Iterators

Instructors: Sam McCauley and Dan Barowy

April 10, 2022

Admin

- Midterm back today (after iterators)
- Any questions?

Iterators

Traversing a data structure

- Let’s say I want to print all of the positive elements of a sequence of integers

- How would I do this?

Traversing a Vector

```
public static void printPositive(Vector<Integer> vec) {
   for(int i = 0; i < vec.size(); i++) {
      if(vec.get(i) > 0) {
         System.out.println(vec.get(i));
      }
   }
}
```

Pretty straightforward for a vector. Can we generalize this so that it works for any sequence of items?

- Sure! We're only using `List` operations in the above

Traversing a List

```
public static void printPositive(List<Integer> l) {
   for(int i = 0; i < l.size(); i++) {
      if(l.get(i) > 0) {
         System.out.println(vec.get(i));
      }
   }
}
```

This does work fine. What's a downside of this?

- What's the running time if `l` is a linked list?
- $O(n^2)$! (Each `get` call is $O(n)$)
- Can we fix this? Of course.

Traversing a Linked List

```
public static void printPositive(SinglyLinkedList<Integer> l) {
   Node<Integer> current = l.head;
   while(current != null) {
      int value = current.value();
      if(value > 0) {
         System.out.println(value);
      }
      current = current.next();
   }
}
```

Does this work?

- No; not as-is. `head` is `protected`.
- We could use inheritance to create a `TraversableSinglyLinkedList` that supports this somehow...seems annnoying

The problem

- Traversing a linear data structure is a fundamental operation that we'll want to do all the time
- Each new data structure needs to be traversed in a new way
- That's terrible in terms of the goal of OOP!

Iterator

- A unified way to traverse java data structures
- Goal: want to be able to create a class whose job it is to traverse a particular data structure
- Use an Interface to group these classes together. In other words: an interface for classes that traverse data structures
- What methods do we want such a class type to have?
  - `next()`: gets the next item in the data structure
  - `hasNext()`: checks to see if the next item exists
- Sounds familiar? You've used iterators a number of times already in this class...
- `Scanner`, `Reader` (in this lab)

Purpose of Iterator

- We can pass around an iterator itself (the same way we can pass around a `Scanner`)
- It has the ability to get us the next element from the data structure—and to determine if such an element is available.
- The iterator knows how to traverse the data structure
- Many data structures have an `iterator()` method that returns a basic iterator to traverse the list
- Let's rewrite our method so that it works with any iterator

Traversing Using an Iterator

```
public static void printPositive(Iterator<Integer> it) {
   while(it.hasNext()) {
      int value = it.next();
      if(value > 0) {
         System.out.println(value);
      }
   }
}
```

How can we use this?

```
   //let's say vec is a Vector<Integer>
   printPositive(vec.iterator());

    //let's say sll is a SinglyLinkedList<Integer>
   printPositive(sll.iterator());
```

Trying out some code

- First: let’s try out actually using an iterator
- Goal: sum all the items in list of integers
- Let’s do it using a while loop and using a for loop

Defining the Iterator⟨E⟩ interface

- Need to `import java.util.Iterator`
- Methods:
  - `boolean hasNext()`
  - `E next()`
  - `remove()`
    - *Use carefully* (or not at all)
    - By default just throws an error

Abstract Iterator

- A `structure5` abstract class to fill in some iterator pieces
- Gives a useful `reset()` method

