Review: Arrays

- **array**: stores many values of the same type
- **element**: one value in an array
- **index**: desired element from 0 to a.length – 1

```
int[] a = new int[10];
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- **Arrays are automatically initialized to the "zero" value.**

Review: Arrays

- **Array elements are selected with a[index] and can be used exactly the same as regular variables.**

```
a[i] = a[i-1] + a[i-2];
```

- **Special syntax for initializing arrays with preset values.**

```
int[] primes = {2, 3, 5, 7, 11, 13, 17};
```

Zybooks Example Multiple arrays for letter postage rates

```
// Weights in ounces
double[] letterWeights = {1.0, 2.0, 3.0, 3.5, 4.0, 5.0, 6.0,
                          7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0};
// Costs in cents (usps.com 2017)
int[] postageCosts = {49, 70, 91, 112, 161, 182, 203,
                      224, 245, 266, 287, 308, 329, 350};
```

```
// Weights in ounces
// Costs in cents (usps.com 2017)
double[] letterWeights = { 1,  2,  3, 3.5,   4,   5,   6,   7,   8,   9,  10,
int[] postageCosts =     {49, 70, 91, 112, 161, 182, 203, 224, 245, 266, 287,
```

Loop design pattern: The Boolean flag to exit the loop.

```
// Prompt user to enter letter weight
System.out.print("Enter letter weight (in ounces): ");
userLetterWeight = scnr.nextDouble();

// Postage costs is based on smallest letter weight greater than
// or equal to mailing letter weight
boolean foundWeight = false;
int i;

for (i = 0; (i < letterWeights.length) && (!foundWeight); ++i) {
   if( userLetterWeight <= letterWeights[i] ) {
      foundWeight = true;
      System.out.print("Postage for USPS first class mail is ");
      System.out.println(postageCosts[i] + " cents");
   }
}

if( !foundWeight )
   System.out.println("Letter is too heavy for USPS first class
```

Alternatively use break

```
// Prompt user to enter letter weight
System.out.print("Enter letter weight (in ounces): ");
userLetterWeight = scnr.nextDouble();

// Postage costs is based on smallest letter weight greater than
// or equal to mailing letter weight
boolean foundWeight = false;
int i;

for (i = 0; i < letterWeights.length; ++i) {
   if( userLetterWeight <= letterWeights[i] ) {
      foundWeight = true;
      break;
   }
}

if ( foundWeight ) {
   System.out.print("Postage for USPS first class mail is ");
   System.out.println(postageCosts[i] + " cents");
} else
   System.out.println("Letter is too heavy for USPS first class
```

Zybooks Challenge Activity:
Write a loop that sets newScores to oldScores shifted once left, with element 0 copied to the end.
Ex: If oldScores = {10, 20, 30, 40}, then newScores = {20, 30, 40, 10}.

Student solution:

Zybooks solution:

Copy last after the loop:

Use **%** to wrap around:

```
for (i = 0; i < SCORES_SIZE; i++)
    newScores[i] = oldScores[(i+1)%SCORES_SIZE];
```

Operations on arrays:

- **find maximum or minimum value** (Zybooks 5.4)
- **find index of maximum or minimum**
- **sum of array** (Zybooks 5.4)
- **average of array** (Zybooks 5.4)
- **running sum of array** (Lecture 16)
- **find index of first or last match**
- **count number of matches** (Zybooks 5.4)
- **reverse the array** (Zybooks 5.8)
- **shift or rotate an array up or down** (Zybooks 5.7)
- **randomly shuffle the array**
- **sort the array**
- **find the median (middle value with ½ larger and ½ smaller)**
- **find the mode (most common value)**

```
index:   [0]  [1]  [2]  [3]  [4]  [5]  [6]  [7]  [8]  [9] [10] [11]
values:   3    4    5    1    2    3    4    5    6    2    3    4
```

Finding the maximum value

```
// Determine largest (max) number
int max = values[0];                    // Largest so far
for (int i = 0; i < SIZE; i++) {
   if (values[i] > max)
      max = values[i];
}
System.out.println("max: " + max);
```

Activity: find the index of the maximum value
Activity: find the index of the first match
Activity: find the index of the last match
Activity: randomly shuffle the array
Activity: sort the array
Activity: find the median (middle value with ½ larger and ½ smaller)
Activity: find the mode (most common value)