Practicing Time Analysis

- Consider the following static method:

```
public static int mystery(int n) {
    int x = 0;
    for (int i = 0; i < n; i++) {
        x += i;          // statement 1
        for (int j = 0; j < i; j++) {
            x += j;
        }
    }
    return x;
}
```

- What is the big-O expression for the number of times that statement 1 is executed as a function of the input **n**?
  - the outer loop performs n repetitions
  - statement 1 is executed once for every repetition – n times!
  - n is O(n)!

What about now?

- Consider the following static method:

```
public static int mystery(int n) {
    int x = 0;
    for (int i = 0; i < 3*n + 4; i++) {
        x += i;          // statement 1
        for (int j = 0; j < i; j++) {
            x += j;
        }
    }
    return x;
}
```

- What is the big-O expression for the number of times that statement 1 is executed as a function of the input **n**?
  - the outer loop performs 3*n + 4 repetitions
  - statement 1 is executed once for every repetition: 3*n+4 times!
  - 3***n** + 4 is O(**n**)!

Practicing Time Analysis

- Consider the following static method:

```
public static int mystery(int n) {
    int x = 0;
    for (int i = 0; i < n; i++) {
        x += i;          // statement 1
        for (int j = 0; j < i; j++) {
            x += j;      // statement 2
        }
    }
    return x;
}
```

- What is the big-O expression for the number of times that statement 2 is executed as a function of the input **n**?
  - the outer loop performs n repetitions
  - the inner loop performs i repetitions

Practicing Time Analysis

- Consider the following static method:

```
public static int mystery(int n) {
    int x = 0;
    for (int i = 0; i < n; i++) {
        x += i;          // statement 1
        for (int j = 0; j < i; j++) {
            x += j;      // statement 2
        }
    }
    return x;
}
```

- What is the big-O expression for the number of times that statement 2 is executed as a function of the input **n**?

value of i	number of times statement 2 is executed
0	0
1	1
…	…
n – 1	n – 1

$1 + 2 + \ldots + n - 1 = O(n^2)$
(as we saw in selection sort!)