## Model 1 Arrays and Loops (optional)

The real power of arrays is the ability to process them using loops, i.e., performing the same task for multiple elements.

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

Here are two specific examples:

```
// 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) Start time:

**1**. What is the value of `array` and `accumulator` at the end of the following code? 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];
    }
}
```

| i | array[i] | accum |
|---|---|---|
| 0 | 5 | 0 |
| 1 | 26 | 26 |
| 2 | 13 | 26 |
| 3 | 12 | 38 |
| 4 | 37 | 38 |

| i | array[i] | accum |
|---|---|---|
| 5 | 15 | 53 |
| 6 | 16 | 69 |
| 7 | 4 | 69 |
| 8 | 1 | 69 |
| 9 | 3 | 72 |

```
array:
  { -5, 26, -13, 12, -37,
   -15, 16,   4, -1,   3}

accumulator:
  72
```

**2**. Implement the following method that creates and returns a new 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
 */
public static double[] pairwiseMax(double[] x, double[] y) {
     double[] z = new double[x.length];
     for (int i = 0; i < x.length; i++) {
         if (x[i] > y[i]) {
             z[i] = x[i];
         } else {
             z[i] = y[i];
         }
     }
     return z;
}
```

**3**. Implement the following method that reads through two integer arrays.

```
/**
 * Computes the final average grade for a student. The labs are
 * worth 40% and the exams are worth 60%. All scores range from
 * 0 to 100, inclusive.
 *
 * @param labs the student's lab scores
 * @param exams the student's exam scores
 * @return weighted average of all scores
 */
public static double finalGrade(int[] labs, int[] exams) {
     int labSum = 0;
     for (int score : labs) {
         labSum += score;
     }
     int examSum = 0;
     for (int score : exams) {
         examSum += score;
     }
     double labGrade = 1.0 * labSum / labs.length;
     double examGrade = 1.0 * examSum / exams.length;
     return 0.40 * labGrade + 0.60 * examGrade;
}
```