Exercise: Operations Time?

- Arithmetic operations (add, subtract, divide, modulo)
  - Integer ops usually practically faster than floating point
- Accessing a stack variable
- Accessing a field of an object
- Accessing a single element of an array
- Doing a primitive comparison (equals, less than, greater than)
- Calling a function/method but NOT waiting for it to finish
- Raising an arbitrary number to arbitrary power
- Allocating an array
- Checking if two Strings are equal
- Determining if an array or ArrayList contains() an object
Practice

Two functions to reverse an array. Discuss
▶ Big-O estimates of runtime of both
▶ Big-O estimates of memory overhead of both
  ▶ Memory overhead is the amount of memory in addition to the input required to complete the method
▶ Which is practically better?
▶ What are the exact operation counts for each method?

reverseE

```java
public static void reverseE(Integer a[]){
    int n = a.length;
    Integer b[] = new Integer[n];
    for(int i=0; i<n; i++){
        b[i] = a[n-1-i];
    }
    for(int i=0; i<n; i++){
        a[i] = b[i];
    }
}
```

reverseI

```java
public static void reverseI(Integer a[]){
    int n = a.length;
    for(int i=0; i<n/2; i++){
        int tmp = a[i];
        a[i] = a[n-1-i];
        a[n-1-i] = tmp;
    }
    return;
}
```
public static String toString( Object [ ] arr )
{
    String result = " [";
    for( String s : arr )
        result += s + " ";
    result += "]";
    return result;
}

- Give a Big-O estimate for the runtime
- Give a Big-O estimate for the memory overhead
Multiple Input Size

What if "size" has two parameters?

- $m \times n$ matrix
- Graph with $m$ vertices and $n$ edges
- Network with $m$ computers and $n$ cables between them

Exercise: Sum of a Two-D Array

Give the runtime complexity of the following method.

```java
public int sum2D(int [][] A){
    int M = A.length;
    int N = A[0].length;
    int sum = 0;
    for(int i=0; i<M; i++){
        for(int j=0; j<N; j++){
            sum += A[i][j];
        }
    }
    return sum;
}
```