[TAP:ARIUL] Iterator

- Which of the following is not a valid way to write an iterator class?
  A. Write a class implementing the *Iterator* interface
  [✓ B.] Write a class implementing the *Iterable* interface
  C. Write a class extending the *AbstractIterator* class
  D. They are all valid
  E. Whatever

Administrative Details

- Lab 6: PostScript is today
  - Individual lab this week
  - GitHub repositories are ready

Today's Outline

- Iterators
  - Iterator interface
  - AbstractIterator abstract class (structure5)
  - Aside: For-each and Iterable interface
  - More Iterator Examples
- Bitwise Operations

Implementation : VectorIterator

```
                  ReverseVectorIterator
public class VectorIterator<E> extends AbstractIterator<E>{
      protected Vector<E> v;
      protected int cur;

      public VectorIterator(Vector<E> v){
            this.v = v;
            ~~reset()~~; cur=v.size()-1;
      }
      public void reset(){ cur=0;}
                                                  cur>=0;
      public boolean hasNext(){ return ~~cur < v.size()~~;}
                                          cur--
      public E next(){ return v.get(~~cur++~~);}
      public E get(){ return v.get(cur);}
}
```

In Vector.java:

```
public Iterator<E> iterator() {
     return new VectorIterator<E>(this);
}
```

ReverseIterator.java

- Goal:
  - Take an iterator `it` and return its values in reverse order
- Implementation:

```
protected AbstractIterator<E> it;
public ReverseIterator(Iterator<E> iter) {
    SinglyLinkedList <E> list = new SinglyLinkedList<E>();
    while (iter.hasNext())
        list.addFirst(iter.next());
    it = (AbstractIterator<E>) list.iterator();
}
public E next(){ return it.next(); }
public boolean hasNext() { return it.hasNext(); }
```

SkipIterator.java

1 4 3 3 3

↑ ↑

- Goal:
  - Take an iterator `it` and a value `val` = 3
  - Return sequential values from `it` as long as they don't match `val`
- Implementation:

```
protected AbstractIterator<E> it;
E val;

public E next(){
    E ret = it.next();
    while (it.get().equals(value) && it.hasNext())   ] skipping "val"
        it.next();
    return ret;
}
```

```
SkipIterator(Iterator<E> iter,
                 E val){
    it=(AbstractIterator<E>)iter;
    this.val = val;
}
```

Today’s Outline

- Iterators
  - Iterator interface
  - AbstractIterator abstract class (structure5)
  - Aside: For-each and Iterable interface
  - More Iterator Examples
- Bitwise Operations

Representing Numbers

- Humans usually think of numbers in base 10
- But even though we write `int x = 23;` the computer stores `x` as a sequence of `1`s and `0`s
  - 00000000 00000000 00000000 00010111

Bitwise Operations

- We can use *bitwise* operations to manipulate the `1`s and `0`s in the binary representation
  - Bitwise 'and': `&`

    $3 \& 6 = 2$

    $$\begin{array}{r} 0\ldots0011 \\ \& \; 0\ldots0110 \\ \hline 0\ldots0010 \end{array} \leftarrow 2$$

  - Bitwise 'or': `|`

    $3 | 6 = 7$

    $$\begin{array}{r} 0\ldots0011 \\ | \; 0\ldots0110 \\ \hline 0\ldots0111 \end{array} \leftarrow 7$$

  - Bit shift left: `<<`

    $1 << 4 = 16$ $\quad 0\ldots01 \Rightarrow 0\ldots010000$

    $a << n = a \cdot 2^n$

  - Bit shift right: `>>`

    $1 >> 4 = 0$ $\quad 0\ldots01 \Rightarrow 0\ldots0$

    $a >> n = \left\lfloor \frac{a}{2^n} \right\rfloor$

[TAP] Bit-shifting

- What is 97 >> 3 ?
  - A. 94
  - [✓ B.] 12
  - C. 13
  - D. None of the above
  - E. Whatever

