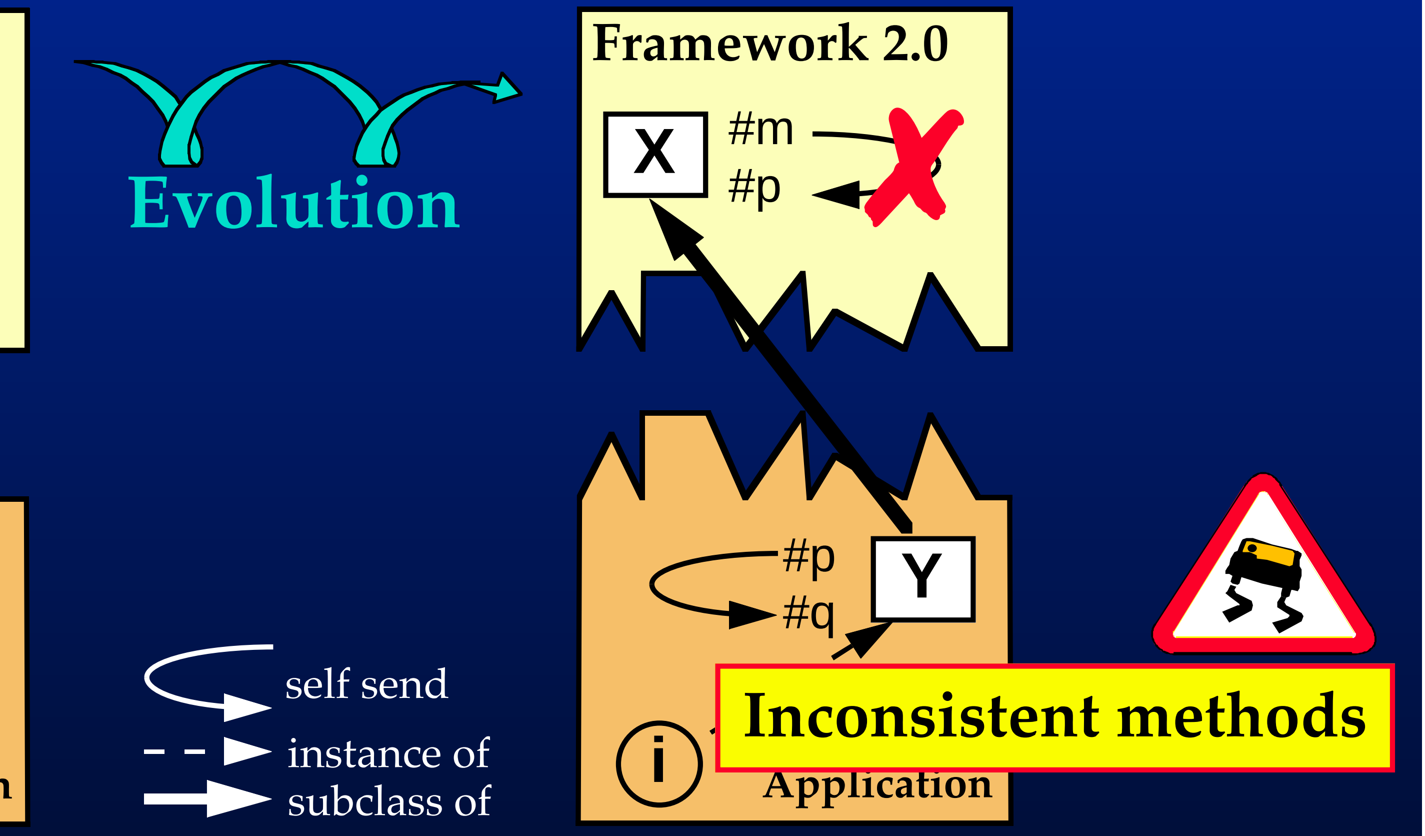


ple Evolution Conflict (1)



Method capture

ple Evolution Conflict (2)



Evolution

Framework 2.0

X

#m

#p

Y

#p

#q

i

Application

self send

instance of

subclass of

Inconsistent methods

e Evolution Conflicts

ace conflicts

name of a reused method / class has been nged

ethod that was added by a reuser has been oduced by the new version of the framework

ticipated recursion

ethod invokes another one in the application le the new version of the framework introduces nvocation of the first by the latter

ting Evolution Problems

ss the changes to the framework
vell-documented (informally), the
cation developer is condemned to
rm code inspection to determine
has changed

evolution conflicts are not spotted
the application is running based on
ew version of the framework

t are the Challenges?

- ting reuse
  - an be reused, what must be adapted, and what e built from scratch ?
  - documentation on how classes are reused
- ting evolution
  - propagation
- t for estimates/testing/metrics
  - ity of reusing a class
  - t of "upgrading" the class repository

view

lems with reuse

lems with evolution

t are reuse contracts?

e contracts at work

nining class hierarchies based on e contracts

e contract research

cises: introduction to the browser

e Contracts

contracts between the framework
loper and the application
loper

what assumptions can be made
t reusable components

how components are actually
ed

e Contract Notation

on based on OMT (UML)

ds are annotated with
isation clauses to make the
isation interface explicit

e operators", or "modifiers", lay
how reuse is achieved

e Contracts for Inheritance

interface of a class
sation clauses

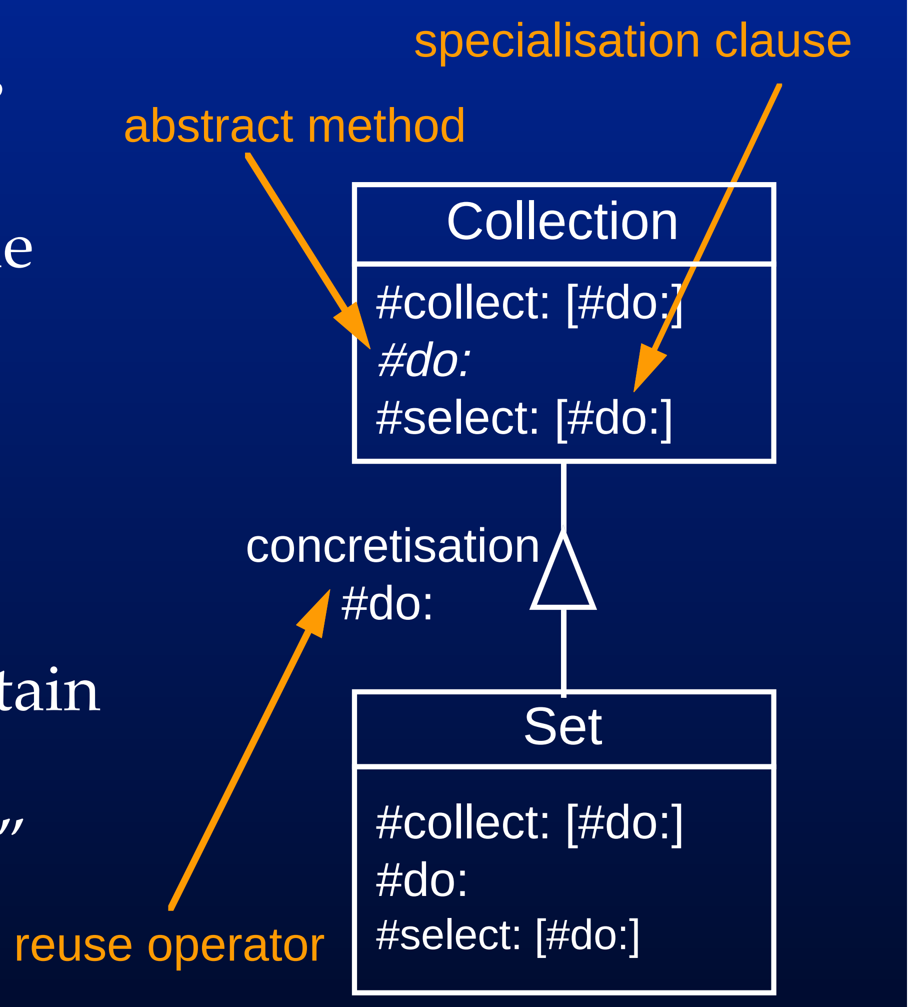
t changes are made
 is subclassed:

tion/abstraction

/cancellation

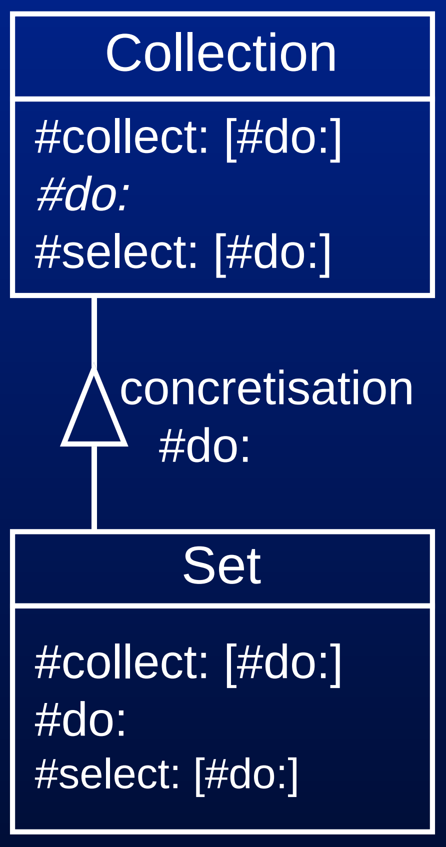
t/coarsening

n clauses may contain
thods invoked
sends, and “super”



e Operator: Concretisation

bstract methods
e
t change the
sation clause of
cretised methods
preserving
= abstraction

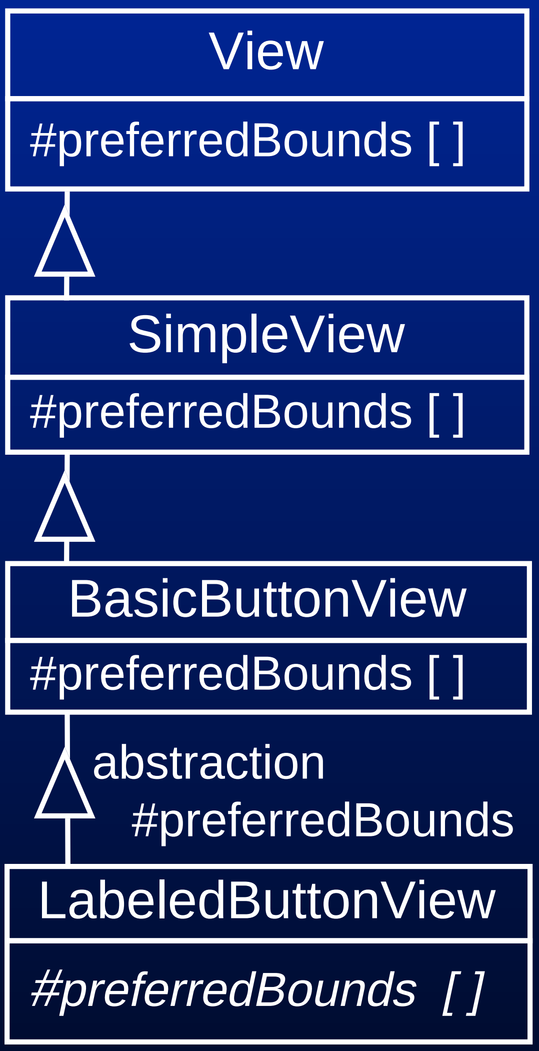

Collection
#collect: [#do:]
#do:
#select: [#do:]
concretisation
#do:
Set
#collect: [#do:]
#do:
#select: [#do:]

