Revised: 4/8/2019

Arrays and Methods

Michael C. Hackett

Computer Science Department

Community College of Philadelphia

Lecture Topics

- Arrays as Method Arguments
- Methods that return Arrays
- Variable-Length Arguments

Colors/Fonts

• Local Variable Names	–	Brown
• Primitive data types	–	Fuchsia
• Literals	–	Blue
• Keywords	–	Orange
• Object names	–	Green
• Operators/Punctuation	–	Black
• Field Names	–	Lt Blue
• Method Names	–	Purple
• Parameter Names	–	Gold
• Comments	–	Gray
• Package Names	–	Pink

Source Code	–	**`Consolas`**
Output	–	`Courier New`

Arrays as Method Arguments

- An array can be passed to a method as an argument.
- Must match the array type specified as the parameter.

```
public int sum(int[] numbers)
```

Arrays as Method Arguments

```
public int sum(int[] numbers) {
    int sum = 0;
    for(int number : numbers) {
        sum += number;
    }
    return sum;
}
```

```
int[] threeNums = {4, 5, 6};
sum(threeNums);
```

Would return 15.

Arrays as Method Arguments

- Arrays are always passed to a method **by reference**.

- **Pass by reference**- The reference to data is passed to the method.
  - Arrays and Objects are always passed by reference in Java.

- **Pass by value**- The data is passed to the method.
  - Primitive data are always passed by value in Java.

Passing by Value

```
public void demoMethod(int number) {
    number = 0;
}
```

Changes the number parameter, not x.

```
int x = 5;
demoMethod(x);
```

Passes x’s value as the argument.

Passing by Reference

```
public void demoMethod(int[] array) {
    array[1]= 0;
}
```

Changes the threeNums array.

```
int[] threeNums = {4, 5, 6};
demoMethod(threeNums);
```

Passes threeNums's reference as the argument.

Variable Length Arguments

- ***Variable Length Arguments*** (or ***varargs***) allow a method to accept an undetermined number of parameters/arguments.

```
public int sum(int... numbers)
```

- The varargs must all be of the correct type.

- The varargs will be treated as an array inside the method.
  - Varargs *are* arrays, just not declared as such.

Variable Length Arguments

```
public int sum(int... numbers) {
    int sum = 0;
    for(int number : numbers) {
        sum += number;
    }
    return sum;
}
```

`sum(4, 5, 6);`	Valid. Would return 15.
`sum(2, 3);`	Valid. Would return 5.
`sum(7, 8.5);`	Not valid.

Variable Length Arguments

```
public int sum(int... numbers) {
    int sum = 0;
    for(int number : numbers) {
        sum += number;
    }
    return sum;
}
```

```
int[] myOriginalArray = {3, 5, 7, 9};

sum(myOriginalArray);
```

You can pass an array to a vararg. The sum method would return 24 in this example.

Variable Length Arguments

- No additional parameters can follow a vararg.

```
public int doMath(int... numbers, String operationType) {   INVALID
```

- Although, there can be any number of normal parameters preceding it.

```
public int doMath(String operationType, int... numbers) {   VALID
```

Variable Length Arguments

```
public int doMath(String operationType, int... numbers) {
    int answer = 0;
    if(operationType.equals("+")) {
        for(int number : numbers) {
            answer += number;
        }
    } else if(operationType.equals("*")) {
        answer = 1;
        for(int number : numbers) {
            answer *= number;
        }
    }
    return answer;
}
```

```
doMath("+", 4, 5, 6);
doMath("*", 7, 3);
```

Valid. Would return 15.
Valid. Would return 21.

Variable Length Arguments

```
public int doMath(String operationType, int... numbers) {
    int answer = 0;
    if(operationType.equals("+")) {
        for(int number : numbers) {
            answer += number;
        }
    } else if(operationType.equals("*")) {
        answer = 1;
        for(int number : numbers) {
            answer *= number;
        }
    }
    return answer;
}
```

```
int[] threeNums = {4, 5, 6};
doMath("+", threeNums);
```

Valid. Would return 15.

Returning an Array from a Method

- An array can be returned by a method.
  - Be sure the method’s return type is an array.

```
public int[] getNumbers() {
   int[] threeNums = {4, 5, 6};
   return threeNums;
}
```