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

Reversing Vector Iterator

```java
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; }
    public boolean hasNext(){ return cur < v.size() ; }
    public E next(){      return v.get(cur++);  }
    public E get(int)    { return v.get(cur);  }
}
```

In Vector.java:

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

• Goal:
  • Take an iterator \texttt{it} and return its values in reverse order

• Implementation:

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

• Goal:
  • Take an iterator \texttt{it} and a value \texttt{val} = 3
  • Return sequential values from \texttt{it} as long as they don’t match \texttt{val}

• Implementation:

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

public E next() {
  E ret = it.next();
  while (it.get().equals(val) \&\& it.hasNext())
    it.next();
  return ret;
}
```
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 1s and 0s
  - `00000000 00000000 00000000 00010111`
Bitwise Operations

- We can use *bitwise* operations to manipulate the 1s and 0s in the binary representation
  - Bitwise ‘and’: &
    
    \[
    3 \& 6 = 2
    \]
  
  - Bitwise ‘or’: |
    
    \[
    3 \mid 6 = 7
    \]
  
  - Bit shift left: <<
    
    \[
    a \ll n = a \cdot 2^n
    \]
    
    \[
    1 \ll 4 = 16
    \]
  
  - Bit shift right: >>
    
    \[
    a \gg n = \left\lfloor \frac{a}{2^n} \right\rfloor
    \]
    
    \[
    1 \gg 4 = 0
    \]
[TAP] Bit-shifting

• What is 97 >> 3 ?
  A. 94
  B. 12
  C. 13
  D. None of the above
  E. Whatever
public static String printInBinary(int n) {
    if (n <= 1)
        return "" + n;
    return printInBinary(n/2)+n%2;
}

...
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)