e Operator: Abstraction

oncrete method

eaching

concretisation



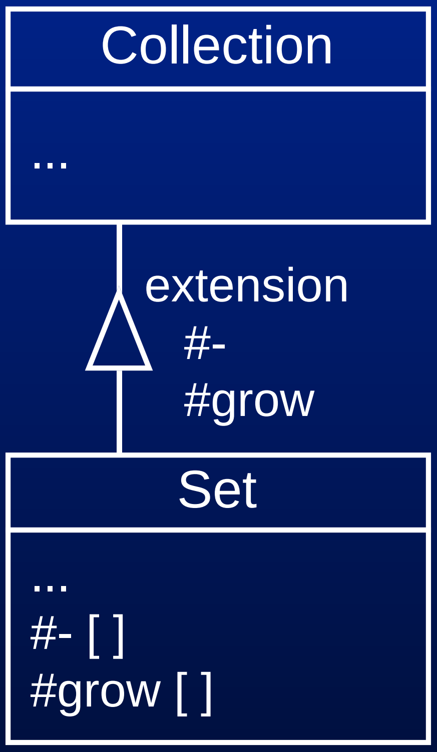
e Operator: Extension

performed by an
n developer to add
n specific behaviour

methods to the
f a class

eserving

cancellation

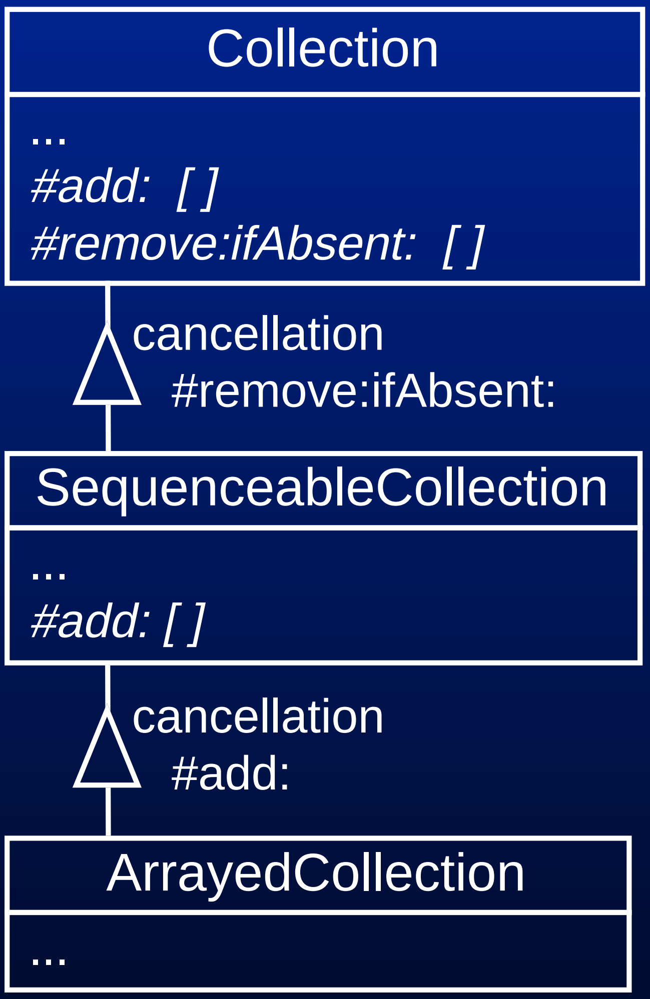


e Operator: Cancellation

performed by an
on developer to
behaviour

methods from
ace of a class
reaching

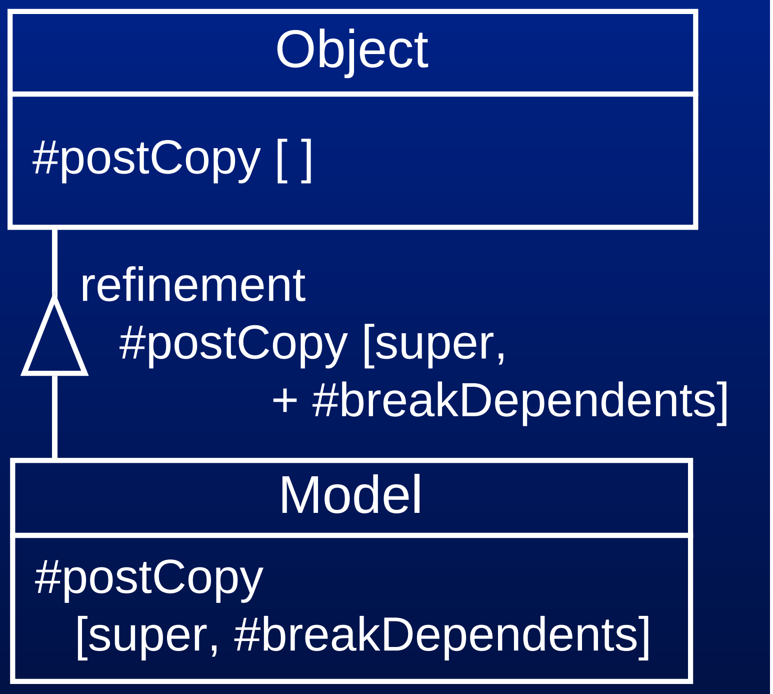
extension



e Operator: Refinement

ents to the
ion clause of a

g. :
dundancy
 the behaviour of
g method with an
behaviour
eserving
coarsening


Object

#postCopy [ ]

refinement
#postCopy [super,
+ #breakDependents]

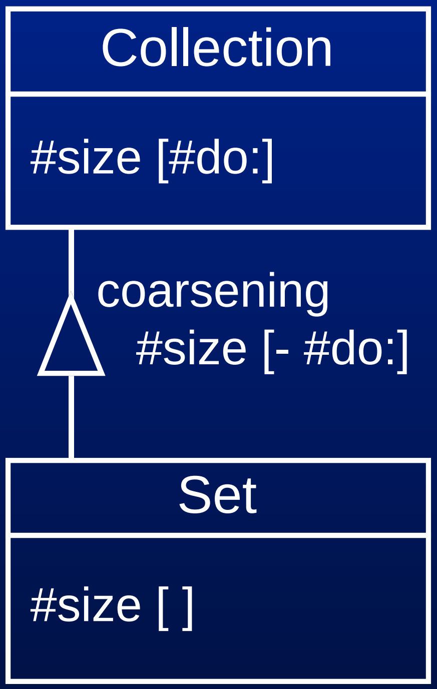
Model

#postCopy
[super, #breakDependents]

e Operator: Coarsening

elements from
isation clause of

g.:
performance
eaching
refinement



Collection

#size [#do:]

coarsening
#size [- #do:]

Set

#size [ ]

e Operators

e a distinction between different
s of inheritance

how a class is derived from its
rclass

rthogonal basic operators

lly, one subclassing step is a
ination of several reuse operators

uently Used Combinations
ruse Operators

on & refinement

ning & cancellation

tisation & refinement

tisation & extension & refinement

ning & refinement = redefinition

ning & extension & refinement
rization

i-Class Reuse Contracts
ort)

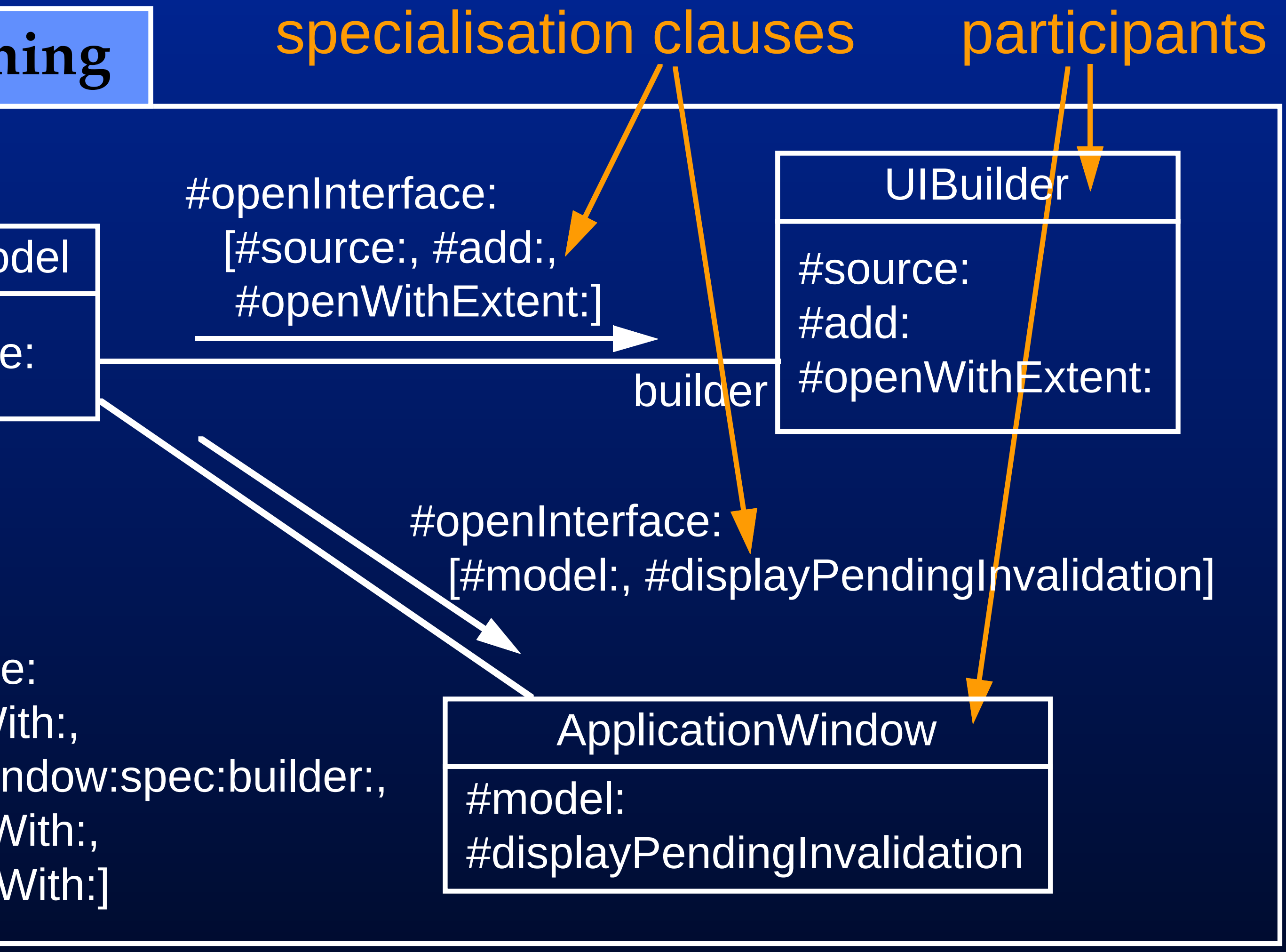
ating classes are put in one reuse
these classes are called “participants”

s of classes as in reuse contracts for
nce

ation clauses are extended with names
ds invoked on other classes

erators identify what changes are
a <u>whole</u> contract

e Contract Notation



ning

specialisation clauses

participants

odel

e:

#openInterface:
[#source:, #add:,
#openWithExtent:]

builder

UIBuilder

#source:
#add:
#openWithExtent:

#openInterface:
[#model:, #displayPendingInvalidation]

e:
ith:,
ndow:spec:builder:,
With:,
With:]

ApplicationWindow

#model:
#displayPendingInvalidation

view

lems with reuse

lems with evolution

t are reuse contracts?

