# Methods

DEBORAH A. TRYTTEN

CS 1323

# Tip of the Day

- I'm the Chair of the Undergraduate Committee for the School of Computer Science
- This means
  - I can advise students
  - I approve transfer courses
  - I approve and support study abroad
  - I approve admissions to the dual degree program
  - I know where the best resources are at OU for students who are struggling
- My office is Devon 252

# A Word on this Material

- This section is almost certainly the most challenging in the class
- Lots of new and confusing vocabulary to learn and practice
- We will work on this topic for a long time
- We will come back to this topic many times
- Be patient—you can do this

# Context

- Up to this point, all of our programs have had one method: main
- This is starting to cause problems
- The main method is getting too long/complicated
- Makes it impossible to reuse code
- We need to start breaking the main program apart into smaller pieces that can be reused

# Vocabulary: Call and Execute

- The main method exists before a program is run
- The main method is ***called*** by the operating system when the program is run
- The main method is ***executed*** when the program is run
  - ***Call*** and ***execute*** mean pretty much the same thing
  - ***invoke*** is similar

# Example Method

- Write a method to print out a name in a requested order
  - Given first, last
  - One booleans: lastFirst
- Create signature
- When we write this method we have to use ***parameters*** because we don't know the ***arguments***
  - Different arguments will be passed different times the method is called
  - We do not know what value the parameter will have when we write the method
    - Parameters are variables
    - Parameters are placeholders for the values that the arguments will provide when the program is run
- Call this method three times

# Return Values

- Method can only return one thing, at most
  - void means nothing returned
- When a method returns (produces) a value, the return type matches the type of the returned value
  - Math.random()?
- Return statement
  - Causes the program to stop execution of  the method immediately
  - Return control to the program that called it

# Example

- Write a method that returns the maximum of three values
- Call that method in a program with three different arguments

# Think, Pair, Share

- Complete the body of a method that takes finds the average of three integers by filling in the missing line below

```
public static double average(int first, int second, int third)
{
    // Write this code
}
```

- Use this code to find the average of 27, 33, and 14.
- Are first, second, third, parameters or arguments?

# Instant Quiz Question 1

Suppose that we have the following method:

```
public static double mean(double first, double second){
    double result = (first + second) / 2.0;
    return result;  }
```

The proper name for the variable second is:

A: argument

B: parameter

C: local variable

D: method

# Writing Methods

- Must create signature first
  - public static returnType methodName (parameterType parameterName, …)
    - Parameters for inputs
    - Return type describes output
- Find signature for the method below:
- Example
  - Method that returns the longer of two Strings

# Think, Pair, Share

- Find the signature of a method that:
- Reverses a String
  - Example: “abcde” returns “edcba”
- Determines whether a String is or is not empty

