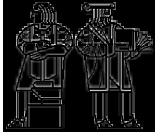


6.170 Lecture 15 Subtyping and Reasoning



John Guttag
MIT EECS

Reasoning About Subtypes

Recall that if T1 is a subtype of T2

T1's specification is stronger than T2's
Programs that work with T2 will also work with T1
Implementations of T1 are implementations of T2

What does this imply about

Rep invariants?
Abstraction functions?

Two distinct cases

Subtype doesn't correspond to subclass
Nothing interesting to say
Subtype corresponds to subclass
Fair amount to say

Abstraction Function and Subtyping

Assume T1 a subtype (and subclass) of T2
Sometimes abstraction functions the same
I.e., extra parts of rep ignored

Consider, for example

```
graph TD
  IntSet --> IntSetWithExtraOps
```

When might AF_T1 be different?
When additions to rep impact the abstract value

Consider, for example

```
graph TD
  Window --> ColoredWindow
  Window --> ShapedWindow
```

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

Stronger by placing constraints on new variables, not
stronger constraints on variables of supertype

Checking the Rep Invariant of a Subtype

Suppose it is the same

If no 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

An Example

```
Class Date {
  Overview: Immutable dates
  public Date (String d) throws notADate
  effects
  if d a valid date in mm/dd/yyyy form
  returns new date representing that
  else throws notADate
  public String unParse( )
  effects returns a string representation
  in mm/dd/yyyy form of the date
  public String dayOfWeek ( )
  effects 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.

6.170 Spring 2000

6.170 Example, cont.

```
Class Date {
  Overview: Immutable dates
  public Date (String d) throws notADate
  effects
    if d a valid date in mm/dd/yyyy form
    returns new date representing that
    else throws notADate
  public String unParse( )
  effects returns a string representation
  in mm/dd/yyyy form of the date
  public String dayOfWeek ( )
  effects 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.

© John Chapin/John Guttag Spring 2000 Slide 7

6.170 Adding A Method

Suppose I want to check the days between dates
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

© John Chapin/John Guttag Spring 2000 Slide 8

6.170 Changing a Method

```
public Date (String d) throws notADate
effects
  if d a valid date in mm/dd/yyyy form
  returns new date representing that
  else if d = "DK" returns a new date
  representing don't know
  else throws notADate
public String dayOfWeek ( )
effects returns a string representation
of the day of the week (e.g., Sunday)
of the date, or "DK" if the date is unknown
```

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

Will this be a subtype of Date?

© John Chapin/John Guttag Spring 2000 Slide 9

6.170 Adding Another Method

```
public NewDate unkDate ( )
effects returns a new date whose value is DK
public String dayOfWeek ( )
effects returns a string representation
of the day of the week (e.g., Sunday)
of the date, or "DK" if the date is unknown
```

Is either of these a subtype of the other?
The waters here are deep and treacherous

Values of Date a subset of values of NewDate
But ...

© John Chapin/John Guttag Spring 2000 Slide 10

6.170 Subtype and Subclassing

Subtyping related to specifications and abstractions
Subclassing an implementation mechanism
Valuable for getting re-use of code

When designing from scratch
Usually possible to gracefully combine two notions

```
      Date
     /  \
NormalDate DateWithDK
```

As programs evolve tensions often arise

© John Chapin/John Guttag Spring 2000 Slide 11

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

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

© John Chapin/John Guttag Spring 2000 Slide 12



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