e contracts at work

nining class hierarchies based on e contracts

e contract research

cises: introduction to the browser

e Contracts at Work

rmal nature of reuse contracts
s their use in a development
nment

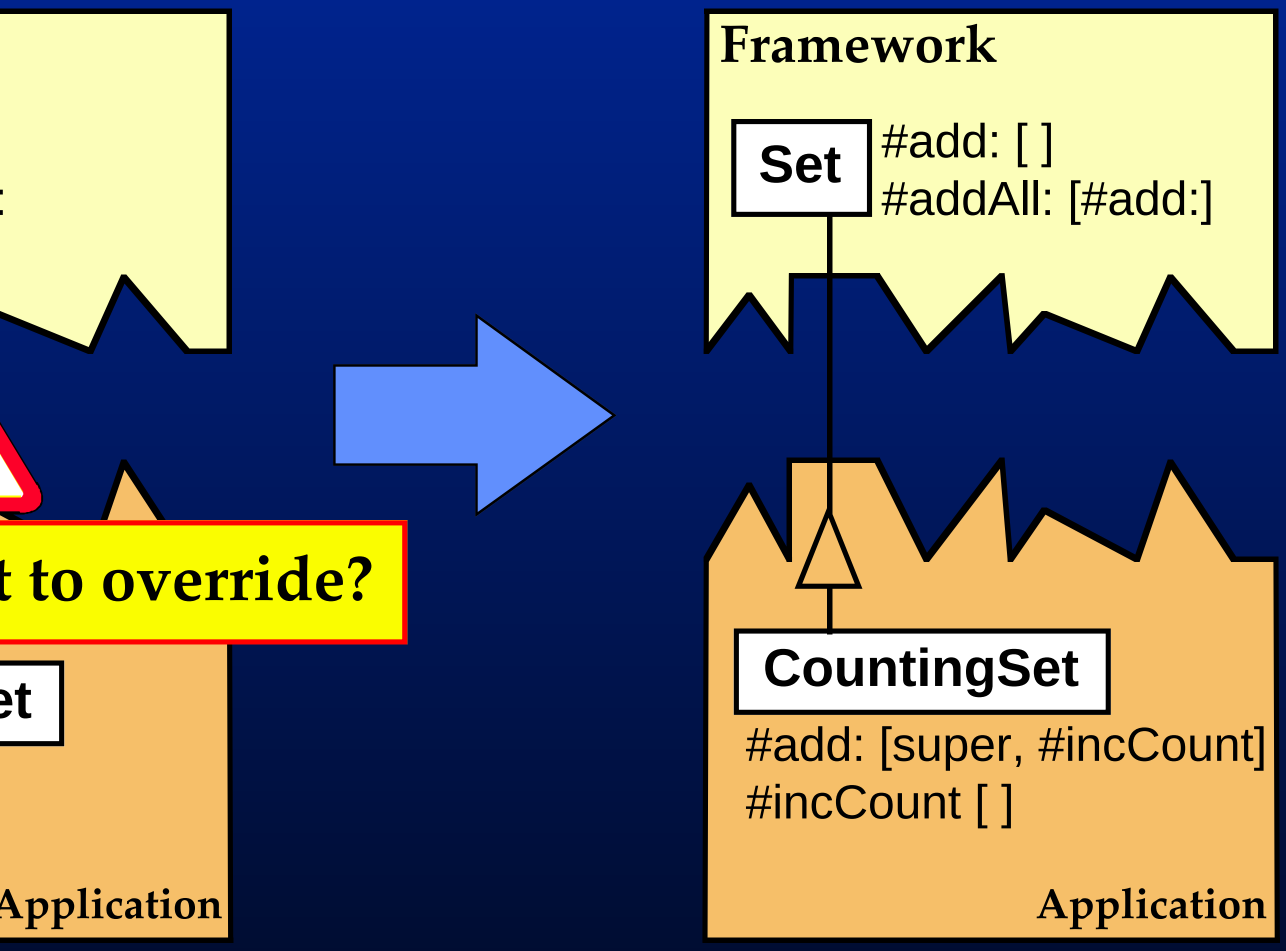
le generation from reuse contracts

pact analysis when a framework
nges (assessing evolution conflicts)

ort estimation for framework
stomisation

raction from source code

nating Reuse



t to override?

Framework

Set

#add: [ ]

#addAll: [#add:]

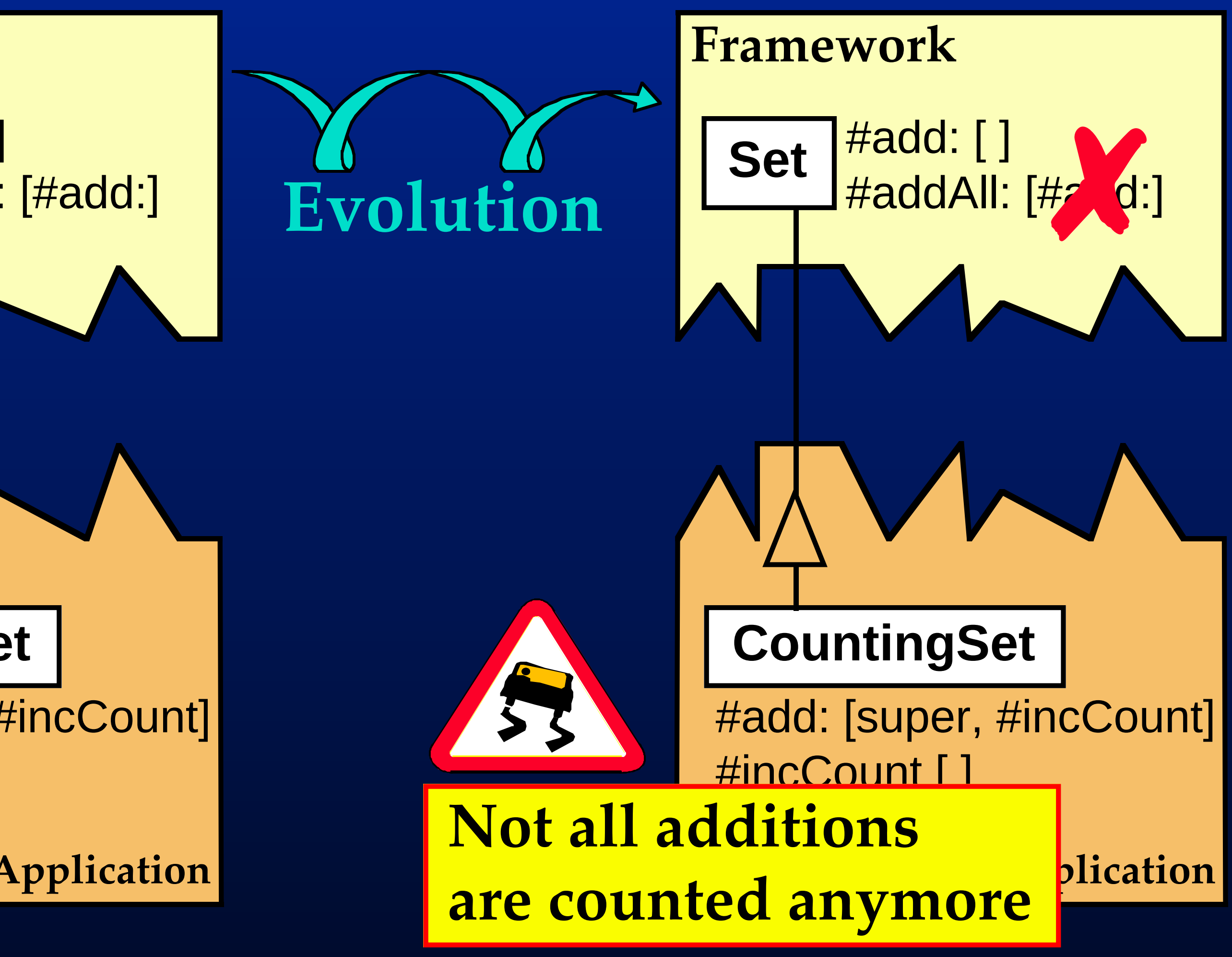
CountingSet

#add: [super, #incCount]

#incCount [ ]

