

## 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);  
} finally {  
    System.out.println("in finally block");  
}
```

- (a) What will print if `doSomething()` throws a `NullPointerException`?
  
  
  
  
  
  
  
  
  
  
- (b) What if `doSomething()` throws an `ArrayIndexOutOfBoundsException`?
  
  
  
  
  
  
  
  
  
  
- (c) What if `doSomething()` doesn't error?

## 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()`?

```

1  class Manager {
2      Merchant merchant = new Merchant();
3      Worker worker = new Worker();
4      String[] goods = new String[1];
5      void work() {
6          try {
7              merchant.trade(goods);
8          } catch (RuntimeException e) {
9              worker.produce("apple pie", goods);
10             merchant.trade(goods);
11             worker.produce("cornbread", goods);
12         } catch (Exception e) {
13             merchant.trade(goods);
14         } finally {
15             System.out.println("All in a day's work");
16         }
17     }
18 }
19 class Merchant {
20     void trade(String[] goods) {
21         try {
22             for (String good : goods) {
23                 if (good != null) {
24                     System.out.println("Traded 1 " + good);
25                     return;
26                 }
27             }
28             throw new RuntimeException("Not enough goods");
29         } catch (Exception e) {
30             System.out.println("Oops");
31             throw e;
32         } finally {
33             System.out.println("I love trading");
34         }
35     }
36 }
37 class Worker {
38     void produce(String item, String[] goods) {
39         int i = 0;
40         while (goods[i] != null) {
41             i += 1;
42         }
43         goods[i] = item;
44         System.out.println("Done making 1 " + item);
45     }
46 }

```

### 3 VoteIterator

- 3.1 Define `VoteIterator`, an iterator 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 };
```

Given the input above, calls to `next()` would eventually return 1 *twice*, 2 *once*, and 4 *once*. Make sure your iterator adheres to standard iterator rules.