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., \( a[0], a[1], \ldots \), 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 $x$ of length $n$

• In Java, a group of variables $a[0], a[1], \ldots, a[n-1]$ is called an array $a$ of length $n$
Declaring an Array

```java
int[] a;
```
Declaring an Array

\[ \text{int}[] \ a; \]

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

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

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

This many! Here, 4 is called the \textit{length} of the array, and it is the value of another variable introduced by this declaration: `a.length`
Declaring and Creating an Array

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

![Diagram of an array with initial values and length]
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

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

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

Here again, we have:

```
a.length = 4
```

But now the 4 array elements have different initial values:

```
a[0] = 6
a[1] = 18
```

etc.
Declaring and Initializing an Array

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>18</td>
<td>9</td>
<td>-10</td>
</tr>
</tbody>
</table>

a.length = 4
Understanding Arrays

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

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

This is illegal Java code, but it is the net effect of the array declarationinitialization above.

    int a[0] = 6;
    int a[1] = 18;
    int a[2] = 9;
    int a[3] = -10;
    int a.length = 4;
```
Array Indexing with Constants

• You may write an \texttt{int constant (literal)} \( c \) between \([\ldots]\) as in \texttt{a[c]}, so long as its value satisfies:

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

• Example:

\[
\begin{align*}
\texttt{int[]} \ a & = \texttt{new int[4]}; \\
a[3] & = 17;
\end{align*}
\]
Array Indexing with Constants

• You may write an \textit{int constant (literal) }\texttt{c} between \texttt{[...]} as in \texttt{a[c]}, so long as its value satisfies:

\[ 0 \leq c < a.\text{length} \]

• Example:

\begin{verbatim}
int[] a = new int[4];
a[3] = 17;
\end{verbatim}

After this code is executed, we have \texttt{a[3] = 17}
Array Indexing in General

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

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

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

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

• Example:

```java
int[] a = new int[4];
a[a.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
  – http://osu.worldcat.org/title/java-for-everyone-late-objects/oclc/808511232