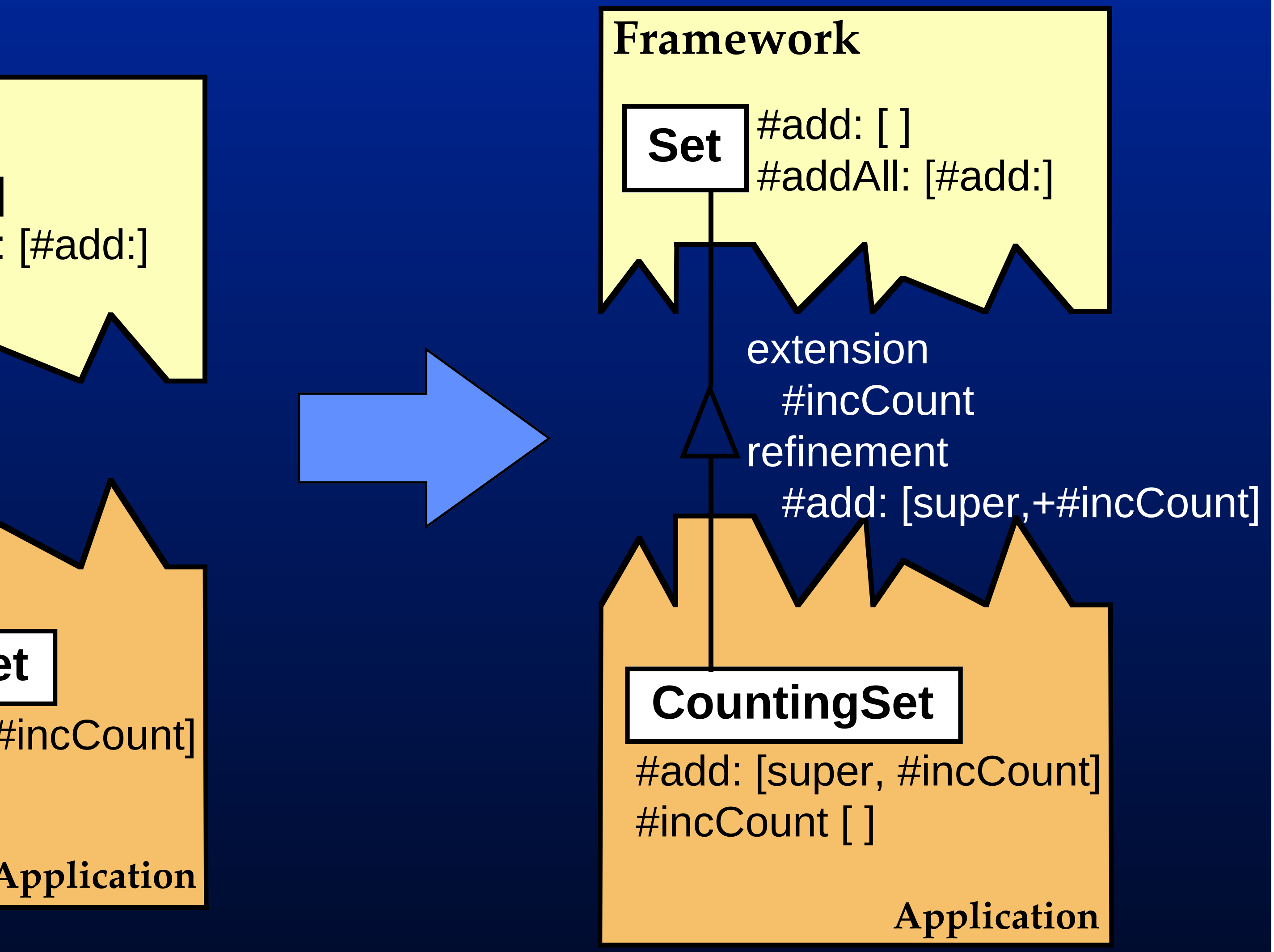
Application

ution

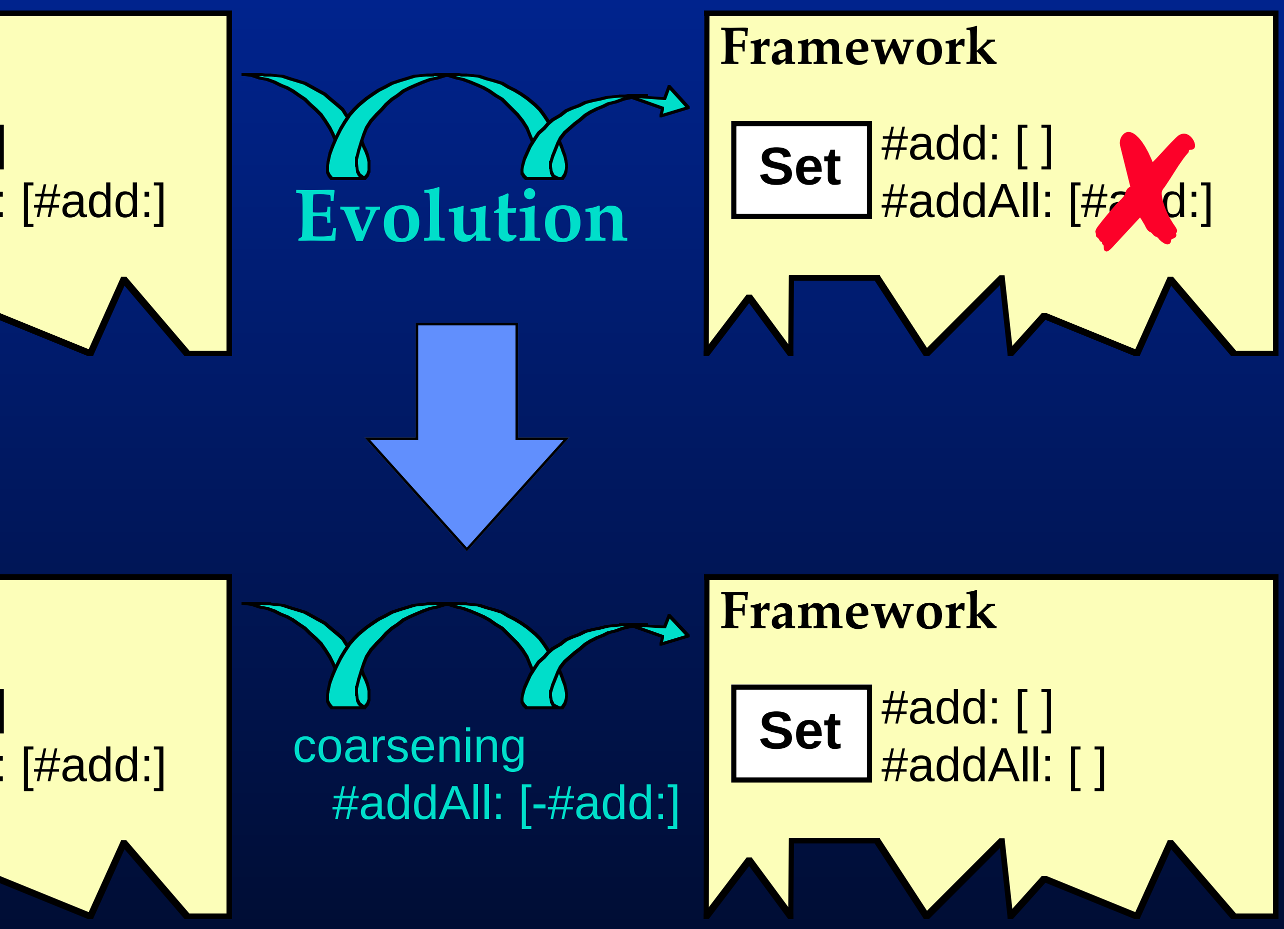


Not all additions are counted anymore

menting Reuse



menting Evolution



: [#add:]

Evolution

Framework

Set #add: [ ]
#addAll: [#add:]

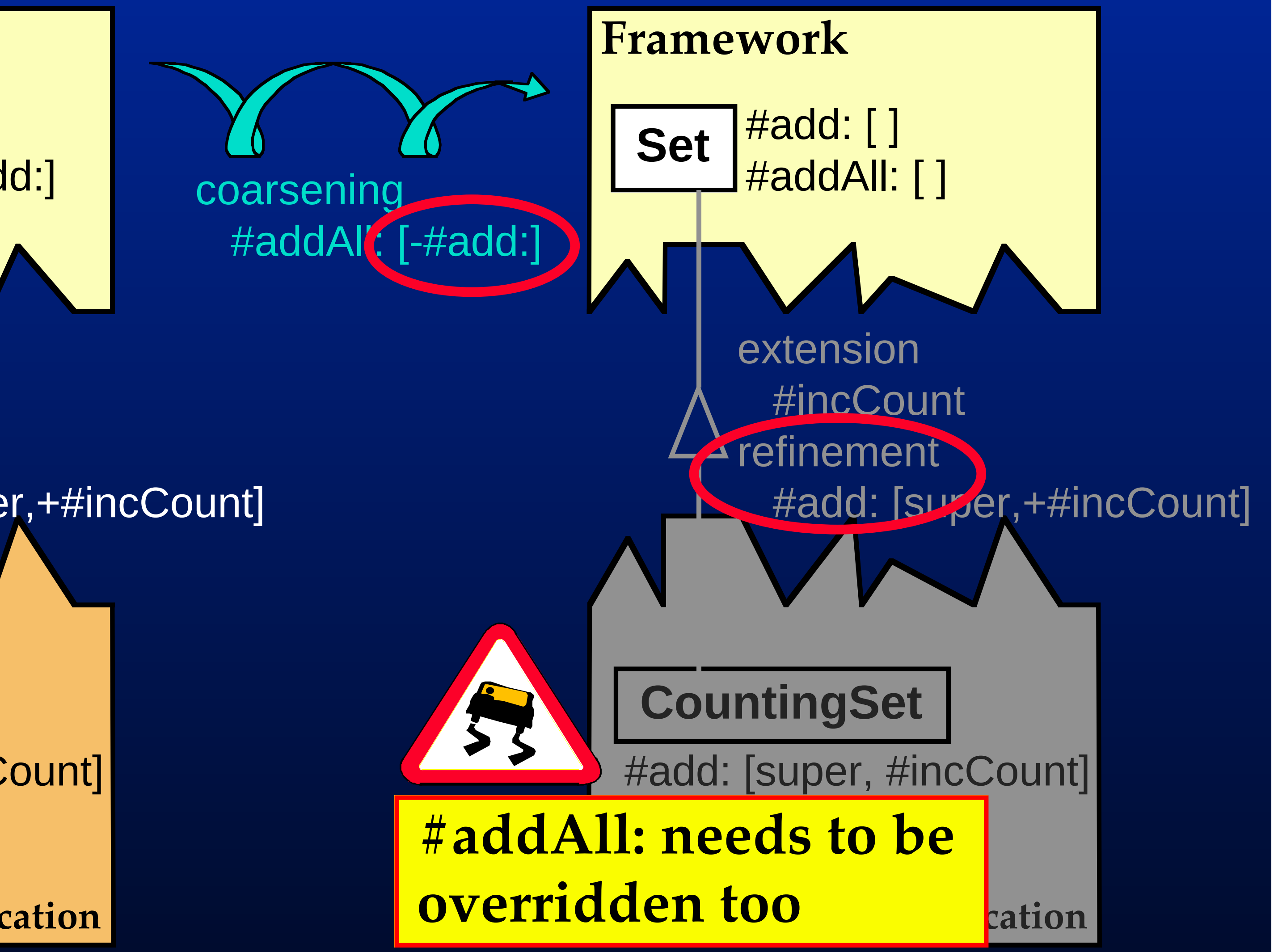
: [#add:]

coarsening
#addAll: [-#add:]

Framework

Set #add: [ ]
#addAll: [ ]

nating Impact of Changes



#addAll: needs to be overridden too

view

lems with reuse

lems with evolution

t are reuse contracts?

e contracts at work

nining class hierarchies based on e contracts

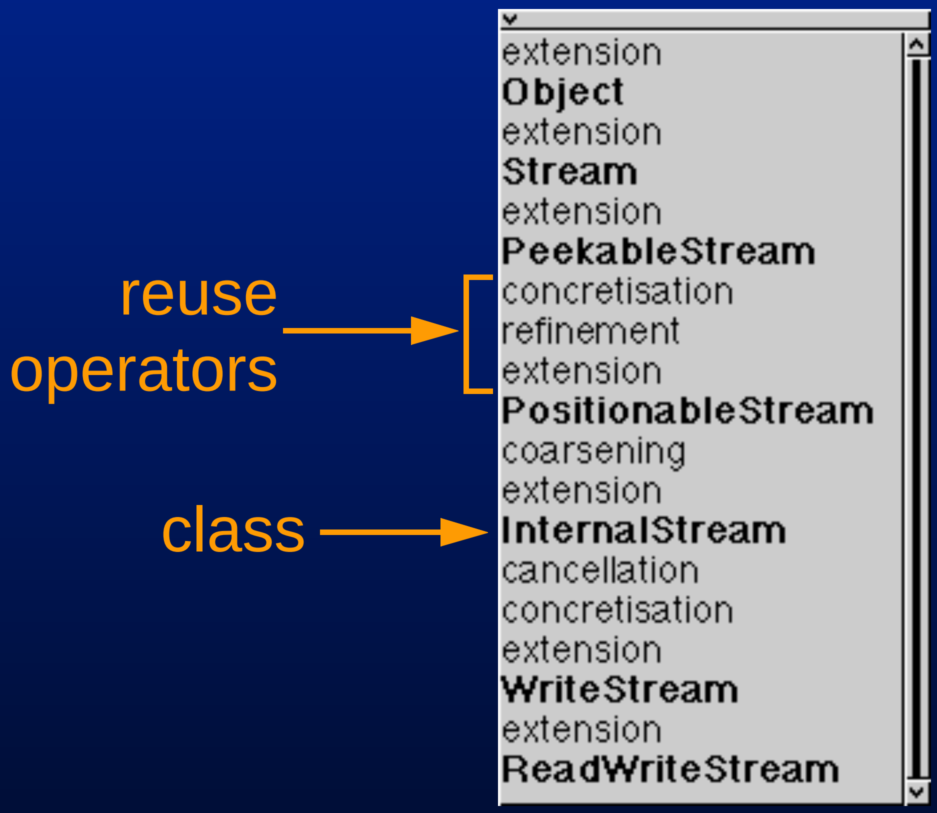
e contract research

cises: introduction to the browser

ction of Reuse Contracts

ntracts for
ce can be
from
code

classing step
posed in a
ion of
n 6 different
rators

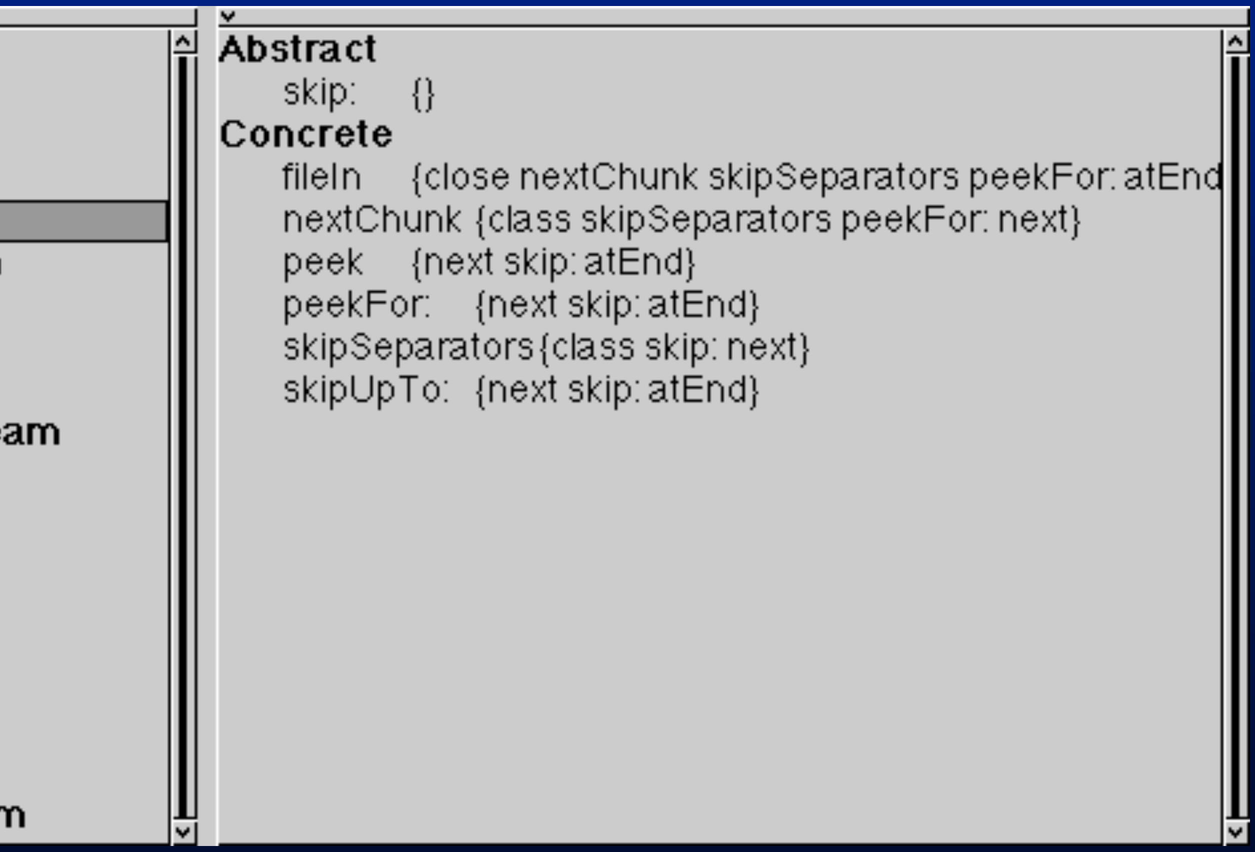


extension
Object
extension
Stream
extension
PeekableStream
concretisation
refinement
extension
PositionableStream
coarsening
extension
InternalStream
cancellation
concretisation
extension
WriteStream
extension
ReadWriteStream

