

6.170 Lecture 13 More on Reasoning



John Guttag
MIT EECS

Plan for Today and Tomorrow

Revisit

- Invariants
- Reasoning about uses of data abstraction
- Exposing the representation

Reasoning about subtypes

Reasoning about code

© Michael Ernst/John Guttag

Spring 2001

Slide 2

Invariants

A predicate that is “always” true

More precisely always true in certain contexts

Rep invariant

Property of implementation

True when not executing an operation of type

R must hold only on entry and exit of ADT ops

Need not hold in the middle, or for private methods

Usually does not hold in the middle

E.g. ?

Abstract invariant

Property of specification

Used to understand client programs

© Michael Ernst/John Guttag

Spring 2001

Slide 3

(Not) Exposing the Rep

Tricky to define formally

Won't try and do it here

Important to understand motivation

Source of many hard to find errors

Want to ensure that

Correctness of implementation is a local property

Can use induction to reason about implementation

Correctness of clients does not depend up rep

Can use induction to reason about clients

If you can't do this

You may have exposed the rep

Which is in very poor taste

© Michael Ernst/John Guttag

Spring 2001

Slide 4

An Example

Class AorBSet {
Mutable sets of AorB objects (also mutable)

```
public AorBSet ()
  effects: makes a new, empty, AorBSet
public insert(AorB e)
  modifies this
  effects: this' = this U {e}
public boolean member(AorB e)
  returns: e ∈ this
public AorB choose()
  requires: this is not empty
  returns: e s.t. e ∈ this
public toVector()
  returns: a vector containing
  the elements in this
```

```
public insert(AorB e) {
  elts.addElement(e);
}
public insert(AorB e) {
  AorB ee;
  ee = new AorB(e.val);
  elts.addElement(ee);
}
public toVector() {
  return elts;
}
```

© Michael Ernst/John Guttag

Spring 2001

Slide 5

An Example

Class AorBSet {
Mutable sets of AorB objects (also mutable)

```
public AorBSet ()
  effects: makes a new, empty, AorBSet
public insert(AorB e)
  modifies this
  effects: this' = this U {e}
public boolean member(AorB e)
  returns: e ∈ this
public AorB choose()
  requires: this is not empty
  returns: e s.t. e ∈ this
public toVector()
  returns: a vector containing
  the elements in this
```

```
public insert(AorB e) {
  elts.addElement(e);
  e.setToA();
}
```

© Michael Ernst/John Guttag

Spring 2001

Slide 6

6.170 Spring 2001

An Example

```

Class AorBSet {
  Mutable sets of AorB objects (also mutable)

  public AorBset ( )
    effects: makes a new, empty, AorBSet
  public insert(AorB e)
    modifies this
    effects: this' = this U {e}
  public boolean member(AorB e)
    returns: e ∈ this
  public AorB choose ( )
    requires: this is not empty
    returns: e s.t. e ∈ this
  public toVector ( )
    returns: a vector containing
           the elements in this

```

```

public insert(AorB e) {
  elts.addElement (e);
}

public choose ( ) {
  return
  elts.elementAt(0);
}

```

© Michael Ernst/John Guttag Spring 2001 Slide 7

An Example

```

Class AorBSet {
  Mutable sets of AorB objects
  public AorBset ( )
    effects: makes a new, empty, AorBSet
  public insert(AorB e)
    modifies this
    effects: this' =
           this U {e1 s.t. e1.val = e.val}
  ...
  public AorB choose ( )
    requires: this is not empty
    returns: e s.t. e ∈ this
  public boolean hasA ( )
    returns: true iff this contains e
           s.t. e.val = A

```

```

public insert(AorB e) {
  elts.addElement (e);
}

public insert(AorB e) {
  AorB ee;
  ee = new AorB(e.val);
  elts.addElement (ee);
}

```

© Michael Ernst/John Guttag Spring 2001 Slide 8

An Example

```

Class AorBSet {
  Mutable sets of AorB objects
  public AorBset ( )
    effects: makes a new, empty, AorBSet
  public insert(AorB e)
    modifies this
    effects: this' =
           this U {e1 s.t. e1.val = e.val}
  ...
  public AorB choose ( )
    requires: this is not empty
    returns: e s.t. e ∈ this
  public boolean hasA ( )
    returns: true iff this contains e
           s.t. e.val = A

```

```

ab.setToA;
s.insert(ab);
ab.setToB;
if (s.hasA ( ))
  print Yes
else print No

```

© Michael Ernst/John Guttag Spring 2001 Slide 9

An Example

```

Class AorBSet {
  Mutable sets of AorB objects
  public AorBset ( )
    effects: makes a new, empty, AorBSet
  public insert(AorB e)
    modifies this
    effects: this' =
           this U {e1 s.t. e1.val = e.val
                  & e1 is a new object}
  ...
  public AorB choose ( )
    requires: this is not empty
    returns: e s.t. e ∈ this
  public boolean hasA ( )
    returns: true iff this contains e
           s.t. e.val = A

```

```

public insert(AorB e) {
  elts.addElement (e);
}

public insert(AorB e) {
  AorB ee;
  ee = new AorB(e.val);
  elts.addElement (ee);
}

