CS 61B
Spring 2019

Iterators and Iterables

Exam Prep 5: February 18, 2018

1 Filtered List

We want to make a `FilteredList` class that selects only certain elements of a `List` during iteration. To do so, we're going to use the `Predicate` interface defined below. Note that it has a method, `test` that takes in an argument and returns `True` if we want to keep this argument or `False` otherwise.

```
public interface Predicate<T> {
        boolean test (T x);
}
```

For example, if `L` is any kind of object that implements `List<String>` (that is, the standard `java.util.List`), then writing

```
FilteredList<String> FL = new FilteredList<String> (L, filter);
```

gives an iterable containing all items, `x`, in `L` for which `filter.test(x)` is `True`. Here, `filter` is of type `Predicate`. Fill in the `FilteredList` class below.

```
import java.util.*;
public class FilteredList<T> implements Iterable<T> {
    List<T> list;
    Predicate<T> pred;
    public FilteredList (List<T> L, Predicate<T> filter) {
        this.list = L;
        this.pred = filter;
    }

    @Override
    public Iterator<T> iterator () {
        return new FilteredListIterator(list, pred);
    }

    private class FilteredListIterator implements Iterator<T> {
        List<T> list;
        Predicate<T> pred;
        int index;
        public FilteredListIterator(List<T> l, Predicate<T> f) {
            list = l;
            pred = f;
            index = 0;
        }


```

```
26
27          @Override
28          public boolean hasNext() {
29              while (index < list.size() && !pred.test(list.get(index)) {
30                  index += 1;
31              }
32              return index < list.size();
33          }
34
35          @Override
36          public T next() {
37              if (!hasNext()) {
38                  throw new NoSuchElementException();
39              }
40              index += 1;
41              return list.get(index - 1);
42          }
43      }
44  }
```

2 Iterator of Iterators

Implement an `IteratorOfIterators` which will accept as an argument a `List` of `Iterator` objects containing `Integers`. The first call to `next()` should return the first item from the first iterator in the list. The second call to `next()` should return the first item from the second iterator in the list. If the list contained `n` iterators, the `n+1`th time that we call `next()`, we would return the second item of the first iterator in the list.

For example, if we had 3 `Iterators` A, B, and C such that A contained the values `[1, 2, 3]`, B contained the values `[4, 5, 6]`, and C contained the values `[7, 8, 9]`, calls to `next()` for our `IteratorOfIterators` would return `[1, 4, 7, 2, 5, 8, 3, 6, 9]`

Feel free to modify the input `a` as needed.

Note - this is only one possible solution, as there are many others.

```
1  import java.util.*;
2  public class IteratorOfIterators implements Iterator<Integer>  {
3      LinkedList<Integer> l;
4      public IteratorOfIterators (ArrayList<Iterator<Integer>> a) {
5          l = new LinkedList<>();
6          int i = 0;
7          while (!a.isEmpty()) {
8              Iterator<Integer> curr = a.get(i);
9              if (!curr.hasNext()) {
10                 a.remove(curr);
11                 i -= 1; //or else we'll skip an element
12             } else {
13                 l.add(curr.next());
14             }
15             if (a.isEmpty()) { //could've removed the last Iterator
16                 break;
17             }
18             i = (i + 1) % a.size();
19         }
20     }
21
22     @Override
23     public boolean hasNext() {
24         return !l.isEmpty();
25     }
26
27     @Override
28     public Integer next() {
29         return l.removeFirst();
30     }
31 }
```

3 Every kth Element (Fall 2014 MT1 Q5)

Fill in the `next()` method in the following class. Do not modify anything outside of `next`.

```
1  import java.util.Iterator;
2  import java.util.NoSuchElementException;
3  /** Iterates over every Kth element of the IntList given to the constructor.
4  *   For example, if L is an IntList containing elements
5  *   [0, 1, 2, 3, 4, 5, 6, 7] with K = 2, then
6  *       for (Iterator<Integer> p = new KthIntList(L, 2); p.hasNext(); ) {
7  *           System.out.println(p.next());
8  *       }
9  *   would print get 0, 2, 4, 6. */
10 public class KthIntList implements Iterator <Integer> {
11     public int k;
12     private IntList curList;
13     private boolean hasNext;
14
15     public KthIntList(IntList I, int k) {
16         this.k = k;
17         this.curList = I;
18         this.hasNext = true;
19     }
20
21     /** Returns true iff there is a next Kth element. Do not modify. */
22     public boolean hasNext() {
23         return this.hasNext;
24     }
25
26     /** Returns the next Kth element of the IntList given in the constructor.
27      *  Returns the 0th element first. Throws a NoSuchElementException if
28      *  there are no Integers available to return. */
29     public Integer next() {
```

```
1          if (curList == null) {
2              throw new NoSuchElementException();
3          }
4          Integer toReturn = curList.head;
5          for (int i = 0; i < k && curList != null; i++) {
6              curList = curList.tail;
7          }
8          hasNext = (curList != null);
9          return toReturn;
```