Much Extracted mation

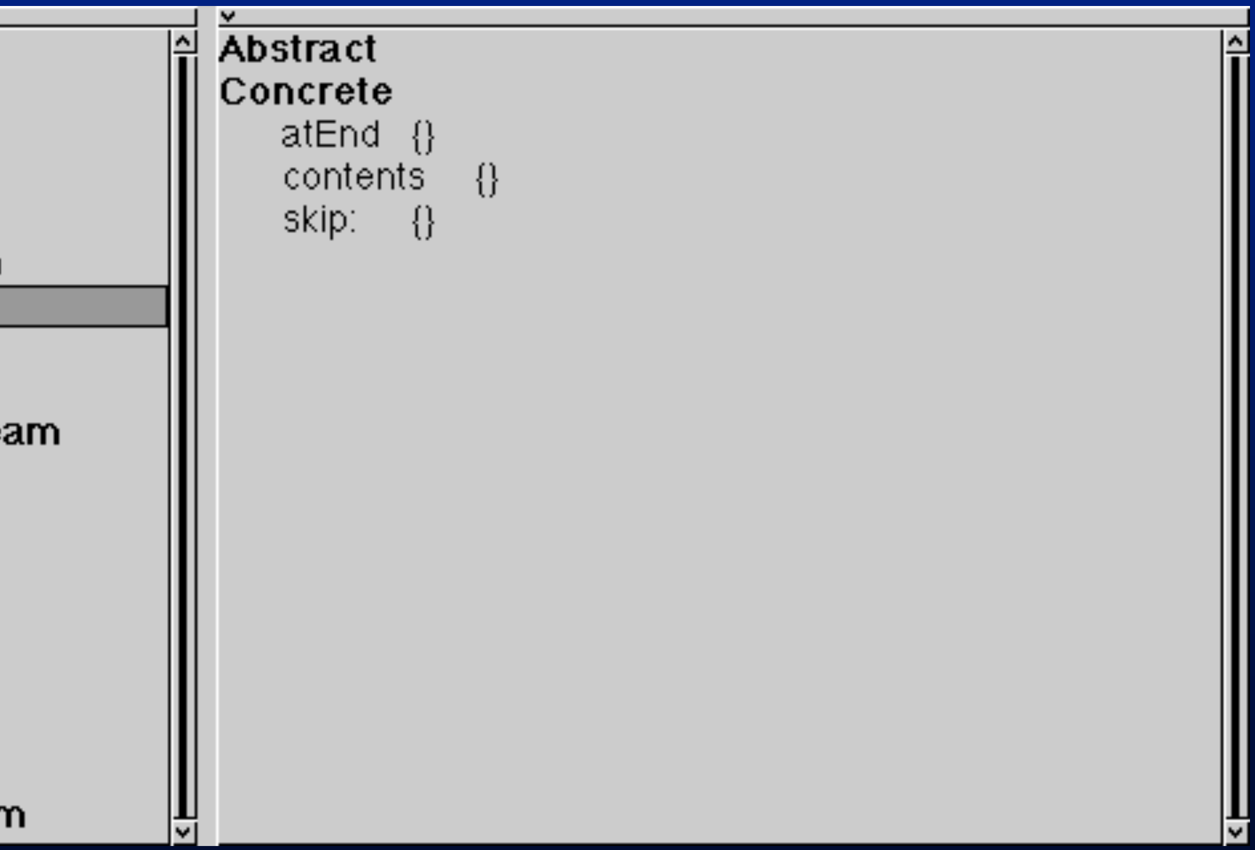
- extractor does not know which ods are important
- action with a developer is required ip implementation details

ecting Extracted
nsions



```
Abstract
    skip:     {}
Concrete
    fileIn     {close nextChunk skipSeparators peekFor: atEnd
    nextChunk {class skipSeparators peekFor: next}
    peek      {next skip: atEnd}
    peekFor:    {next skip: atEnd}
    skipSeparators{class skip: next}
    skipUpTo:   {next skip: atEnd}
```

ecting Extracted
retisations



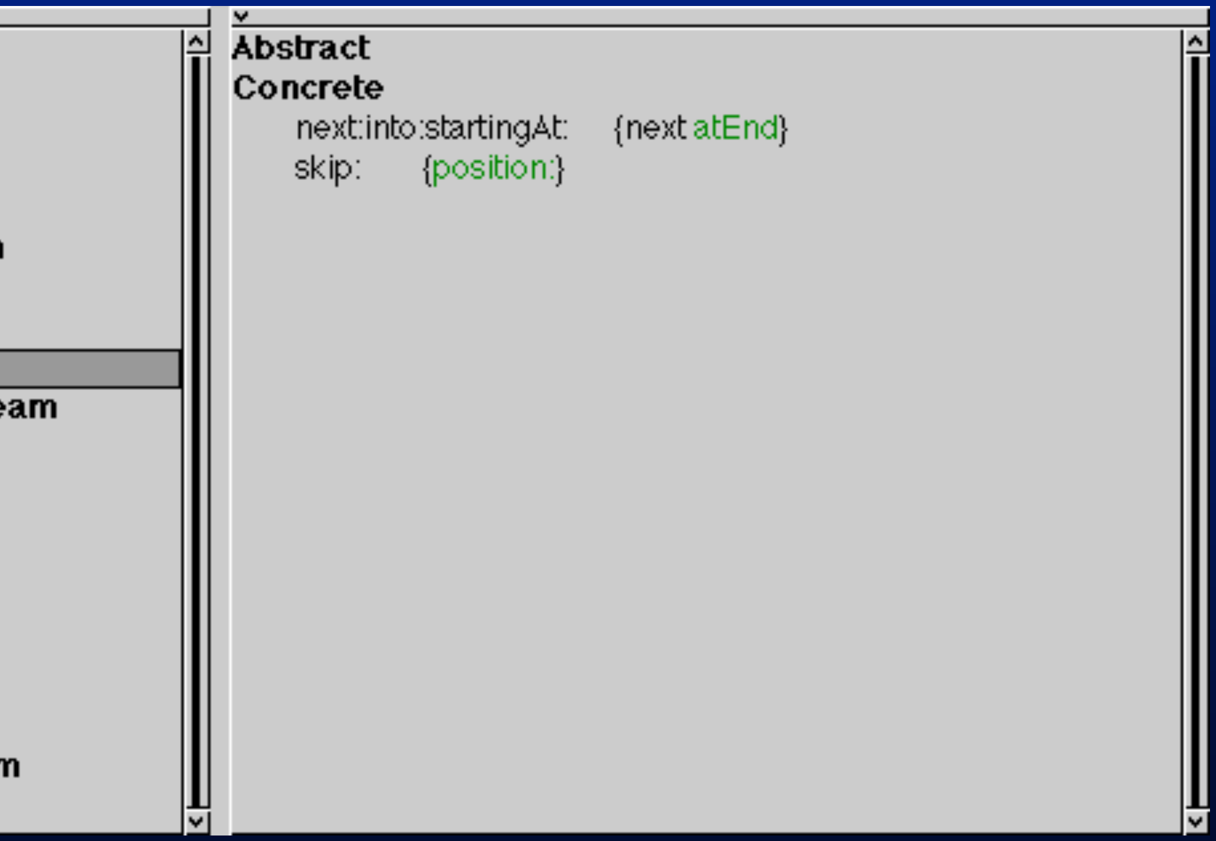
am

m

Abstract
Concrete

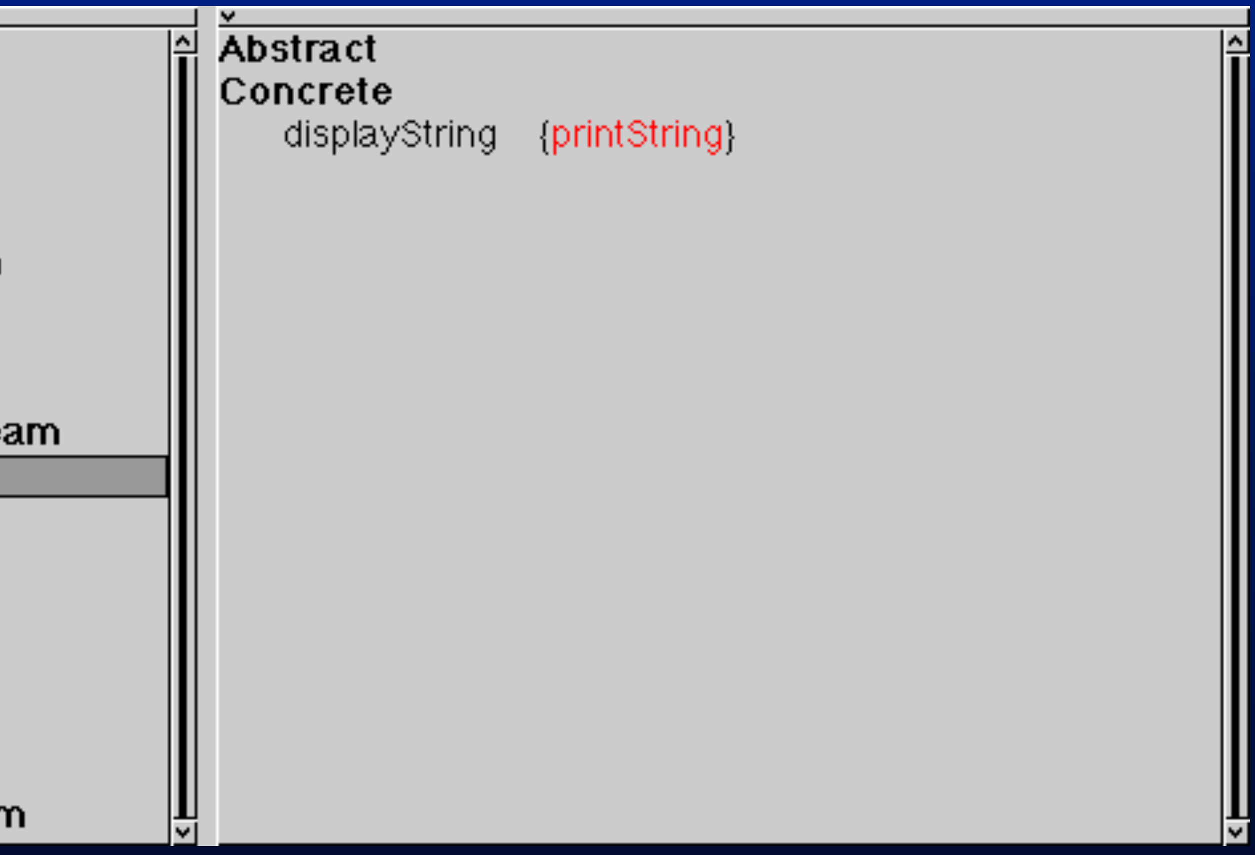
```
    atEnd  {}
    contents    {}
    skip:    {}
```

