Object models of code

Daniel Jackson

November 7, 2007
topics for today

heap semantics of Java
• assignment, field set, aliasing
• reachability and garbage collection

object models of code
• snapshots as heap graphs
• boxes as classes and interfaces, relations as fields

example: EXIF filtering
heap semantics of Java
pop quiz

what happens when this code is executed?

```java
String s = "hello";
s.concat("world");
System.out.println (s);
s = s.concat(" world");
System.out.println (s);
StringBuffer sb = new StringBuffer ("hello");
sb.append(" world");
System.out.println (sb);
StringBuffer sb2 = sb;
sb2.append ("!");
System.out.println (sb);
```
solutions

what you needed to know to answer correctly

immutable and mutable types

- `String` is immutable, `StringBuffer` is mutable
- method call on immutable object can’t affect it

assignment semantics

- the statement `x = e` makes `x` point to the object that `e` evaluates to

aliasing

- the statement `x = y` makes `x` point to the same object as `y`
- subsequent mutations of the object are seen equivalently through `x` and `y`
- since immutable objects can’t be mutated, sharing is not observable
how mutation happens

through field setting

- statement `x.f = y` makes `f` field of `x` point to object `y`

through array update

- statement `a[i] = y` makes `element_i` ‘field’ of `a` point to object `y`
null and primitives

primitive values

- eg, integers, booleans, chars
- are immutable (and aren’t objects)
- so whether shared is not observable

null

- is a value of object type
- but does not denote an object
- cannot call method on null, or get/set field
the operator ==

- returns true when its arguments denote the same object (or both evaluate to null)

for mutable objects
- if \( x == y \) is false, objects \( x \) and \( y \) are observably different
- mutation through \( x \) is not visible through \( y \)

for immutable objects
- if \( x == y \) is false, objects \( x \) and \( y \) might not be observably different
- in that case, can replace \( x \) by \( y \) and save space (called ‘interning’)
- Java does this with \texttt{Strings}, with unpredictable results
- lesson: don’t use \( == \) on immutables (unless you’re doing your own interning)
heap reachability

an assignment or field set can leave an object unreachable

from example before

• after these statements

    String s = "hello";
    s = s.concat(" world");

• string object “hello” is unreachable

once an object is unreachable

• it cannot be reached again

• so removing it will not be observable

garbage collector (aka “automatic memory management”)

• marks unreachable objects, then deallocates them
conceptual leaks

storage leak
· use of memory grows, but active state isn’t growing

no storage leaks in garbage-collected language?
· unfortunately, can still happen

exercise: what’s wrong with this code?

```java
public class ArraySet {
    private Object [] elements;
    private int size;
    ...
    public void delete (Object o) {
        for (int i = 0; i < size; i++) {
            if (elements[i].equals(o)) {
                elements[i] = elements[size-1];
                size--;
            }
        }
    }
}
```
object models of code
consider our immutable set represented as a list:

```java
public interface Set {...}
public class ListSet implements Set {
    private List elements;
    ...
}
public interface List {...}
public class EmptyList implements List {...}
public class NonEmptyList implements List {
    private Object element;
    private List rest;
    private int size;
    ...
}
```

what do sample states look like?

- in class exercise
a relational view of the heap

heap contains
• all objects that have been allocated
• (and not yet garbage-collected)

each field
• links an object to another object
• (unless it’s value is null or primitive)

so heap is a graph
• nodes are objects
• edges are field references
object model

can represent all possible snapshots as a code object model
multiplicities

what do multiplicities mean for code?

target multiplicity (n)

- $? \leq 1$ field $f$ can be null, $! = 1$ otherwise
- later, when we consider abstract fields, we’ll use $\ast \geq 0$ and $+ > 0$ also

source multiplicity (m)

- $? \leq 1$ A’s cannot share B objects through $f$
- $\ast \geq 0$ otherwise

exercise: add multiplicities to ListSet model
classification

what does the subset arrow mean for code?

• B extends or implements A

note

• problem OM: top-level boxes represent disjoint sets
• code OM: tedious to show Object, so interfaces often top-level (even though they don’t represent disjoint sets of objects)
example: EXIF filtering
problem object model

Photo
  tags
    Tag
      key
        Key
          key
            Basic
              Compound
                AndFilter
                OrFilter

matches

Filter
  filters

Object
  value
    Object
      range
code object model

- Photo
  - directory

- Directory
  - tags

- Tag
  - key
  - value

- Object

- Key
  - key

- Basic
- Compound

- Filter
  - elements

- Set
  - filters

Filter
  - AndFilter
  - StringFilter
  - DateFilter
  - OrFilter

String
  - val

Date
  - from, to

StringFilter
  - Date

bold elements
- show where changes were made
changes

introducing collections
\· for filter set and tag set
\· directory class is from third-party library

implementing relations with code
\· matches relation implemented as a method

reifying conceptual relations
\· range relation for filter value becomes from/to, eg
\· basic filters split into date and string, to allow specialized comparisons

see handout for full Java code
caching results

how’s performance?
• actually pretty good: 10-100ms to search through 2000 photos
• but treating each query afresh is wasteful
• very large repository will need better scheme

how to improve performance?
• cache results of previous filter queries
• if filter matches one previously used, just use old results
• if filter is contained in previous filter, can filter old results

example: date ranges
• can cache date queries for each month of the year
• then filtering on a given day or week will always start with just the month’s photos
conceptual model of caching

notes

- some subset of filters are cached, and have matches stored against them
- each of these cached filters subsumes some set of filters
code model of caching

ideas

‣ FilterTree implements subsumes
‣ TrueFilter allows more uniform coding of FilterTree
‣ FilterMap holds cached query results against filters

limitations

‣ only basic filters cached
‣ only used at top level of filter
summary

heap structure

• for side-effect-free programming, a tree is enough
• but now must view as a graph, to explain sharing
• natural representation of set of possible heaps with OM

translating from problem to code

• often fairly direct
• but have to modify to make easy to implement
• implementation ideas often lead to extensions to the conceptual model too