Lec 3: Methods and Arrays

Sam McCauley

February 17, 2026

Admin

- Lab 1 today
  - Get settled, then meet in the bigger room with windows
  - We'll do introductions and then get started
  - You'll download the starter code, and submit your final lab, using `git`. The Lab 00 instructions may be helpful to review
  - Remember to ask questions!
- No class Friday! (Winter Carnival)

Methods

Methods in Java

- Often called “functions” in other languages
- Snippets of code we can call repeatedly
- May or may not have *parameters* (way for us to give information to the method)
- May or may not have a *return value* (way for the method to pass back information)
- These are just the basics of methods. We’ll learn much more when we cover objects and classes

Method Syntax

```java
public static int addThreeInts(int x, int y, int z) {
    int sum = x + y + z;
    return sum;
}

public static void main(String[] args) {
    int total = addThreeInts(1, 2, 4);
    System.out.println(total);
    System.out.println(addThreeInts(10, 20, 30));
}
```

- Use `public static` at beginning of method (for now)
- State the *type* of value being returned: this method returns an `int`
  - If you are returning nothing, use `void`
- Then name of the method: `addThreeInts`
- List the parameters in the parentheses. Don't forget their types!
- The `return` keyword tells Java to go back to where the function was called. The original function call is "replaced" with the return value

Method Syntax

Let's trace execution through the following code.

```
1  public static int addThreeInts(int x, int y, int z) {
2      int sum = x + y + z;
3      return sum;
4  }
5
6  public static void main(String[] args) {
7      int total = addThreeInts(1, 2, 4);
8      System.out.println(total);
9      System.out.println(addThreeInts(10, 20, 30));
10 }
```

Method Practice

- Let’s write some code to print all of the prime numbers from 1 to 100
- (Prime means: not evenly divisible by any positive integer other than 1 and itself)

Arrays (First data structure!)

What is an array?

- Sequence of items, kind of like a list
- Access any one of the items using brackets: `[` and `]`
- The first item is index `0`
- *Fixed* length!
    - If you want to “change” the length, you need to make a brand new array and copy everything over

An Array

Declaring and Using Arrays

- Like variables, specify the type up front: array of `int`, `float`, `char`, `String`, etc.
- Use brackets after the type to specify that it is an array
- Need to initialize it using the “new” keyword with its size (we’ll come back to this)

```
1 int[] arr = new int[3];
2 arr[0] = 4;
3 arr[1] = 3;
4 System.out.println(arr[0] + arr[1]); //prints 7
```

Array Length

- Can get the length of an array using `.length`

```
1  int[] arr = new int[3];
2  arr[0] = 4;
3  arr[1] = 3;
4  System.out.println(arr.length); //prints 3
```

Declaring Short Arrays

- For short arrays where you know the contents up front, you can use this syntax instead:

```
1  int[] arr = {2,4,8,16};
2  System.out.println(arr[0] + arr[3]);
```

Printing an Array

- Let’s try to print an array and see what happens
- Answer: we get some random-looking characters back
- In pairs: How can we print an array? What do we want the computer to do?
- Let’s try it together

Strings

Strings in Java

- Strings can be declared like any other variable
- Note that `String` is capitalized
    - String is actually a class, not a primitive type—we'll come back to this next week.

```
1    String message = "Hello World!";
2    System.out.println(message);
```

String Concatenation

- Concatenate with +

```
1       String message1 = "Hello ";
2       String message2 = "World!";
3       System.out.println(message1 + message2); //prints: Hello
           World!
```

String Concatenation

- Can also concatenate (using +) `Strings` with non-`String` variables: converts them to a `String` and then concatenates.

- Very useful for output!

```
1      String message1 = "x is ";
2      int x = 100;
3      System.out.println(message1 + x); //prints: x is 100
4      System.out.println("x is " + 100); //prints: x is 100
```

String length

- Can get length with `.length()`

- Note that for `Strings` there are parentheses, but not for arrays!

```
1       String message = "Hello World!";
2       System.out.println(message.length()); //prints 12
```

Newline character

- `\n` is the newline character: prints a newline

```
1       System.out.println("Line 1");
2       System.out.print("Line 2\nLine 3\n");
```

Prints:

```
Line 1
Line 2
Line 3
```

Wrapping Up Operations

Arithmetic Shortcuts

- The notation `a += b` is a shortcut for `a = a + b`
- other operations like `-=`, `*=`, `/=` are defined similarly.
- `a++` is a shortcut for `a = a + 1`.
-

Lines 2–4 of the following code all do the same thing.

```
1    int x = 1;
2    x = x + 1; //x is now 2
3    x += 1;    //x is now 3
4    x++;       //x is now 4
```

Logical Operators

- Often we want to combine conditionals
  - && (and): true if *both* conditional expressions are true
  - || (or): true if *at least one* of the conditional expressions is true. (Could be both, or could be just one.) The key to press to make this character is located above the enter key.
  - ! (not): converts true into false and false into true
- Use parentheses with these!
- Let's do an example

Roller Coaster Requirements

- Height must be between 52" and 77"
- Kids under 12 must have a parter who is over 12
- In Pairs: how can we write this using Java notation?