CS61B Midterm Review

Winston Liaw and Amir Kamil

Agenda

1. Static vs. Dynamic
2. Box and pointer diagrams
3. Code Questions and examples
4. Bit representation
5. Algorithmic Analysis
6. Access rights
7. Quickie questions

Static vs. Dynamic

- What happens? Assume we have defined:

```
Homer h = new Homer();
Bart b = new Bart();

    Homer h1 = b;
    h1.talk2();
```

Answer: Homer, Bart: dude

Static vs. Dynamic cont.

- What happens? Assume the same definitions

```
Bart b1 = b;
b1.talk2();
```

Answer: Homer, Bart: dude

```
Cartoon c1 = h;
((Homer)c1).talk4();
```

Answer: Homer, Homer: doh!

Static vs. Dynamic cont.

- What happens?

```
Cartoon c2 = b;
((Bart)c2).whoa();
```

Answer: dude

```
Lumpy l2 = b;
((Homer)l2).talk4();
```

Answer: Bart: dude

Static vs. Dynamic cont.

- For calls with an object of interest (i.e. h.f()), static methods are called based on static type, non-static methods are based on dynamic type
- For calls involving “this”, things get a little trickier. Static calls “stay” in the same class, dynamic calls are based on the dynamic set.

Box and Pointer diagrams

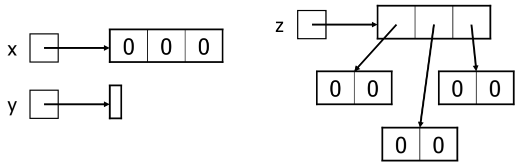
- Draw the diagrams that result from this code:

```
int [] x = new int[3];
int [] y = new int[0];
int [][] z = new int[3][2];
```

Box and Pointer diagrams

- Draw the diagrams that result from this code:

```
int [] x = new int[3];
int [] y = new int[0];
int [][] z = new int[3][2];
```



Box and Pointer diagrams cont.

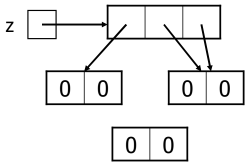
- Modify the box and pointer diagram according to this code:

```
z[1] = z[2];
```

Box and Pointer diagrams cont.

- Modify the box and pointer diagram according to this code:

```
z[1] = z[2];
```



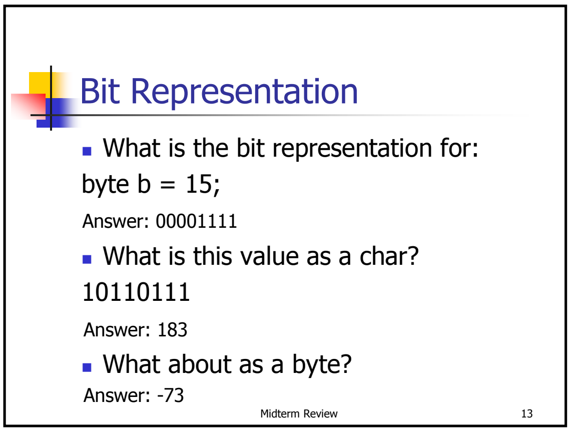
Coding Question

- Finish this method:

```
/* Given an IntList, it will reverse it
   destructively and return the new list */
public IntList reverse(IntList l) {
        …
}
```

Coding Question Solution

```
public IntList reverse(IntList l) {
    IntList prev = null;
    IntList next = l.tail;
    while (l != null) {
          next = l.tail;
          l.tail = prev;
          prev = l;
          l = next;
    }
    return prev; //once we are done reversing all the pointers
                     //we need to set l's head to the new head
}
```



Bit Representation

- What is the bit representation for:

byte b = 15;

Answer: 00001111

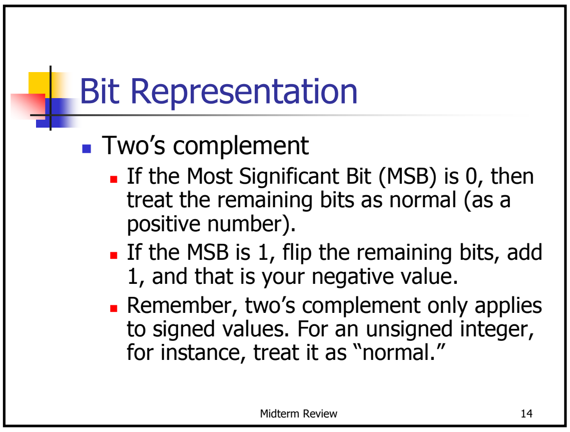
- What is this value as a char?