$0000\,1100001 \;\Big|\; \Rightarrow 000\,1100 \,\Big|\, \cancel{001}$

$\uparrow$ 12

Revisiting printInBinary()

```
public static String printInBinary(int n) {
       if (n <= 1)
           return "" + n;

       return printInBinary(~~n/2~~)+~~n%2~~;
}
```

n>>1   n&1

↑

0... 01

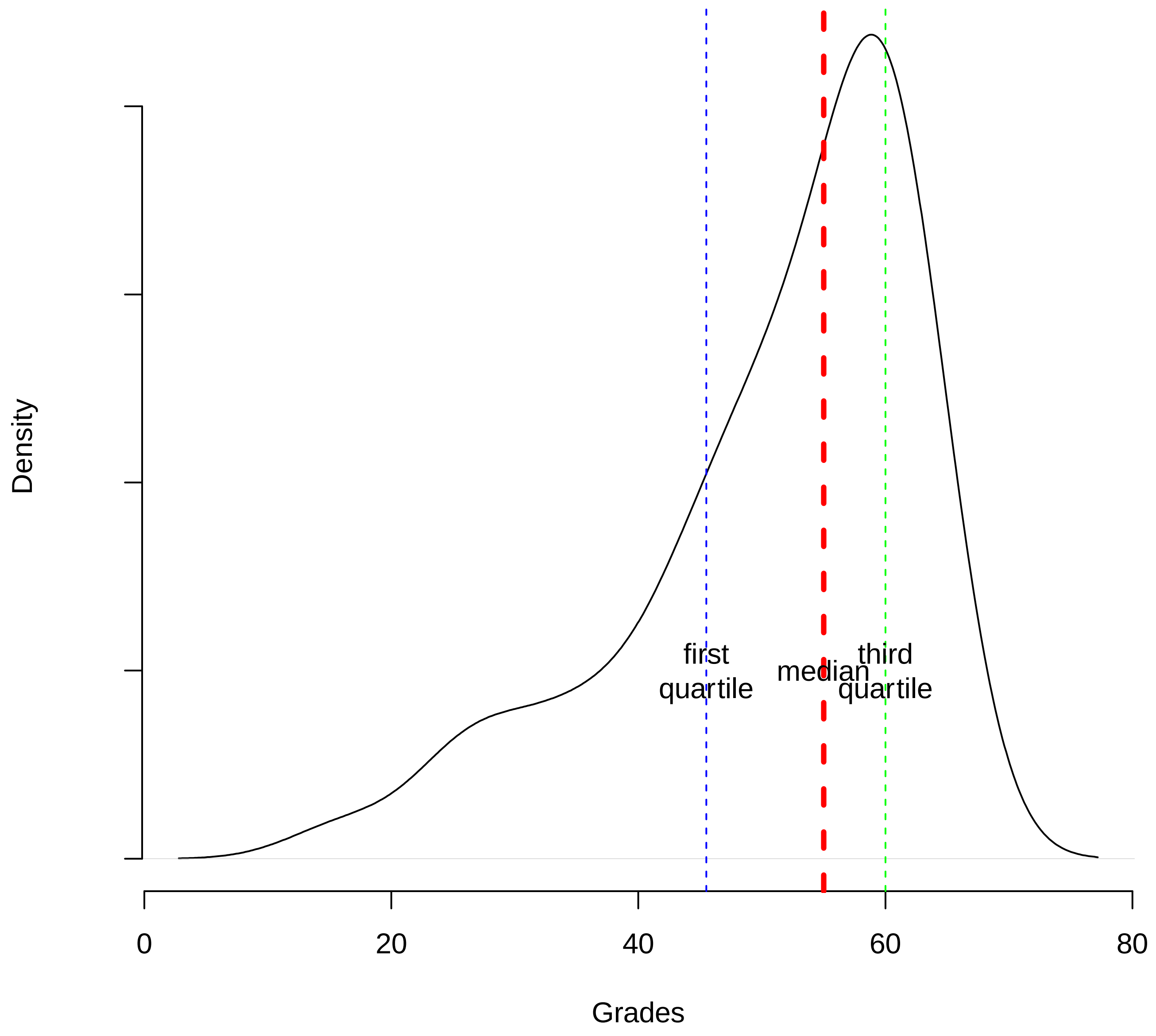
Revisiting printInBinary()

```
public static String printInBinary(int n) {
    String result = "";
    mask = 1 << 31; // since there are 32 bits
    while (mask > 0){
        if (n & mask == 1)
            result += 1;
        else
            result += 0;
        mask = mask >> 1;
    }
    return result;
}
```

Midterm Exam

- Score is out of 65 points
  - Median 55 (1st quartile: 45.5, 3rd quartile: 60)
  - Just one part of your semester grade
    - View as diagnostic: strategize for final
  - We will answer questions, and regrade if a mistake was made
  - No one who submits their work and masters the material should fail this course
    - Anyone with a “failing” midterm grade will have an opportunity to elevate to a passing midterm grade
    - We will reach out with details

Midterm Grade Density (out of 65 points)


Density
first quartile
median
third quartile
Grades