SEPARATION OF CONCERNS

Lecture 20

CS2110. Spring 2019

Separation of concerns

Term used by Edsger Dijkstra in discussing the development of algorithms and mathematical proofs.

Rarely discussed, but its use is crucial in some situations.

Illustrate with the development of bubble down.

```
/** class invariant –array b, integer size, HashMap map– is true,
      except that b[k] may be out of place.
      Bubble b[k] down. */
public void  bubbleDown(int k)
```

Simplicity

```
int cR= 2 * k + 2;  int cL= 2 * k + 1;
while (cL < size && cR < size + 1) {
   if (compareTo(cL, cR) == 0) { // same priority for left and right
      if (compareTo(k, cL) < 0) {
         swap(cR, k);k= cR;
         cR= 2 * k + 2;   cL= 2 * k + 1;
      } else if (compareTo(k, cL) >= 0) { return; }
   } else if (compareTo(cL, cR) > 0 || !map.containsValue(cR) && …){
         // left has higher priority, or just left exists
         if (compareTo(k, cL) < 0) {
            swap(cL, k); k= cL;
            cR= 2 * k + 2; cL= 2 * k + 1;
         } else if (compareTo(k, cL) >= 0) { return; }
   } else if (compareTo(cR, cL) > 0 || !map.containsValue(cL) && … ) {
         // right has higher priority, or just right exists
         if (compareTo(k, cR) < 0) {
            swap(cR, k); k= cR;
         cR= 2 * k + 2; cL= 2 * k + 1;
   } else if (compareTo(k, cR) >= 0) { return; }
}
```

Separation of concerns


```
// inv: class invariant is true except k may be out of place
while ( 2k+1 < size     &&   k belongs below a child ) {
    bubble down

}
```

Checking k belongs below a child requires
a lot of initialization, perhaps a helper method.
It may require knowing which child to bubble.
But that's another concern!
Try developing without it here!

Concerns:

When to stop bubbling:

1. k has no children.
2. ! k belongs below a child.

Which child to bubble with?
How to do the bubbling?

Separation of concerns


```
// inv: class invariant is true except k may be out of place
while ( 2*k+1 < size) {
    // bubble down if possible, return if not




}
```

Concerns:

When to stop bubbling:

1. k has no children.
2. ! k belongs below a child.

Which child to bubble with?

How to do the bubbling?

Separation of concerns


```
// inv: class invariant is true except k may be out of place
while ( 2*k+1 < size) {
    //Set c to the child to bubble with
    int c= 2*k+1;  // left child
    if (c+1 < size  && compareTo(c+1, c) >= 0)  c= c+1;
}
```

Concerns:

When to stop bubbling:

1. k has no children.
2. ! k belongs below a child.

Which child to bubble with?

How to do the bubbling?

Separation of concerns


```
// inv: class invariant is true except k may be out of place
while ( 2*k+1 < size) {
    //Set c to the child to bubble with
    int c= 2*k+1;  // left child
    if (c+1 < size  && compareTo(c+1, c) >= 0)  c= c+1;

    if (compareTo(k, c) >= 0)  return;


}
```

When to stop bubbling:

1. k has no children.
2. ! k belongs below a child.

Which child to bubble with?

How to do the bubbling?

Separation of concerns


```
// inv: class invariant is true except k may be out of place
while ( 2*k+1 < size) {
    //Set c to the child to bubble with
    int c= 2*k+1;  // left child
    if (c+1 < size  && compareTo(c+1, c) >= 0)  c= c+1;

    if (compareTo(k, c) >= 0)  return;
    swap(k, c);
    k= c;
}
```

When to stop bubbling:

1. k has no children.
2. ! k belongs below a child.

Which child to bubble with?

How to do the bubbling?