Iterators, Iterables

Discussion 05

Announcements

Week 5

- ❏ Midterm 1 is this Wednesday
- ❏ Lab sections on Wednesday will be OH for midterm material
- ❏ Homework 1 is due next Monday
- ❏ Live lecture this Wednesday

Content Review

Subtype Polymorphism

Subtype polymorphism allows us to use a subtype of a class or interface in its place so we can write functions that can operate on a wide array of objects so long as they have a certain attribute in common.

```
public ComparableArray <T implements Comparable> implements Comparable { ... }
```

In this example, we wouldn't be able to compare the elements of our ComparableArray unless they too were comparable, so we bind our generic to be Comparable items only. Now we can do:

```
T item1 = ...;
T item2 = ...;
item1.compareTo(item2);
```

Exceptions

Exceptions are used to stop the code and inform the person running it when something occurs that is not "allowed".

Examples:

NullPointerException - You tried calling something on a variable that contains null.

IndexOutOfBoundsException - You tried accessing an array index larger than the size of the array.

You can throw your own exceptions:

```
throw new RuntimeException("Because I said so.");
```

You can catch exceptions too (so your code keeps running):

```
try {
    /* Something that may throw an exception */
} catch (Exception e) {
    /* Do something about it */
}
```

Iterators & Iterable

Iterators are objects that can be iterated through in Java (in some sort of loop).

```
public interface Iterator<T> {
    boolean hasNext();
    T next();
}
```

Iterables are objects that can produce an iterator.

```
public interface Iterable<T> {
    Iterator<T> iterator();
}
```

Worksheet

CS 61B
Fall 2020

Iterators and Iterables

Discussion 5: September 21, 2020

1 Iterators Warmup

1. If we were to define a class that implements the interface `Iterable<Integer>`, what method(s) would this class need to define? Write the function signature(s) below.

   public Iterator<Integer> iterator()

2. If we were to define a class that implements the interface `Iterator<Integer>`, what method(s) would this class need to define? Write the function signature(s) below.

   public boolean hasNext()
   public Integer next()

3. What's one difference between `Iterator` and `Iterable`?

   iterable produces an iterator
   ↳ are the ones we actually iterate over.

2 OHQueue

The goal for this question is to create an iterable Office Hours queue. We'll do so step by step.

The code below for `OHRequest` represents a single request. Like an `IntNode`, it has a reference to the `next` request. `description` and `name` contain the description of the bug and name of the person on the queue.

```
1  public class OHRequest {
2      public String description;
3      public String name;
4      public OHRequest next;
5
6      public OHRequest(String description, String name, OHRequest next) {
7          this.description = description;
8          this.name = name;
9          this.next = next;
10     }
11 }
```

First, let's define an iterator. Create a class `OHIterator` that implements an iterator over `OHRequest` objects that only returns requests with good descriptions. Our `OHIterator`'s constructor will take in an `OHRequest` object that represents the first `OHRequest` object on the queue. We've provided a function `isGood` that accepts a `description` and says if the description is good or not. If we run out of office hour requests, we should throw a `NoSuchElementException` when our iterator tries to get another request.

```
import java.util.Iterator;
public class OHIterator _____ {
    OHRequest curr;

    public OHIterator(OHRequest queue) {

    }

    public boolean isGood(String description) {
        return description != null && description.length() > 5;
    }
```

```
}
```

Now, define a class `OHQueue`. We want our `OHQueue` to be iterable, so that we can process `OHRequest` objects with good descriptions. Our constructor will take in an `OHRequest` object representing the first request on the queue.

```
import java.util.Iterator;
public class OHQueue ___________________________________ {


    public OHQueue (OHRequest queue) {

    }









}
```

Fill in the main method below so that you make a new `OHQueue` object and print the names of people with good descriptions. Note : the main method is part of the `OHQueue` class.

```
public class OHQueue ... {
    ....

    public static void main(String [] args) {
        OHRequest s5 = new OHRequest("I deleted all of my files", "Alex", null);
        OHRequest s4 = new OHRequest("conceptual: what is Java", "Omar", s5);
        OHRequest s3 = new OHRequest("git: I never did lab 1", "Connor", s4);
        OHRequest s2 = new OHRequest("help", "Hug", s3);
        OHRequest s1 = new OHRequest("no I haven't tried stepping through", "Itai", s2);


        for (_____________ : _________________) {

        }

}
```

3 Thank u, next

Now that we have our `OHQueue` from problem 2, we'd like to add some functionality. We've noticed that occasionally in office hours, the system will put someone on the queue twice. It seems that this happens whenever the description contains the words "thank u." To combat this, we'd like to define a new version of our previous iterator, `TYIterator`.

If the current item's description contains the words "thank u," it should skip the next item on the queue. As an example, if there were 4 `OHRequest` objects on the queue with descriptions `["thank u", "thank u", "im bored", "help me"]`, calls to `next()` should return the 0th, 2nd, and 3rd `OHRequest` objects on the queue. Note: we are still enforcing good descriptions on the queue as well!

Hint - To check if a description contains the words "thank u", you can use:

```
curr.description.contains("thank u")
```

```
public class TYIterator extends ______________________________________ {
    boolean skipNext = false;
    public TYIterator(OHRequest queue) {


    }









}
```