10110111

Answer: 183

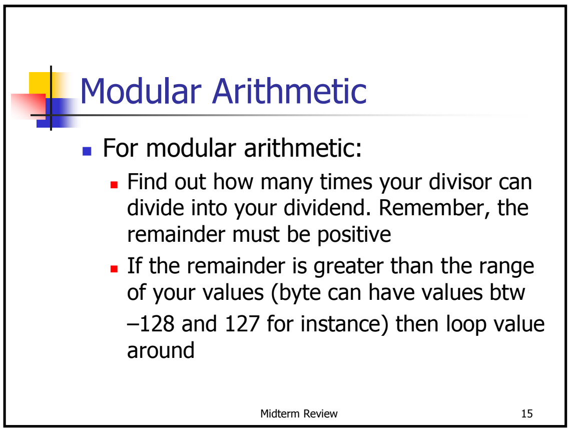
- What about as a byte?

Answer: -73



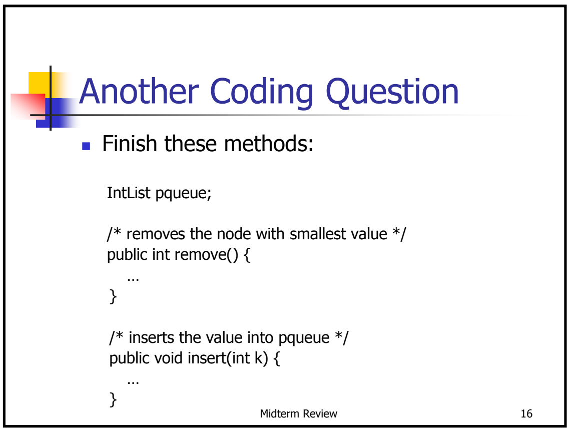
Bit Representation

- Two's complement
  - If the Most Significant Bit (MSB) is 0, then treat the remaining bits as normal (as a positive number).
  - If the MSB is 1, flip the remaining bits, add 1, and that is your negative value.
  - Remember, two's complement only applies to signed values. For an unsigned integer, for instance, treat it as "normal."



Modular Arithmetic

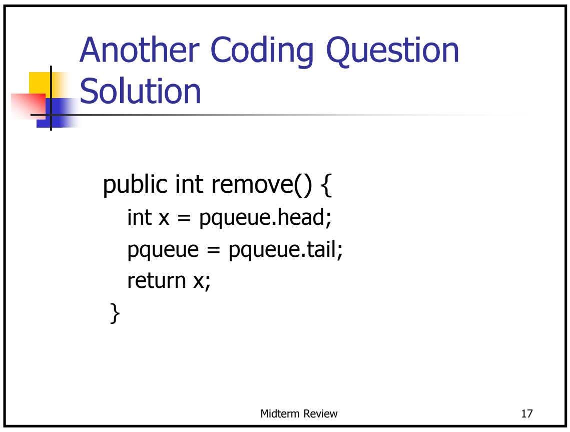
- For modular arithmetic:
  - Find out how many times your divisor can divide into your dividend. Remember, the remainder must be positive
  - If the remainder is greater than the range of your values (byte can have values btw –128 and 127 for instance) then loop value around



Another Coding Question

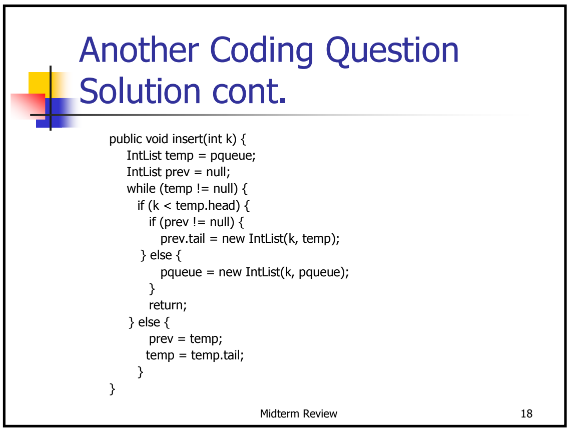
- Finish these methods:

```
IntList pqueue;

/* removes the node with smallest value */
public int remove() {
    …
}

/* inserts the value into pqueue */
public void insert(int k) {
    …
}
```



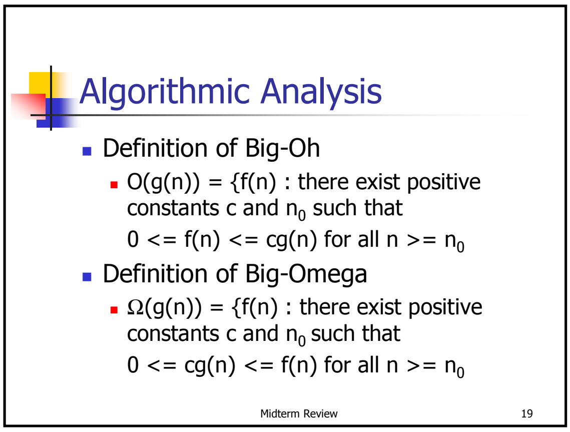
Another Coding Question Solution

```
public int remove() {
    int x = pqueue.head;
    pqueue = pqueue.tail;
    return x;
}
```



