CSM 61B Fall 2017

Exceptions & More Interfaces

Mentoring 5: September 25, 2017

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

1.1
```
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

2 Mercantilism

2.1 Let's model a feudal society where managers, merchants, and workers cooperate to deliver goods. What happens when we call **new** `Manager().work()`?

```
class Manager {
    Merchant merchant = new Merchant();
    Worker worker = new Worker();
    String[] goods = new String[1];
    void work() {
        try {
            merchant.trade(goods);
        } catch (RuntimeException e) {
            worker.produce("apple pie", goods);
            merchant.trade(goods);
            worker.produce("cornbread", goods);
        } catch (Exception e) {
            merchant.trade(goods);
        } finally {
            System.out.println("All in a day's work");
        }
    }
}
class Merchant {
    void trade(String[] goods) {
        try {
            for (String good : goods) {
                if (good != null) {
                    System.out.println("Traded 1 " + good);
                    return;
                }
            }
            throw new RuntimeException("Not enough goods");
        } catch (Exception e) {
            System.out.println("Oops");
            throw e;
        } finally {
            System.out.println("I love trading");
        }
    }
}
class Worker {
    void produce(String item, String[] goods) {
        int i = 0;
        while (goods[i] != null) {
            i += 1;
        }
        goods[i] = item;
        System.out.println("Done making 1 " + item);
    }
}
```

```
Oops
I love trading
Done making 1 apple pie
Traded 1 apple pie
I love trading
All in a day's work
ArrayIndexOutOfBoundsException
```

Meta: This is a walkthrough question. Giving students time to work on their own usually results in one small mistake leading to a tangent off into different aspects of the code. Highly recommended to have students chime in as you step through the problem on the board, or through slides.

Slides: Mudit's Slides

3 VoteIterator

3.1 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.