CS 61B Fall 2020

Iterators and Iterables

Exam Prep Discussion 5: September 21, 2020

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.Iterator;
import java.util.Iterable;
import java.util.NoSuchElementException;
public class FilteredList<T> ________________________________________ {


        public FilteredList (List<T> L, Predicate<T> filter) {

        }

        @Override
        public Iterator<T> iterator() {

        }









}
```

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.

```
import java.util.*;
public class IteratorOfIterators  _______________________________ {


    public IteratorOfIterators(List<Iterator<Integer>> a) {









    }

    @Override
    public boolean hasNext() {




    }



    @Override
    public Integer next() {




    }
}
```

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() {
30         ____________________________________________________________
31         ____________________________________________________________
32         ____________________________________________________________
33         ____________________________________________________________
34         ____________________________________________________________
35         ____________________________________________________________
36         ____________________________________________________________
37         ____________________________________________________________
38         ____________________________________________________________
39         ____________________________________________________________
40     }
41 }
```