(US)ACM Working Group on Algorithmic Transparency and Accountability

“...[discussing] the growing impact of algorithmic decision-making on our society...”

or, in other words, why you should be concerned about computer algorithms deciding...
• whether your resume is selected or trashed
• what websites should be displayed in your search results
• how much you should pay for car insurance
• how likely you are to commit a crime
• ...etc

read the statement
http://www.acm.org/public-policy/algorithmic-panel

watch the panel
https://www.youtube.com/watch?v=DDW-nM8idgg&feature=youtu.be
Today’s big ideas

• An iterator is an object that returns one element at a time. It can return elements from a data structure or any sequence we want.

• By making Iterator an interface, we can write code that uses an iterator but doesn’t need to know what data structure the elements came from.

• It is helpful to identify an invariant when implementing an iterator.
why you should care about Iterators

class ListPrinter {
    public static void print(LinkedList ls) {
        for (int i=0; i<ls.size(); i++) {
            System.out.print(ls.get(i));
            System.out.print(",");
        }
    }
}

If N is the size of the linked list, give the tightest upper bound on running time in big-Oh.

a) \(O(1)\)
b) \(O(N)\)
c) \(O(N\log N)\)
d) \(O(N^2)\)
e) \(O(N^3)\)

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
**Iterator**

*iterator*: An object that provides elements one at a time until there are no more

```java
public interface Iterator {
    // return the next element or throw error if there is no next element
    int next();

    // return true if the Iterator has more elements
    boolean hasNext();
}
```
A search engine sends me 15 search results then I scroll to the bottom of the page...

Can think of the search results being provided by an Iterator. Each call to next() returns 15 more results from the list of thousands of results.
Example: List iterator

list is [1, 5, 7, 8, 14]

Iterator iter = list.iterator()
iter.hasNext() => true
iter.next() => 1
iter.hasNext() => true
iter.next() => 5
iter.hasNext() => true
iter.next() => 7
iter.hasNext() => true
iter.next() => 8
iter.hasNext() => true
iter.next() => 14
iter.hasNext() => false
Peer instruction

list is [1,5,8,14]

Suppose that Evens, is an Iterator that returns the even data of the given list in order. What are the return values of the last 5 lines of code?

a) true, 8, true, 14, false
b) 1, 5, 8, 14, null
c) true, 8, true, 14, ERROR
d) false, null, false, null, true
e) false, 1, false, 5, true
f) true, 1, true, 5, true
g) 8, 14, null, null, null
h) 8, 14, ERROR
Iterators provide element-at-a-time access to “streams”

Example: The stream of prime numbers

primes = [2,3,5,7,11,13,17,...]

infinite sequence, so we cannot store it in a data structure, but we can pretend...

just calculate the next prime when someone asks for it
What does this code print?

```java
public class Streams2 {
    public static class PeerInstruction implements Iterator<Integer> {
        private int z;
        private int y;
        public PeerInstruction() {
            z = 0;
            y = 1;
        }
        public boolean hasNext() {
            return true;
        }
        public Integer next() {
            int t = z;
            z = y;
            y = y + t;
            return t;
        }
    }
    public static void main(String[] args) {
        Iterator<Integer> mystery = new PeerInstruction();
        while (mystery.hasNext()) {
            System.out.print(mystery.next()); System.out.print);
        }
    }
}
```

a) 1,1,2,3,5,8,13,... ... means never ends
b) 0,1,1,2,3,5
c) 0,1,2,3,4,5,6,...
d) 0,1,1,2,3,5,8,13,...
e) 1,2,3,4,5,6

https://b.socrative.com/login/student/ room CS2230X ids 1000-2999 room CS2230Y ids 3000+
why you should care about Iterators
Part II

class ListPrinter {
    public static void print(LinkedList ls) {
        Iterator iter = ls.iterator();
        while (iter.hasNext()) {
            System.out.print(iter.next());
        }
    }
}

