Function overloading.

Ex:
Overload1.java
Overload2.java
Chapter 5: Arrays & Containers

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

```
<table>
<thead>
<tr>
<th>addr</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>104</td>
<td>108</td>
<td>112</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Addresses in memory:

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

One int
To declare an array variable:

```
int[] list;
```

To allocate an array object, use `new`:

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

Declareration / allocation in one step:

```
int[] list = new int[10];
```
Resulting Picture of Memory:

```
list
  |
  v
  5
  0 1 2 3 -3 4 5 6 7 8 9
reference variable

array object
```

to initialize:

```
list[0] = 5;
list[4] = -3;
list[9] = list[0] + list[4];
```
Read from array:

System.out.println (list[i]);

The length of an array is given by: \( \text{the expression} \)

\[ \text{list.length} \]

\[ \text{evaluates to 10} \]

Note: Index in brackets can be any int expression:

\begin{align*}
\text{int } i &= 2, j = 4; \\
\text{list }[i + j] &= 15;
\end{align*}
This expression must be in range:

\[ 0 \leq \text{expression} \leq \text{list.length} - 1 \]

equivalently:

\[ 0 \leq \text{exp} < \text{list.length} \]

Ex. ArrayEx1.java

Ex. ArrayEx2.java
Declare, allocate, initialize all at once:

```java
int[] list = {0, 3, 6, 9, 12, 15, 18, 21, 24, 27};
```
Primitive types and reference types are fundamentally different.

```plaintext
int a = 6;

int[] A = [5, 6, 7];
```

Reference variable `A` references an array with elements 5, 6, and 7.
Ex: String word = "happy";

- handle

Reference

Same object

Object type

Another variable

In this case, variables are also in

 Syndicate

"Hello"
Two Categories of data type

- Primitive: byte, char, short, int, long, float, double, boolean
- Reference: all others
  - String, Scanner, ...
  - int[], double[], String[]

Base type may be primitive or may be reference.
Primitive type semantics

Ex.

```java
int a = 6, b;
  a [6]  b
b = a;
b = 7;
System.out.println(a);  // 6
```
Reference type semantics

Ex.

\[ \text{int} \{ 1 \} A = \{ \{ 5, 6, 7 \} \} \]

\[ \text{int} \{ 2 \} B \]

\[ B[2] = 8 \]

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