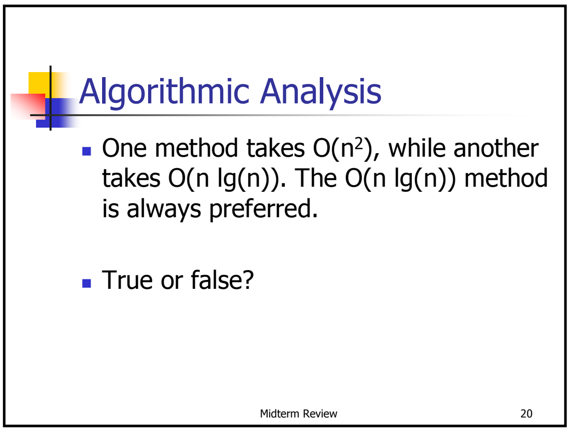
Another Coding Question Solution cont.

```
public void insert(int k) {
    IntList temp = pqueue;
    IntList prev = null;
    while (temp != null) {
      if (k < temp.head) {
        if (prev != null) {
          prev.tail = new IntList(k, temp);
       } else {
          pqueue = new IntList(k, pqueue);
        }
        return;
    } else {
        prev = temp;
       temp = temp.tail;
      }
}
```



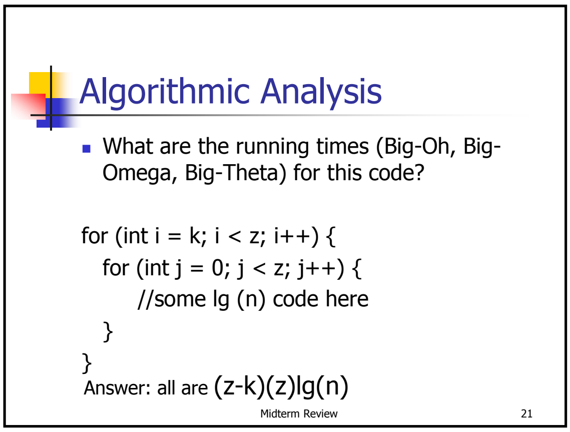
Algorithmic Analysis

- Definition of Big-Oh
  - O(g(n)) = {f(n) : there exist positive constants c and n_0 such that 0 <= f(n) <= cg(n) for all n >= n_0
- Definition of Big-Omega
  - Ω(g(n)) = {f(n) : there exist positive constants c and n_0 such that 0 <= cg(n) <= f(n) for all n >= n_0



Algorithmic Analysis

- One method takes O(n^2), while another takes O(n lg(n)). The O(n lg(n)) method is always preferred.
- True or false?

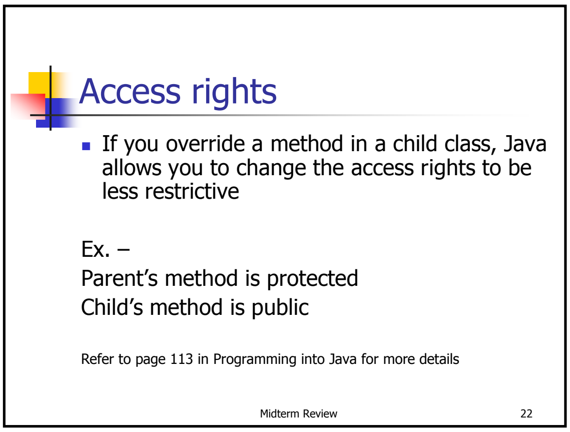


Algorithmic Analysis

- What are the running times (Big-Oh, Big-Omega, Big-Theta) for this code?

```
for (int i = k; i < z; i++) {
   for (int j = 0; j < z; j++) {
        //some lg (n) code here
   }
}
```

Answer: all are (z-k)(z)lg(n)

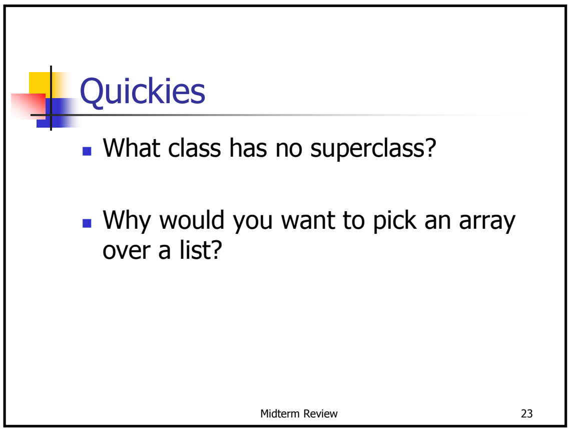


Access rights

- If you override a method in a child class, Java allows you to change the access rights to be less restrictive

Ex. –
Parent’s method is protected
Child’s method is public

Refer to page 113 in Programming into Java for more details



Quickies

- What class has no superclass?
- Why would you want to pick an array over a list?