CSM 61B Spring 2019

Iterators & Exceptions

Mentoring 4: February 18, 2019

1 Iterator Interface

In Java, an **iterator** is an object which allows us to traverse a data structure in linear fashion. Every iterator has two methods: `hasNext` and `next`.

```
interface IntIterator {
    boolean hasNext();
    int next();
}
```

1.1 Consider the following code that demonstrates the `IntArrayIterator`.

```
int[] arr = {1, 2, 3, 4, 5, 6};
IntIterator iter = new IntArrayIterator(arr);
if (iter.hasNext()) {
    System.out.println(iter.next());     // 1
}
if (iter.hasNext()) {
    System.out.println(iter.next() + 3); // 5
}
while (iter.hasNext()) {
    System.out.println(iter.next());     // 3 4 5 6
}
```

Define an `IntArrayIterator` class that works as described above.

```
public class IntArrayIterator implements IntIterator {
    private int index;
    private int[] array;
    public IntArrayIterator(int[] arr) {
        array = arr;
        index = 0;
    }
    public boolean hasNext() {
        return index < array.length;
    }
    public int next() {
        int value = array[index];
        index += 1;
        return value;
    }
}
```

1.2 Define an `IntListIterator` class that adheres to the `IntIterator` interface.

```
public class IntListIterator implements IntIterator {
    private IntList node;
    public IntListIterator(IntList list) {
        node = list;
    }
    public boolean hasNext() {
        return node != null;
    }
    public int next() {
        int value = node.first;
        node = node.rest;
        return value;
    }
}
```

Meta: This can be gone through and explained faster since the students just implemented `IntArrayIterator`.

1.3 Define a method, `printAll`, that prints every element in an `IntIterator` **regardless of how the iterator is implemented.**

```
public static void printAll(IntIterator iter) {
    while (iter.hasNext()) {
        System.out.println(iter.next());
    }
}
```

Meta: In general, for all parts of this question, give students time to work on the problem as well as prompting them to think about what they need in order to start them off. (For example, what do you need to do to implement the `IntIterator` interface? What do you need to know for `hasNext` to work?)

2 VoteIterator

2.1 Define `VoteIterator`, an `IntIterator` that takes in an **`int[]`** `array` of vote counts and iterates over the votes. The input array contains the number of votes each candidate received.

```
int[] array = { 0, 2, 1, 0, 1, 0 };
```

Each candidate, represented by their index, `i`, should be returned from each call to `next()` `array[i]` times in total. Given the input above, calls to `next()` would eventually return `1` *twice*, `2` *once*, and `4` *once*.

```
public class VoteIterator implements IntIterator {
    private int[] votes;
    private int index, current;
    public VoteIterator(int[] votes) {
        this.votes = votes;
        this.index = this.current = 0;
    }
    public boolean hasNext() {
        if (current < votes[index]) {
            return true;
        }
        for (int i = index + 1; i < votes.length; i++) {
            if (votes[i] > 0) {
                return true;
            }
        }
        return false;
    }
    public int next() {
        while (current == votes[index]) {
            index += 1;
            current = 0;
        }
        current += 1;
        return index;
    }
}
```

Meta: Students often get stuck at understanding what the question wants them to implement. Go through an example (like the provided array) to let them understand what the solution should return at each step.

3 Catch Blocks, Exceptions, and Try-s, Oh My

3.1

```
try {
    doSomething();
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("caught array index exception");
} catch (Exception e) {
    System.out.println("caught an exception");
    throw e;
} catch (NullPointerException e) {
    System.out.println("caught null pointer exception");
} finally {
    System.out.println("in finally block");
}
```

The code is a trick! Java doesn't like ambiguity or unreachable code, so `catch (NullPointerException e)` is a compile-time error as `NullPointerException` is a subclass of `Exception`. Assume the catch block is removed.

(a) What will print if `doSomething()` throws a `NullPointerException`?

```
caught an exception
in finally block
NullPointerException
```

(b) What if `doSomething()` throws an `ArrayIndexOutOfBoundsException`?

```
caught array index exception
in finally block
```

(c) What if `doSomething()` doesn't error?

```
in finally block
```

Meta: Remember that completing a catch prevents all other catches in the same block from running. Even on a re-throw, no other catches in the same block are called, because the exception has already been 'handled'!

Slides: Mudit's Slides