Arrays
Array

- An **array** is a group of similar variables, all of the same type, and with systematically related names that involve special syntax using [...] 
- Each **array element**, e.g., \(a[0]\), \(a[1]\), …, acts like a single variable of the type used in the declaration of array \(a\)
Compare to Mathematics

• This is entirely parallel to the use of subscripted variables in mathematics, e.g., $x_0, x_1, \ldots$

• Just as $x_0$ is pronounced “x-sub-0” in mathematics, $a[0]$ is usually pronounced “a-sub-0” in a Java program

• Consider, similarly, $x_{i+2}$ and $a[i+2]$
Compare to Mathematics

• In mathematics, a group of related variables \( x_0, x_1, \ldots, x_{n-1} \) is called a vector \( \mathbf{x} \) of length \( n \)

• In Java, a group of variables \( a[0], a[1], \ldots, a[n-1] \) is called an array \( \mathbf{a} \) of length \( n \)
Declaring an Array

\texttt{int[]} a;
Declaring an Array

```cpp
int[] a;
```

The `[]` in this declaration indicates there will be some number of variables named `a[0]`, `a[1]`, ... But, how many?
Declaring and Creating an Array

```java
int[] a = new int[4];
```
Declaring and Creating an Array

```java
int[] a = new int[4];
```

This many! Here, 4 is called the **length** of the array, and it is the value of another variable introduced by this declaration:

```java
a.length
```
Declaring and Creating an Array

```java
int[] a = new int[4];
```

```
0 0 0 0
```

```
a.length = 4
```
Understanding Arrays

```java
int[] a = new int[4];
```
Understanding Arrays

```java
int[] a = new int[4];

int a[0] = 0;
int a[1] = 0;
int a[2] = 0;
int a[3] = 0;
int a.length = 4;
```

This is illegal Java code, but it is the net effect of the array declaration/creation above.
Declaring and Initializing an Array

```cpp
int[] a = { 6, 18, 9, -10 };
```
Declaring and Initializing an Array

```java
int[] a = { 6, 18, 9, -10 };
```

Here again, we have:

```
a.length = 4
```

But now the 4 array elements have different initial values:

```
a[0] = 6
a[1] = 18
e tc.
```
Declaring and Initializing an Array

```java
int[] a = { 6, 18, 9, -10 };
```

![Diagram of array initialization with values 6, 18, 9, and -10 in the array `a`]
Understanding Arrays

```cpp
int[] a = { 6, 18, 9, -10 };`
Understanding Arrays

```java
int[] a = { 6, 18, 9, -10 };
int a[0] = 6;
int a[1] = 18;
int a[2] = 9;
int a[3] = -10;
int a.length = 4;
```

This is illegal Java code, but it is the net effect of the array declaration/initialization above.
Array Indexing with Constants

• You may write an `int constant (literal) c` between `[…]` as in `a[c]`, so long as its value satisfies:

\[ 0 \leq c < a.length \]

• Example:

```java
int[] a = new int[4];
a[3] = 17;
```
Array Indexing with Constants

- You may write an `int constant (literal)` $c$ between $[\ldots]$ as in $a[c]$, so long as its value satisfies:

$$0 \leq c < a.length$$

- Example:

```java
int[] a = new int[4];
a[3] = 17;
```

After this code is executed, we have $a[3] = 17$
Array Indexing in General

• You may write an *int-valued expression* `exp` between `[...]` as in `a[exp]`, so long as its value satisfies:

```
0 ≤ exp < a.length
```

• Example:

```java
int[] a = new int[4];
a[a.length - 1] = 17;
```
Array Indexing in General

• You may write an *int*-valued expression \( \text{exp} \) between \([\ldots]\) as in \( a[\text{exp}] \), so long as its value satisfies:
  \[
  0 \leq \text{exp} < a\.\text{length}
  \]

• Example:

\[
\text{int}[\] a = \text{new int}[4];
ap[a\.\text{length} - 1] = 17;
\]

After this code is executed, we have:
\[
a[3] = 17
\]
Resources

• Java Tutorials
  – http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

• Java for Everyone, Chapter 6
  – https://library.ohio-state.edu/record=b8347056~S7