```

© Michael Ernst/John Guttag Spring 2001 Slide 10

Abstraction Functions and Subtyping

Assume T1 a subtype (and subclass) of T2
 Sometimes abstraction functions the same
 I.e., extra parts of rep ignored
 They supply redundant information
 Consider, for example IntSet

```

IntSet
  ↓
IntSetWithExtraOps

```

© Michael Ernst/John Guttag Spring 2001 Slide 11

Abstraction Functions and Subtyping

When might AF_T1 be different?
 When specification overview changes
 Not necessarily, but often
 When changes to rep impact the abstract value
 How about ColoredWindow

```

ColoredWindow
  ↙     ↘
RedWindow  BlueWindow

```

© Michael Ernst/John Guttag Spring 2001 Slide 12

6.170 Rep Invariants & Subtyping

Assume T1 a subtype (and subclass) of T2

Is T1's rep invariant
Weaker than T2's?
The same?
Stronger?
Incomparable?

Risky to make it weaker or incomparable
Inherited code might break

Same or stronger OK

© Michael Ernst/John Guttag Spring 2001 Slide 13

6.170 Checking the Rep Invariant of a Subtype

Suppose it is the same
If no direct access to rep of supertype
No way for subtype to break invariant
But ops should still call method that checks invariant
If access to part of rep of supertype
Must proceed in usual way

Suppose it is stronger
Don't place constraints on parts of rep used by supertype
Can't count on inherited code to maintain them
OK to constrain parts of rep introduced in subtype

© Michael Ernst/John Guttag Spring 2001 Slide 14

6.170 An Example

```

Class Date {
  Overview: Immutable dates
  public Date (String d) throws notADate
  returns: new date representing
  mm/dd/yyyy
  throws: notADate if d is not a valid date
  in mm/dd/yyyy form
  public String unParse( )
  returns: a string representation
  in mm/dd/yyyy form of the date
  public String dayOfWeek ( )
  returns: a string representation
  of the day of the week (e.g., Sunday)
  of the date
}
    
```

Rep: an integer, dt

Inv: dt >= 0

Abst fcn: dt is the number of days between 1900 and the date. E.g., if the date is 1/1/1900, dt = 0.

© Michael Ernst/John Guttag Spring 2001 Slide 15

6.170 Example, cont.

```

Class Date {
  Overview: Immutable dates
  public Date (String d) throws notADate
  returns: new date representing
  mm/dd/yyyy
  throws: notADate if d is not a valid date
  in mm/dd/yyyy form public
  String unParse( )
  returns: a string representation
  in mm/dd/yyyy form of the date
  public String dayOfWeek ( )
  returns: a string representation
  of the day of the week (e.g., Sunday)
  of the date
}
    
```

Rep: an integer, dt
string, day

Inv:
dt >= 0
& day is the day returned by dayOfWeek

Abst fcn: dt is the number of days between 1900 and the date. E.g., if the date is 1/1/1900, dt = 0.

© Michael Ernst/John Guttag Spring 2001 Slide 16

6.170 Adding an Observer

Suppose I want the number of days between dates
Could write a procedure outside of type to do that
Highly inefficient

Does adding such a method change the type?
Yes, because specification changes

Can I make the new type a subtype of the old one?
Yes, because programs that use old type will work

Which of rep. invariant, and abst. fcn must change?
None

© Michael Ernst/John Guttag Spring 2001 Slide 17

6.170 Changing a Method

Suppose I want a new kind of date, "DK"
Represents an unknown date

```

public Date (String d) throws notADate
returns: if d = "DK" a new date representing don't know
else a new date representing mm/dd/yyyy
throws: notADate if d is not "DK" or a valid
date in mm/dd/yyyy form

public String dayOfWeek ( )
returns: a string representation
of the day of the week (e.g., Sunday) of the date,
or "DK" if the date is unknown
    
```

Will this be a subtype of Date?

© Michael Ernst/John Guttag Spring 2001 Slide 18

6.170 A Slightly Different Example

```
public Date (String d) throws notADate
  returns: new date representing mm/dd/yyyy
  throws: notADate if d is not a valid date in mm/dd/yyyy form
public unkDate ()
  returns: a new date representing an unknown date
```

Would this be a subtype of Date? Supertype? Neither?

```
//requires: d != null
//returns: true
boolean test(Date d) {
  String w = d.dayOfWeek();
  boolean ok = w.equals("Monday") || w.equals("Tuesday")
  || ... || w.equals("Sunday");
  return ok;}

```

© Michael Ernst/John Guttag Spring 2001 Slide 19

6.170 Summarizing

Abstraction function and rep invariant tools
For designing rep
For reasoning about code
Make reasoning modular
No need to consider effect of one op on another
Documentation

Preservation of rep invariant
R holds before and after each op of the ADT
Inductive argument works
If no rep exposure
Java can't enforce this, i.e., it's your job

Can also use induction to prove abstract invariants
Again, watch out for rep exposure

© Michael Ernst/John Guttag Spring 2001 Slide 20

6.170 Summarizing, cont.

When is rep exposed
When modular reasoning broken

When subclassing used to implement subtyping
Subtyping a property of specifications
Subclassing a mechanism for implementing subtypes
Must again be wary of destroying modularity

© Michael Ernst/John Guttag Spring 2001 Slide 21