ecting Extracted ements



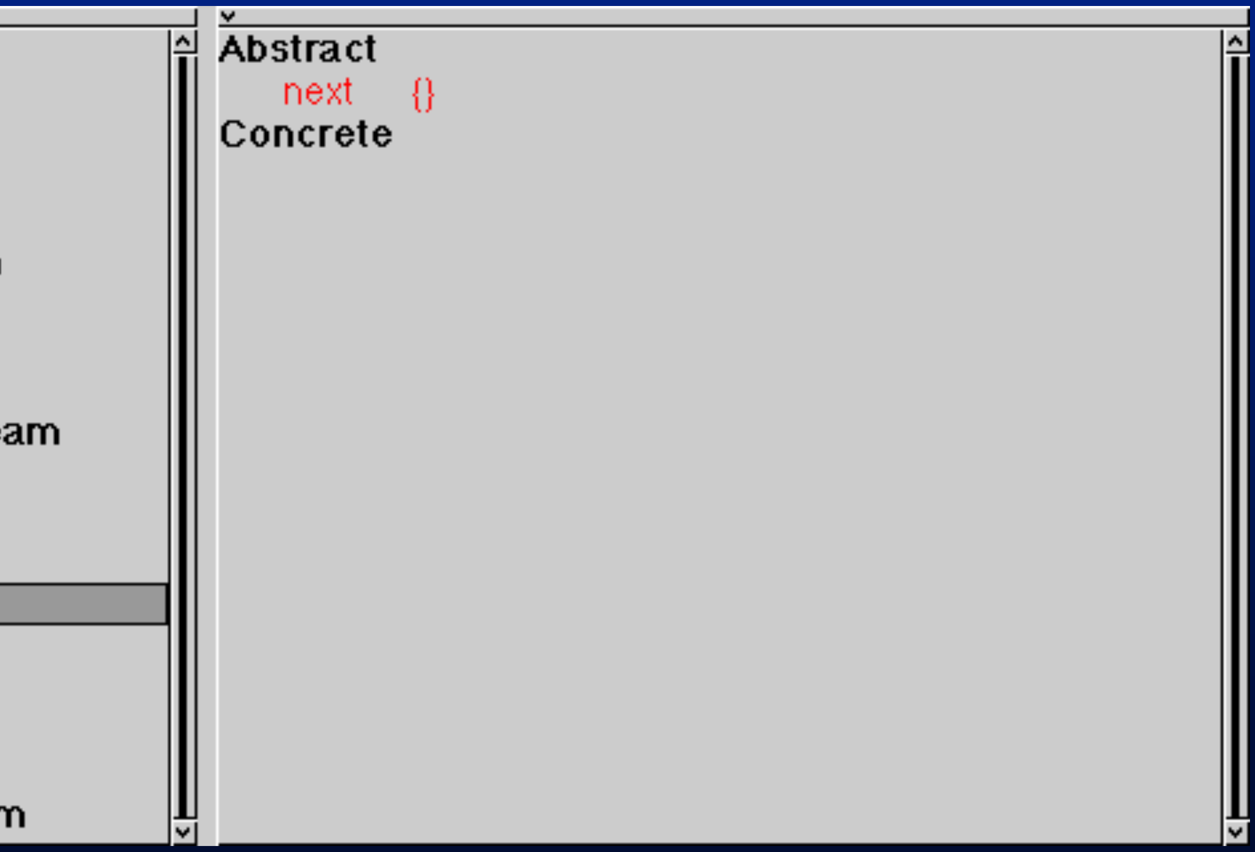
```
Abstract
Concrete
    next:into:startingAt:    {next atEnd}
    skip:    {position:}
```

ecting Extracted senings



Abstract
Concrete
displayString {printString}

ecting Extracted ellations



am

m

Abstract

next {}

Concrete

ting Bad Designs in a s Hierarchy

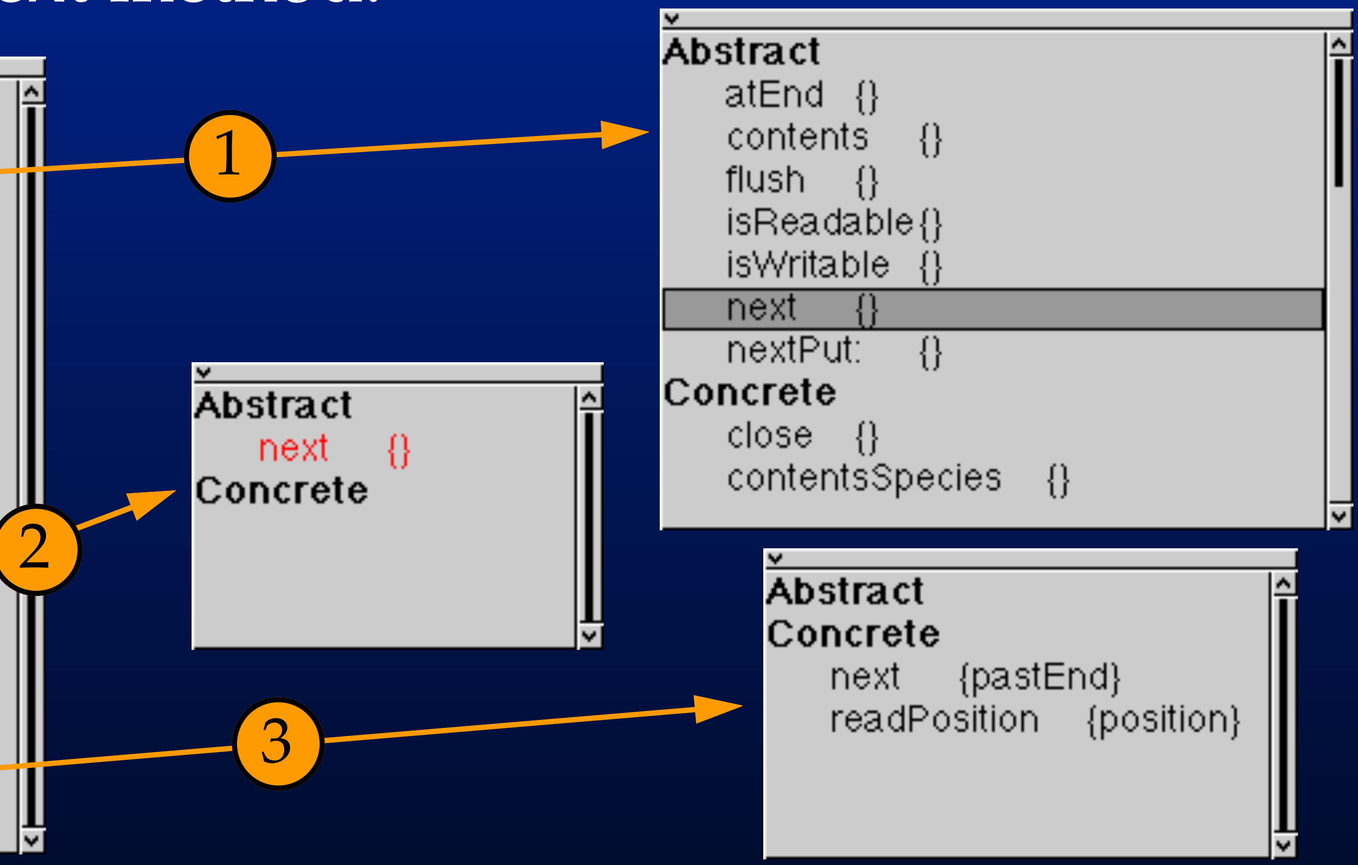
for design breaching reuse ators

y indicate methods that do not respect design as laid down by a superclass

nine what happens with the ted methods in reuse operators that pplied later on

ting Bad Designs: Example

hierarchy is awkward
ext method.



Abstract
    atEnd {}
    contents {}
    flush {}
    isReadable {}
    isWritable {}
    next {}
    nextPut: {}
Concrete
    close {}
    contentsSpecies {}

Abstract
    next {}
Concrete

Abstract
Concrete
    next {pastEnd}
    readPosition {position}

act of Bad Coding Style

coding style is troublesome for the
ctor

only super send, bad super send, ...

has driven us to make qualitative
sment of source code possible

nalysis tool is integrated in our
ser

view

lems with reuse

lems with evolution

: are reuse contracts?

e contracts at work

nining class hierarchies based on
e contracts

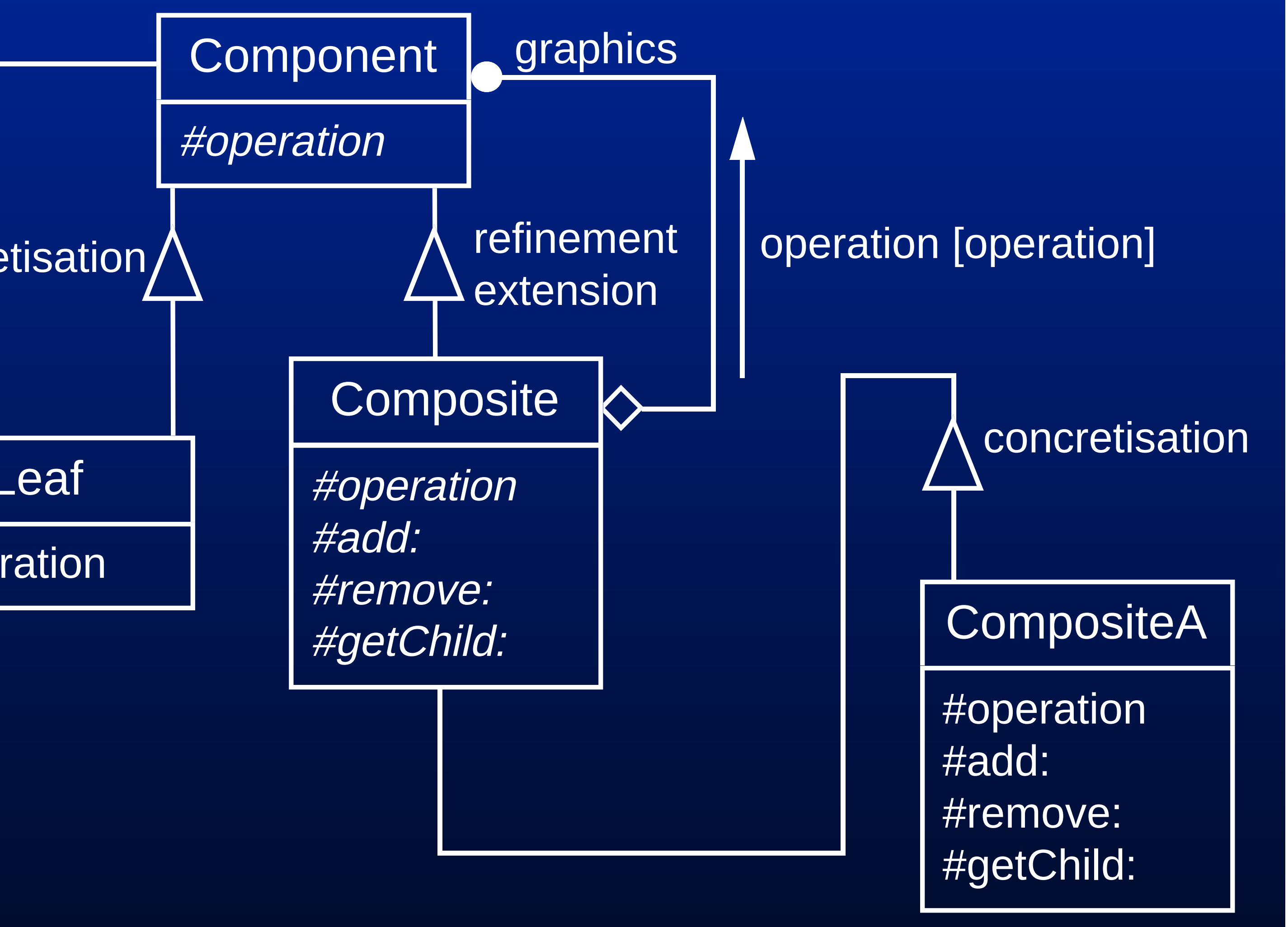
e contract research

cises: introduction to the browser

e Contract Research

- ontracts have been applied to
  - s (inheritance)
  - f interacting classes/components
  - liagrams
- nvestigation:
  - use contracts describe design patterns?
  - c reuse contracts
  - tion of multi-class reuse contracts
  - are architectures and componentware
  - contracts in a development environment
  - documentation than interfaces and invocations

gn Pattern Example



nary: Theory

contracts formally document what r can assume about a “reusable nent”

operators formally document how ble component is actually reused

rules for change propagation automatic detection of evolution s

nary: Practice

e contracts for inheritance can be cted

mination of existing source code

derstanding the design

man input is needed to filter out plementation details

l coding style may give rise to extraction blems

view

lems with reuse

lems with evolution

t are reuse contracts?

