Arrays and Subscripts

Decide who will be what role for today's activity. List the names of your team members next to their role.

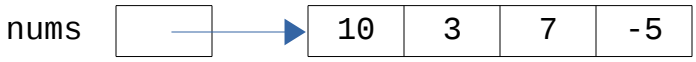
Manager:
Spokesperson:
Quality Control/Recorder:
Process Analyst:

If you have only three people, one may serve as both manager and spokesperson. If you have five team members, then assign two people to process analyst.

Model 1: Memory Diagram

Array elements are stored together in one *contiguous* block of memory. To show arrays in memory diagrams, we simply draw adjacent boxes.

```
int[] nums = {10, 3, 7, -5};
```



Questions (15 min)

1. Draw a memory diagram for the following array declarations.

```
int[] sizes = new int[5];
sizes[2] = 7;
```

```
char[] codes = new char[3];
codes[2] = 'X';
```

```
double[] costs = new double[4];
costs[0] = 0.99;
```

```
Die[] dice = new Die[2];
dice[1] = new Die(6);
```

2. What is the *default* value for uninitialized array elements in arrays? (Hint: You should have no empty boxes in your memory diagram.)

3. Determine how much memory each of these arrays will require. Assume that Java references require 64 bits (8 bytes).

Model 2: Array Initialization

Arrays can be initialized using an initialization list enclosed in braces:

```
int[] sizes = {3, 5, 7, 2, 1};
String[] names = {"James", "Madison", "University"};
```

However, this syntax only works for initialization. If an array has already been initialized, its contents can be changed with the following notation:

```
sizes = new int[] {55};
names = new String[] {"bob", "ann", "sue", "sam"};
```

Questions (10 min)

5. Write *statements* that declare and initialize variables for the arrays.

0	14	1024	127	3	5521

3.23	1.52	4.23	32.5	2.45	5.23	3.33

6. What is the type and value for each of the four *expressions* below?

```
int[] a = {3, 6, 15, 22, 100, 0};
double[] b = {3.5, 4.5, 2.0, 2.0, 2.0};
String[] c = {"alpha", "beta", "gamma"};

1.   a[3] + a[2]

2.   b[2] - b[0] + a[4]

3.   c[1].charAt(a[0])

4.   a[4] * b[1] <= a[5] * a[0]
```

Model 3: Arrays and Loops

The real power of arrays is the ability to process them using loops, i.e., performing the same task for multiple elements. The standard form of iteration is as follows:

```
for (int i = 0; i < array.length; i++) {
   ... process array[i] ...
}
```

For example:

```
// set all of the elements of x to -1.0
double[] x = new double[100];
for (int i = 0; i < x.length; i++) {
    x[i] = -1.0;
}

// sum the elements of scores
int sum = 0;
for (int i = 0; i < scores.length; i++) {
    sum += scores[i];
}
```

Questions (15 min)

7. What is the value of `array` and `accumulator` after the following iteration? Trace the loop by hand in the space below.

```
int[] array = {5, 26, 13, 12, 37, 15, 16, 4, 1, 3};
int accumulator = 0;
for (int i = 0; i < array.length; i++) {
    if (array[i] % 2 == 1 && i + 1 < array.length) {
        array[i] *= -1;
        accumulator += array[i+1];
    }
}
```

8. Implement the following method that creates and returns a double array.

```
/**
 * Return a new array containing the pairwise maximum value of
 * the arguments. For each subscript i, the return value at [i]
 * will be the larger of x[i] and y[i].
 *
 * @param x an array of double values
 * @param y an array of double values
 * @return pairwise max of x and y; returns null if x.length
           != y.length or if either of the arrays is null
 */
public static double[] pairwiseMax(double[] x, double[] y) {









}
```