ICS 111
Introduction to Computer Science I

Nikki Manuel
Leeward Community College

Methods

Week 10
Fall 2019

Methods

What is a Method?

A **method** is a group of statements that perform an operation. We use methods as a way to avoid retyping code.

You've already used methods before:

```
System.out.println()
Math.pow(x, y)
Math.random()
```

Defining a Method

We can create our own methods!

To define a new method, we use this syntax:

```
   modifier returnValueType methodName(parameters)
{
      // Method body
   }
```

Note: We write our new method separately from the **`main`** method!

Example #1: Adding a Set of Numbers Together

I want to find the sum of the numbers from 1 to 10, from 41 to 76, and from 203 to 598. We can find the answer like this:

```
int sum1to10 = 0;
for (int i = 1; i <= 10; i++) sum1to10 = sum1to10 + i;
System.out.println(sum1to10);

int sum41to76 = 0;
for (int i = 41; i <= 76; i++) sum41to76 = sum41to76 + i;
System.out.println(sum41to76);

int sum203to598 = 0;
for (int i = 203;  i <= 598; i++) sum203to598 = sum203to598 + i;
System.out.println(sum203to598);
```

Example #1: Adding a Set of Numbers Together

This all looks similar… it follows a pattern!

```
int sum1to10 = 0, sum41to76 = 0, sum203to598 = 0;

for (int i = 1; i <= 10; i++)
     sum1to10 = sum1to10 + i;
System.out.println(sum1to10);

for (int i = 41; i <= 76; i++)
     sum41to76 = sum41to76 + i;
System.out.println(sum41to76);

for (int i = 203;  i <= 598; i++)
     sum203to598 = sum203to598 + i;
System.out.println(sum203to598);
```

Example #1: Adding a Set of Numbers Together

So let's create a **method** that will just let us write this code once and reuse it.

```
public static int exampleSum(int num1, int num2) {
    int sumOfNumbers = 0;
    for (int i = num1; i <= num2; i++) {
        sumOfNumbers = sumOfNumbers + i;
    }

    return sumOfNumbers;
}
```

Example #1: Adding a Set of Numbers Together

We can now use our **exampleSum** method to find the sum of the numbers from 1 to 10, from 41 to 76, and from 203 to 598:

```
public static int exampleSum(int num1, int num2) {
    int sumOfNumbers = 0;
    for (int i = num1; i <= num2; i++)
        sumOfNumbers = sumOfNumbers + i;
    return sumOfNumbers;
}
public static void main(String[] args) {
    System.out.println("Sum from 1 to 10: " + exampleSum(1, 10));
    System.out.println("Sum from 41 to 76: " + exampleSum(41, 76));
    System.out.println("Sum from 203 to 598: " + exampleSum(203, 598));
```

Example #1: Syntax

Syntax:

modifier

return value type

method name

parameters

method head

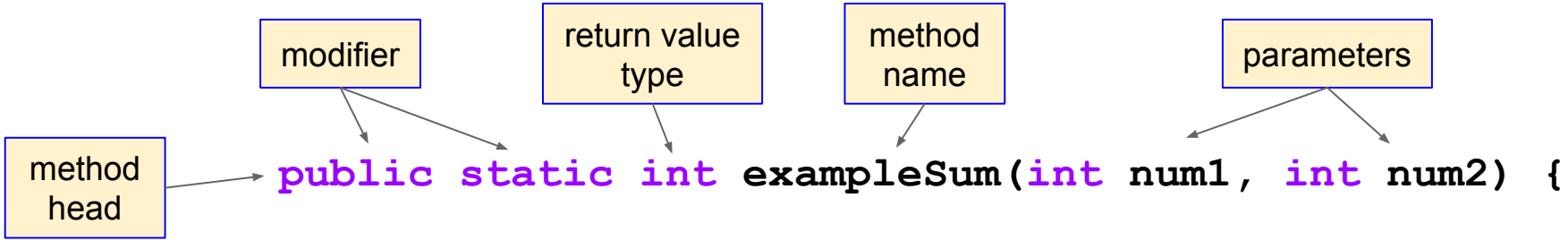
```
public static int exampleSum(int num1, int num2) {
    int sumOfNumbers = 0;
    for (int i = num1; i <= num2; i++) {
        sumOfNumbers = sumOfNumbers + i;
    }

    return sumOfNumbers;
}
```

method body

return value

Method Syntax : Method Head



Modifier: We'll learn why we use `public static` next week!

Return Value Type: A method may or may not return a value. The Return Value Type is the data type of the value that your method will return. If it doesn't return any value, use the keyword `void`.

Parameters: The data you'll pass along to use for your method. Optional.

Method Syntax: Method Body

Syntax:

```
int sumOfNumbers = 0;
for (int i = num1; i <= num2; i++) {
    sumOfNumbers = sumOfNumbers + i;
}
return sumOfNumbers;
```

method body

return value

Method Body: These statements are executed when you use the method.

Return: In order for the method to return a result when used, you *must* use the keyword `return`. The method terminates when the return statement is executed.

Practice Making More Methods!

Create a method called exampleMax that takes two numbers and returns the number that is highest.

Create a method called weatherMethod that takes an integer representing temperature. If the number is > 90, it prints the string “It’s a hot day!” If the number is 72 - 89, it prints the string “It’s a pleasant day.” If the number is below 72, it prints “I guess it’s winter in Hawai’i!” -- Create this method in 2 ways: one with and one without a return statement.

More Method Practice!

Write a method that increments a given number by one.

Write a method that prints a message a specified number of times.

Write a method that swaps the values of two variables.

Write a method that takes a number from 1-12 and returns the corresponding month name (ex. 1 returns January)

The `main` Method

The `main` method's header: `public static void main(String[] args)`

Just like we've learned about creating methods methods, the `main` method's header includes the modifiers `public` and `static`, the return value type `void`, the method name `main`, and a parameter of the `String[]` type.

The statements in the `main` method can invoke other methods that are defined in the class that contains the main method, or in other classes.

Overloading Methods

You can have different methods with the same name, as long as their signatures are different. This is called **overloading**.

For example, we can have a method to find the max # between 2 numbers and one to find the max # between 3 numbers:

```
public static int maxNum(int num1, int num2) {
    // method body
}
public static int maxNum(int num1, int num2, int num3) {
    // method body
}
```

Test Yourself

What are the benefits of using a method?

How do you define a method?

How do you use a method?

What is the return type of the main method?