# Instant Quiz Question 2

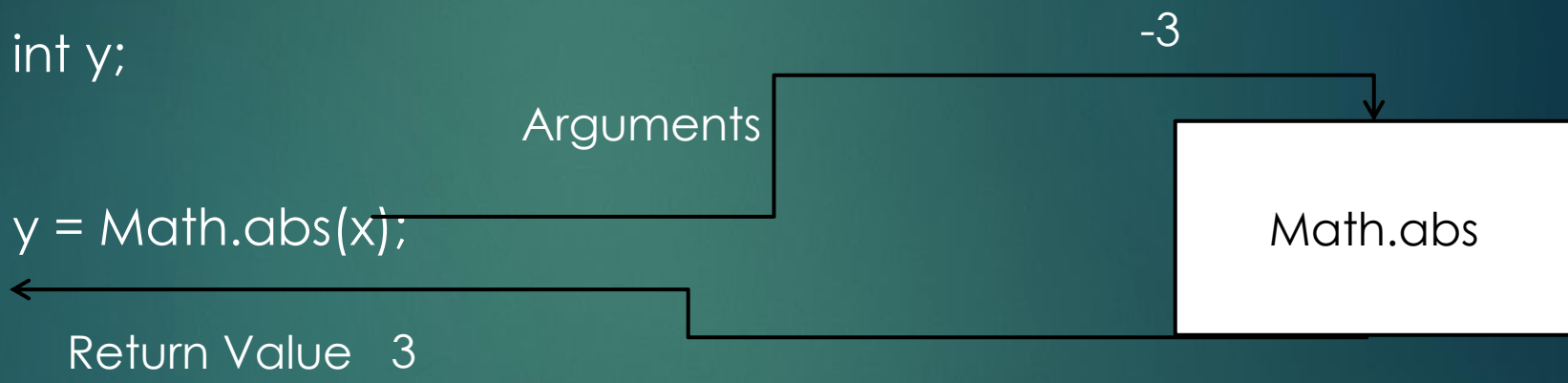
- Which signature below is correct for a method that determines whether a given integer is or is not prime?
- Note: Prime means that it has no factors other than 1 and itself.  7 is prime, but 8 is not because 8 has 2 and 4 as factors.
- A: void isPrime()
- B: void isPrime(int number)
- C: boolean isPrime(double number)
- D: int isPrime(boolean prime)
- E: boolean isPrime(int number)

# Behind the Scenes

- Methods can be confusing because it's hard to tell what's going on behind the scenes
  - Debugging will be impossible if you don't know this
- Must understand the mechanics of
  - Arguments
  - Parameters
  - Return values

# Method Calls Are Interruptions

- int x = -3;
- int y;
- y = Math.abs(x);




- A method call puts the current method on hold
  - The current method resumes when the called method is completed

# Mechanics: Stack Frames

- The main method has a stack frame
- When a method is called, a new stack frame is created
  - Move argument values to parameter variables
  - Create local variables
  - This frame is active until method returns
- When the method returns, a value may return
  - Stack frame is deleted
  - All parameters and local variables in stack frame are lost

# Example

- Show trace of main program:
  - double value = -5.96;
  - double absolute = Math.abs(value);

```
public static double abs(double a) {
    if (a < 0)
        a = -a;   // did we mess up the argument?
    return a;
}
```

# Pass By Value

- In the abs method, the parameter a was changed
- Was the argument value changed?
- Rule: Primitive data type arguments cannot be changed by methods
- The **value** of the argument is passed to the parameter
  - The parameter and argument are in different memory locations and do not influence each other
- Source of many program bugs

# Example

- Write a method that swaps two int values
- Signature
- Write method
- Trace

# Example

- Analyze a method that finds the sum of integer values read from a Scanner object until a -1 sentinel is entered
  - Select signature
  - Write code for method
  - Write code that calls method
  - Show memory diagram
  - Where are the parameters? Arguments? Return value?

# Instant Quiz Question 3

What values are in min and max after the call below?

```
int minAndMax(int first, int second, int third, int min, int max)
{
        min = Math.min(first, Math.min(second, third));
        max = Math.max(first, Math.max(second, third));
        return min;
}
```

- Call:

```
int min = 0, max = 0;
minAndMax(3, 9, 5, min, max);
```

a) min = 3, max = 9

b) min = 0, max = 9

c) min = 0, max = 0

d) min = 3, max = 0

# Big Goal

- Learn to break programs apart into methods during design instead of implementation
- Short methods generally better than long
- Requires practice
  - Sometimes I'll give signatures
  - Sometimes you'll have to find them—especially later in the class

# Example

- Examine using methods to improve the following program (from while loops)
- Write a program that counts how many students passed (70 and over) an examination
  - Enter data in console, separated by spaces
- Why couldn't we do passed/failed, as the original program did?

# Advantages of Methods

- Make it possible to reuse code
- Keep code in manageable chunks
- Promote organized and orderly programming
- Hide unnecessary details from people who don't need to know them
- Clarify communication
  - Parameters for data supplied by the calling method
  - Return value for data returned to the calling method