e contracts at work

nining class hierarchies based on
e contracts

e contract research

cises: introduction to the browser

Browser for the Exercises

made fully-functional browser

sed of reusable “browser part

ents” built with ApplFLab

ily be

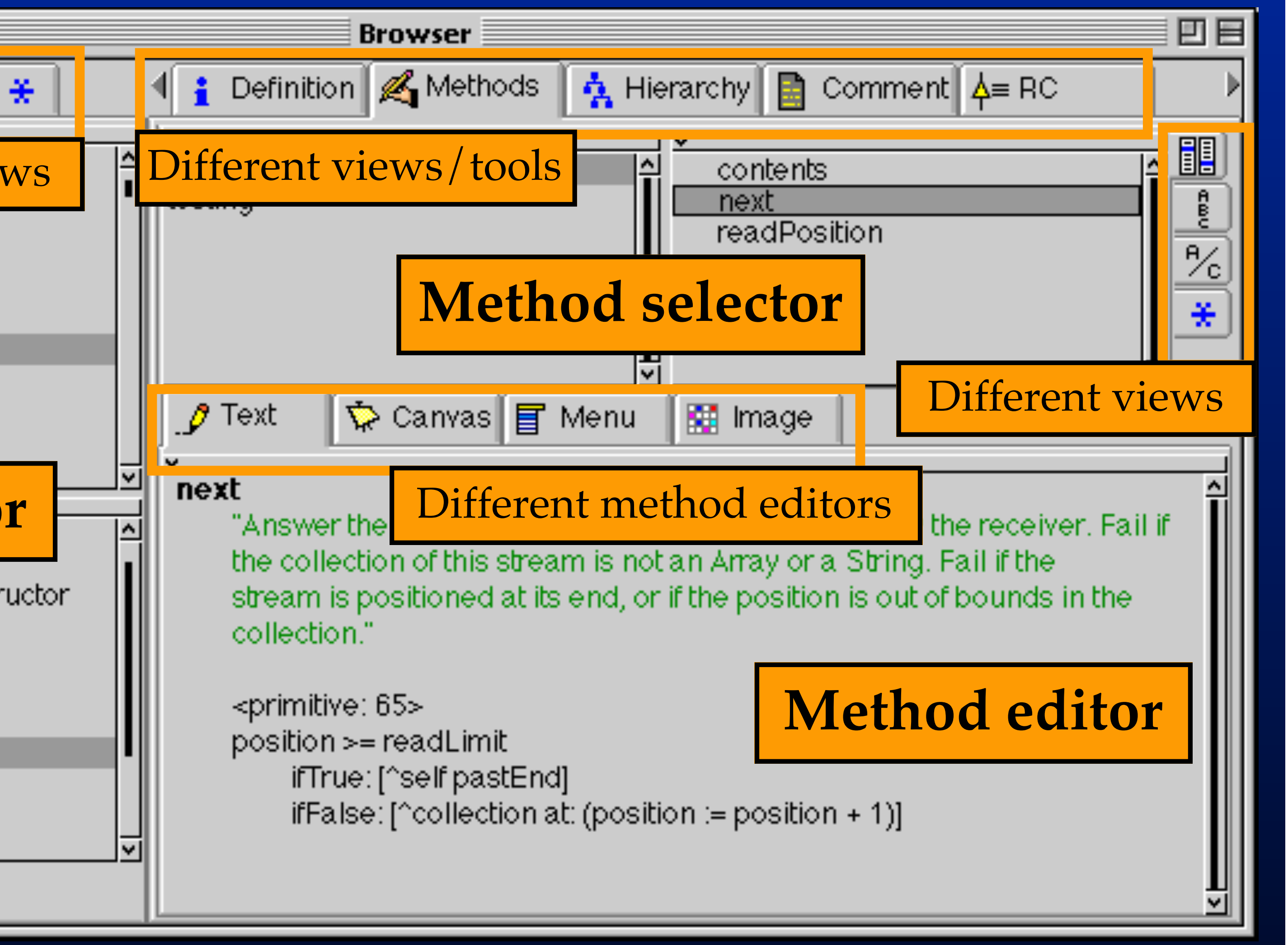
d with other

iew/editor

ents”

See ESUG’96 Summer School
“ApplFLab: Custom-made user
interface components in VisualWorks”

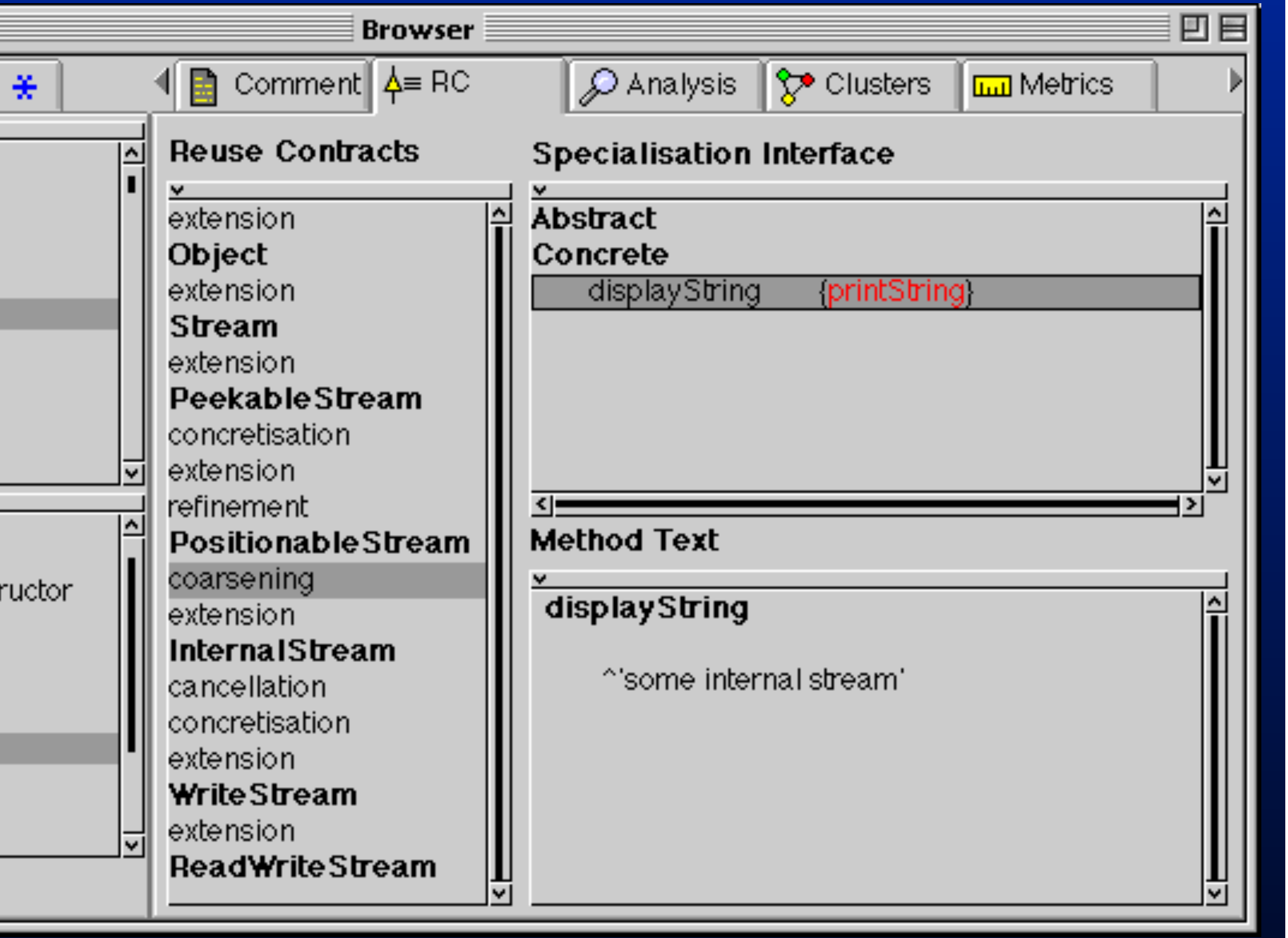
nced Browser — General



```
next
    "Answer the next object in the receiver. Fail if
    the collection of this stream is not an Array or a String. Fail if the
    stream is positioned at its end, or if the position is out of bounds in the
    collection."

    <primitive: 65>
    position >= readLimit
        ifTrue: [^self pastEnd]
        ifFalse: [^collection at: (position := position + 1)]
```

...ser — Reuse Contracts



Reuse Contracts

extension
Object
extension
Stream
extension
PeekableStream
concretisation
extension
refinement
PositionableStream
coarsening
extension
InternalStream
cancellation
concretisation
extension
WriteStream
extension
ReadWriteStream

Specialisation Interface

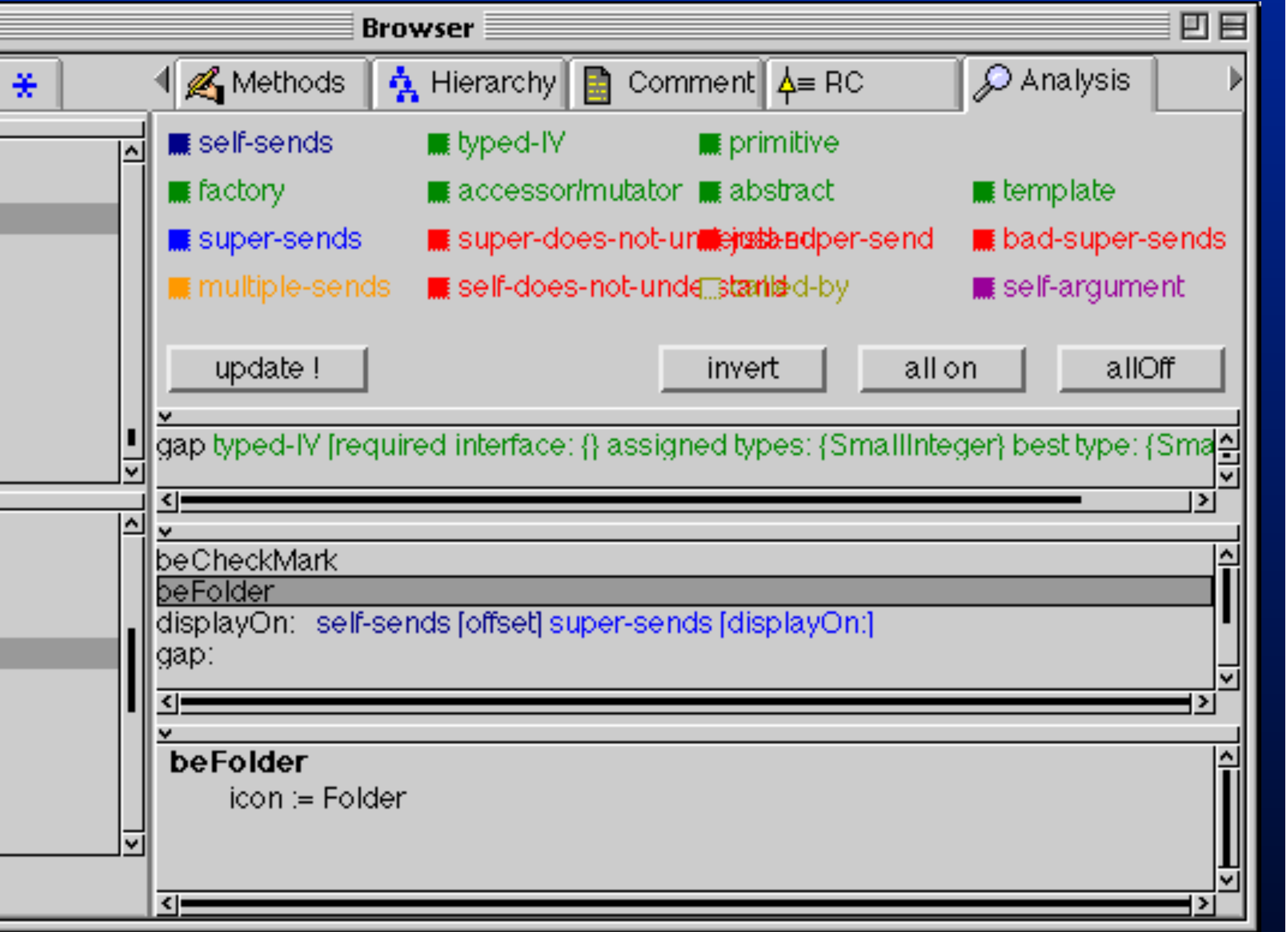
Abstract
Concrete
displayString {printString}