N is the size of the linked list. Assuming a reasonable implementation of the iterator, which is the tightest upper bound of running time of ListPrinter.print?

a) \( O(1) \)
b) \( O(N) \)
c) \( O(N\log N) \)
d) \( O(N^2) \)
e) \( O(N^3) \)

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Return of *invariants*

*invariant*: an expression that is always true
(something that does not vary is invariant)

What non-trivial invariants are there in the following code?

```java
int sum = 0;
for (int i=0; i<5; i++) {
    sum += i;
}
```

Invariants are going to be useful when implementing Iterators
Implementation of an iterator for LinkedList

```java
class LinkedList {
    private ListNode header;
    public LinkedList() {
        header = new ListNode(0);
    }
    public Iterator iterator() {
        return new LinkedListIterator(header.next);
    }
    private class LinkedListIterator implements Iterator {
        private ListNode current;
        public LinkedListIterator(ListNode first) {
            this.current = first;
        }
        public boolean hasNext() {
            return this.current != null;
        }
        public Object next() {
            if (!hasNext()) throw new NoSuchElementException();
            int result = (Integer) current.data;
            current = current.next;
            return result;
        }
    }
}
```

What is the invariant for a LinkedListIterator? (should be in terms of current)

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
RepeatsIterator returns x copies of each element x
for list [3,1,2] the iterator will return 3,3,3,1,2,2

```java
class RepeatsIterator implements Iterator {
    private Iterator fromIter;
    private int count;
    private int currentElement;
    public RepeatsIterator(Iterator from) {
        fromIter = from;
        count = 0;
        currentElement = 0; // okay because 0 is never returned
        findNext();
    }
    private void findNext() {
        while (count == currentElement && fromIter.hasNext()) {
            currentElement = (Integer) fromIter.next();
            count = 0;
        }
    }
    public boolean hasNext() {
        return count < currentElement;
    }
    public Object next() {
        if (!hasNext()) throw new NoSuchElementException();
        int result = currentElement;
        count++;
        // invariant broken here
        findNext(); // findNext() fixes the invariant
        return result;
    }
}
```

What is the invariant for a RepeatsIterator before and after (but not necessarily during) each call to next()?
(should be in terms of fromIter, count, currentElement)
Peer instruction

GreaterThanOrIterator will return only the elements of list whose value is greater than K

```java
private class GreaterThanOrIterator implements Iterator<Integer> {
    private int nextEle; // index of next element to return
    private ArrayList<Integer> list; // list being iterated over
    private final int K;
}
```

Which of these options would be the most useful invariant?

a) `list.get(nextEle) > K || nextEle >= list.size()`  
b) `list.get(nextEle) <= K && nextEle < list.size()`  
c) `nextEle == 0`  
d) `nextEle > K`  
e) `nextEle < list.size()`  
f) `nextEle >= list.size()`
Building part of a database management system

“How many flights did SkyWest Airlines have in 2015?”

Answer: 45001

The key: Iterators do all the work of processing the data!
Query processed by a chain of iterators

```
Your job:
• build various iterators
• use them to answer some queries on real data
```

```
TextFileReader("flights.csv")
```

```
Filter(airline == "SkyWest" and year == 2015)
```

```
Count
```

```
while (iter.hasNext()) {
  print(iter.next());
}
```

```
SkyWest, 2015, 4:45pm
SkyWest, 2010, 12:02pm
Delta, 2015, 11:20pm
SkyWest, 2015, 4:45pm
1
```
Two deadlines for HW5

• Oct 11: set up your git repository and update PROGRESS_REPORT.txt
• Oct 16: full assignment submission
Today’s big ideas

• An **iterator** is an object that returns one element at a time. It can return elements from a data structure or any sequence we want.

• By making Iterator an **interface**, we can write code that uses an iterator but doesn’t need to know what data structure the elements came from.

• It is helpful to identify an invariant when implementing an iterator.