lab 4: ext. 4 day

Chapter 5: Arrays.

An array is a contiguous set of memory locations all storing the same type of data.

<table>
<thead>
<tr>
<th>Memory address</th>
<th>list array indices</th>
<th>array name</th>
<th>array contents or array elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>1</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>1008</td>
<td>2</td>
<td>-12</td>
<td></td>
</tr>
<tr>
<td>1012</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
access array elements using indices

```
  list[2] refers to -12

  System.out.println(list[2] + " " + list[4]);
  // output: -12 13
```

```
  list[53] = 99; // puts 99 into array
```

To declare an array variable:

```
  int[1] list;
```

**Note**: this does not allocate any space for an array object.
To allocate an array object

```java
list = new int[10];
```

Can declare and allocate at once:

```java
int[] list = new int[10];
```

To initialize we can do
list[0] = 5
list[1] = 23
.
list[8] = -2
list[9] = 15

Note: It is an error to reference indices outside range 0-9.

ArrayIndexOutOfBoundsException

We can obtain an array's length by the expression.
list.length

so we can do

int n = list.length;

then use n to control a loop that processes the array.

ex. ArrayEx1.java

ex. ArrayEx2.java
To declare, allocate and initialize all at once:

\[ \text{int list} = [0, 3, 6, 9, 12, 15, 18, 21, 24, 27] \]

\underline{note:} length 10 is inferred by compiler by length at \( i \) ...

\underline{note:} array name are fundamentally different from primitive variable names.
int a = 6;

\[
\begin{array}{c}
\text{a} \\
\hline
6
\end{array}
\]

int x = \{5, 6, 7\};

\[
\begin{array}{c}
\text{x} \\
\hline
5 \\
6 \\
1 \\
7 \\
2
\end{array}
\]

The reference var. A actually stores the address of the 0th array element.
more generally, a reference variable stores the address of some other object in memory.

```
|Reference Variable| Some Object|
```

Note: these are called pointer variables in other languages.
Ex.

```java
String word = "happy";
```

Recall: two categories of data types.

- **Primitive**: `& int`, ...
- **Reference**: all others
  - `String, Scanner`, ...
  - `int`, `String`, ...
  - base type `int` is Primitive,
    - array type `int[]` is Reference.
Ex. Primitive types

```java
int a = 6, b;
b = a;
b = 7;
System.out.println(a);  // Prints 6
```

Ex. Reference types: Arrays

```java
int[] A = {5, 6, 7};
int[] B;
B = A;
```

```java
System.out.println(A[2]); // Prints 8
```

**Note:**

An array parameter to a function is the reference (i.e., address) not an array object.