Method Text

```
displayString

    ^'some internal stream'
```

ser — Code Analysis

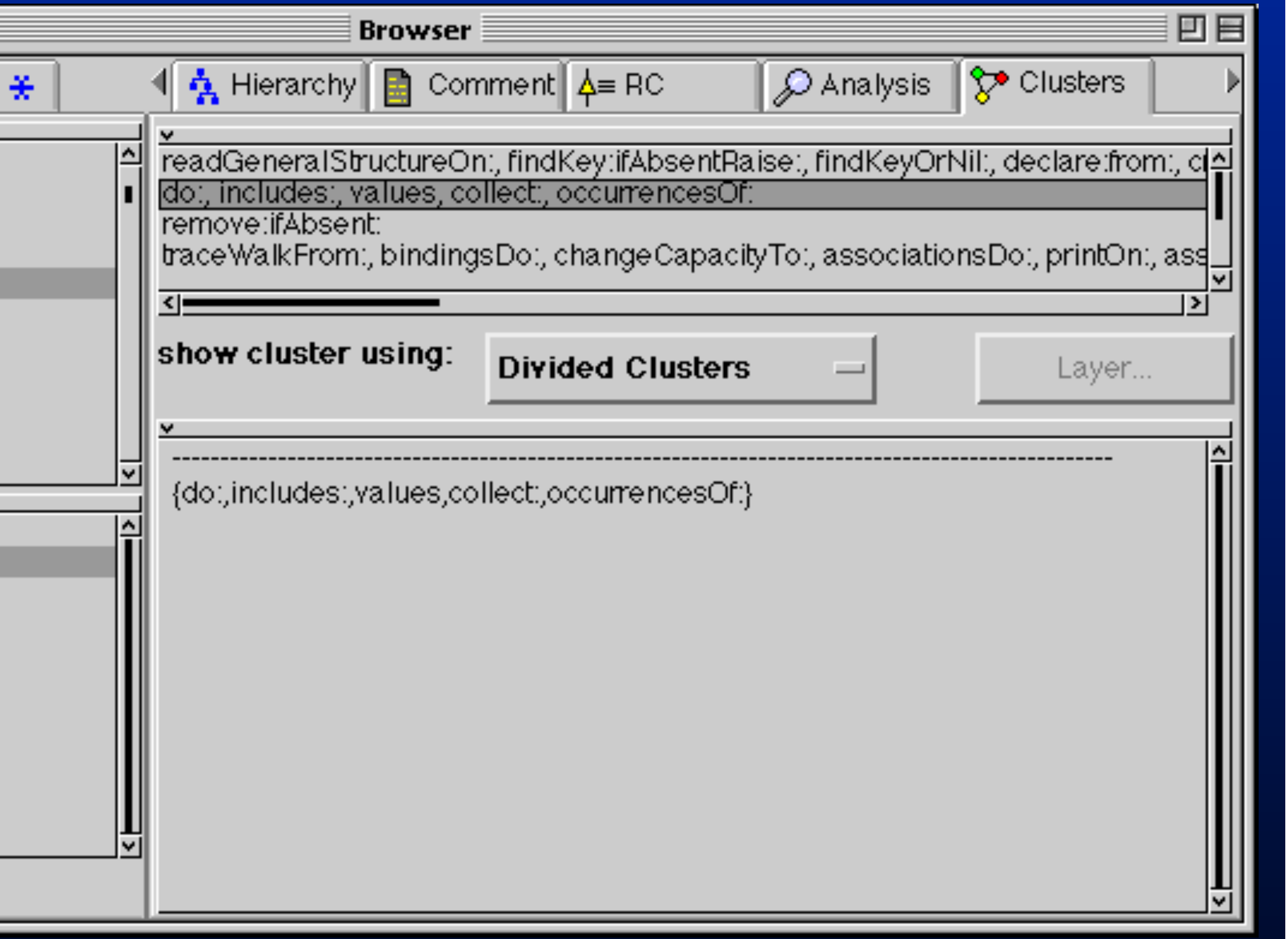


gap typed-IV [required interface: {} assigned types: {SmallInteger} best type: {Sma

beCheckMark
beFolder
displayOn: self-sends [offset] super-sends [displayOn:]
gap:

```
beFolder
	icon := Folder
```

ser — Clusters

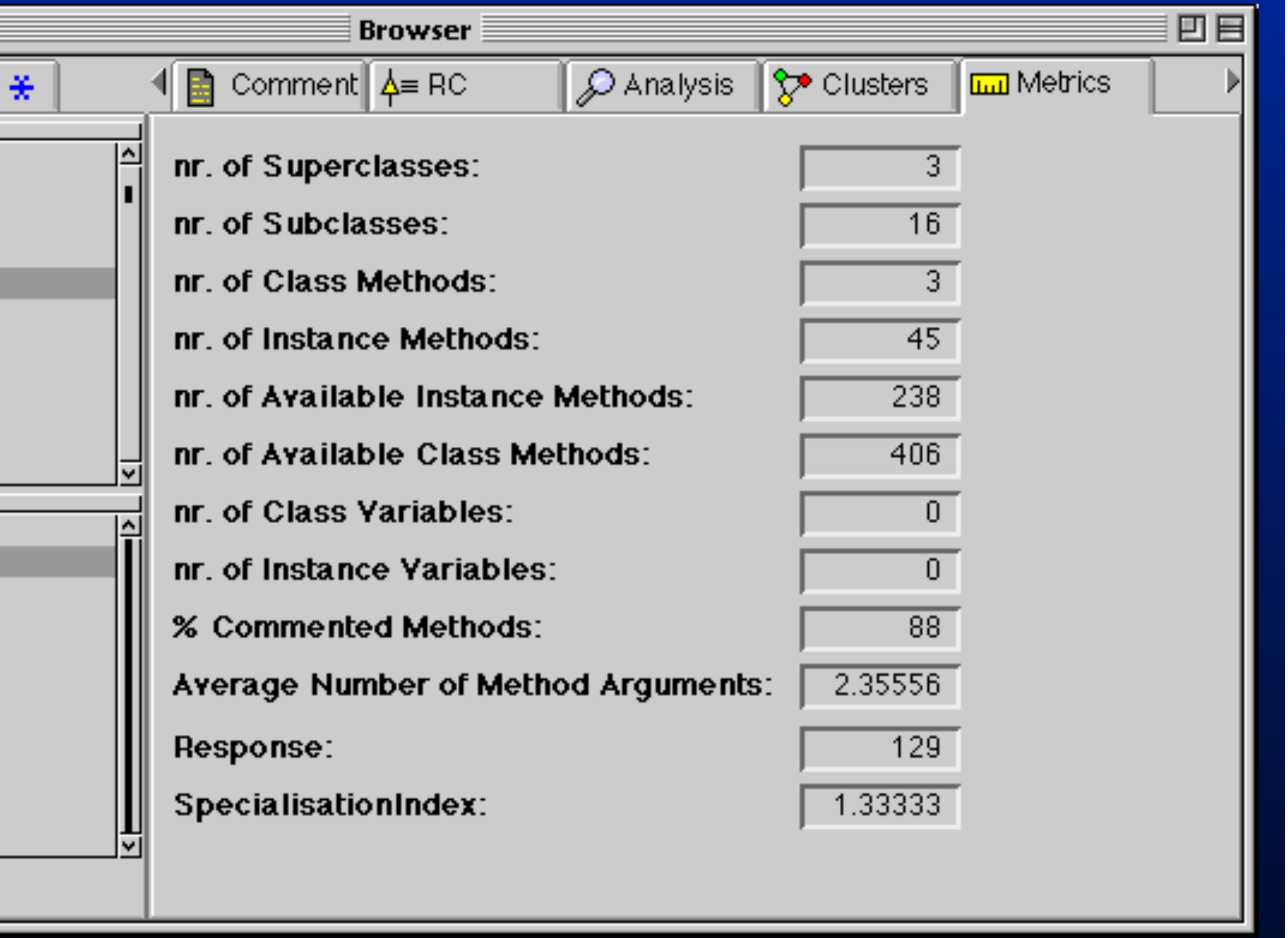


readGeneralStructureOn:, findKey:ifAbsentRaise:, findKeyOrNil:, declare:from:, c
do:, includes:, values, collect:, occurrencesOf:
remove:ifAbsent:
traceWalkFrom:, bindingsDo:, changeCapacityTo:, associationsDo:, printOn:, ass

show cluster using: Divided Clusters

```
--------------------------------------------------------------------------------
{do:,includes:,values,collect:,occurrencesOf:}
```

ser — Metrics



nr. of Superclasses:	3
nr. of Subclasses:	16
nr. of Class Methods:	3
nr. of Instance Methods:	45
nr. of Available Instance Methods:	238
nr. of Available Class Methods:	406
nr. of Class Variables:	0
nr. of Instance Variables:	0
% Commented Methods:	88
Average Number of Method Arguments:	2.35556
Response:	129
SpecialisationIndex:	1.33333

cises

he enhanced browser to
stigate Smalltalk code

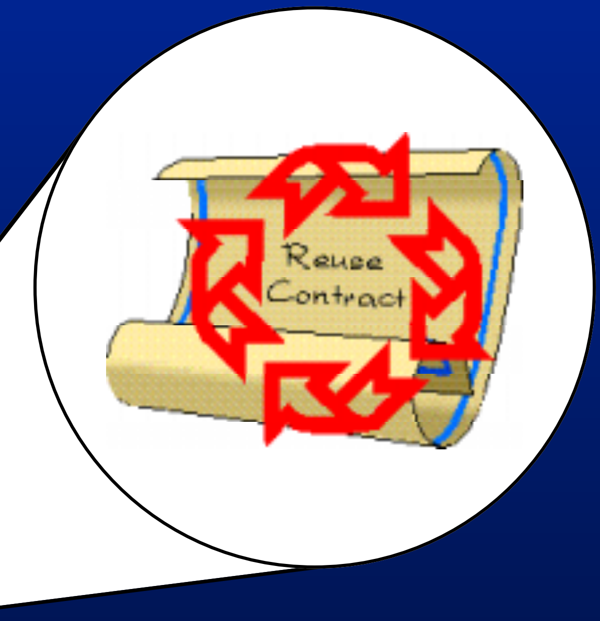
mine class hierarchies based on
racted reuse contracts

alyse the code to find methods that
der reuse

plore the different tools

n your own Smalltalk classes/
eworks

o-date Information



p://progwww.vub.ac.be/prog/pools/rcs/