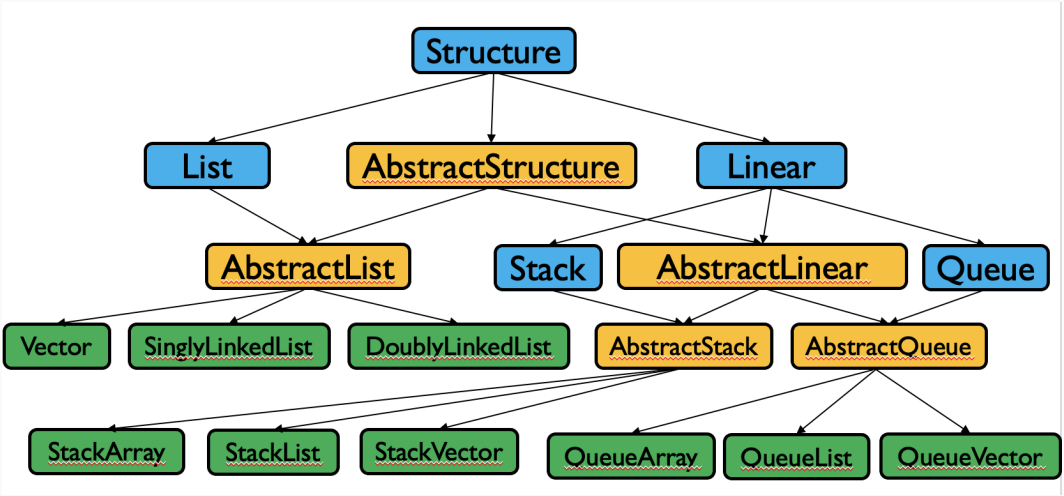
Let’s make a couple practice iterators

- First: iterator to traverse from the tail to the head of a `DoublyLinkedList`
- Can we make an iterator that takes *another* iterator as an argument, and gives its elements in reverse order? (OK if destroys original iterator)
  - Would need to store them
  - What data structure would be best to use?

Iterable Interface

- Most of the data structures we've seen have a built-in `iterator()` method
- Very handy—we shouldn't have to make our own class to iterate over a `Vector`; people do that all the time
- We can imagine writing code that works on *any* data structure with such a method (in fact, we already did so, but we required it was a `List`
- `Iterable<T>` is an interface
  - Built-in; no import needed
- One required method: `Iterator<T> iterator()`

Iterable data structures



`Structure<E>` extends `Iterable<E>`, so all of these are iterable

(Arrays are too)

for each loops

```
int[] grades = { 100, 78, 92, 87, 89, 90 };
int sum = 0;
for (int g : grades)
   sum += g;
```

- For-each loops work using iterators!
- Can do the above with any `Iterable` data structure

for each loops

```
Stack<String> strStack = new StackList<String>();
strStack.push("0");
strStack.push("1");
strStack.push("2");
strStack.push("3");
strStack.push("4");
for (String s : strStack)
   System.out.println(s);
```

Let's look at the `StackList` code that's used here.

Care about iterators

- Like for-each loops, iterators work best on data structures that are not changing

- Be very careful about changing the data structure while iterating over it!

Generalizing iterators

- Bear in mind: *anything* with a `hasNext()` and `next()` method is an iterator
- It's often used for iterating over a data structure, but doesn't have to be
- (If we have time): Let's make an iterator that prints the Fibonacci numbers

Midterm Comments

Reminder: resubmissions

- You have two resubmissions in this course
- Can be used for any lab, or for the midterm
- Basic idea: correct any mistakes you made and you can get points back
- Formal requirements in handout on website (soon), and on paper right now. Also giving you an example of what a good resubmission looks like
- Due at the *end of the semester*.
- We designed the midterm knowing that you can do this. If you got (for example) a 70, you can get 15 points back to make it up to an 85.

Midterm performance overall

- Average and median were in the 70s
- Reasonable range from our perspective considering regrades
- Bear in mind: only worth 25% of your final grade. Same as the final; much less than labs

Questions/Comments/Etc.

- Please do check through your midterm to make sure that we haven't missed anything, double check our math, etc.
- We're always happy to explain what we thought was wrong with an answer or how we mapped various errors to point values
- We're almost never going to be willing to change that mapping
    - In short: there are 60 of you and many students probably made the same mistake. We gave the same points to everyone with equivalent answers
- So: please do come, especially if something seems off or if you're confused! But we want to be clear about likely outcomes.