

methods

Chapter 6



October 23, 2019

Thomas J. Short

Cis 260-01

**Chapter 6 checkpoints**

4.2 How do you define a method? How do you invoke a method?

To define a method, we must choose a method name, then we decide what type of data we want the method to return. The return value type default is void, no value is returned, but we could also use a value returning method such as int, or double. We must also give it a modifier, such as, public static, which I do not know much about yet. Also, all methods must have a method body, which is a collection of statements which implement the method. Optional formal parameters are usually defined in the method signature next to the method name in parentheses, but not always such as math.random() or println().

To invoke or call a method we must know what the method is called and what datatype the formal parameters expect us to pass them. The order in which we enter our actual parameters must match our formal parameters, if the order is wrong, I believe we will receive a runtime error. This is what make the header and the method signature such an important part of defining and invoking a method.

Header modifier return value type **method name (formal parameters**){

Body [ signature above ]

Return value

}

4.5 The return type of a main method is **void** because it does not return a value. The main method is usually the caller program which is typically used to invoke a method or a series of other methods.

4.6 What would be wrong with not writing a return statement in a value returning-method? **If a return value is not given in a value retuning method, we will have a syntax error.** Can u have a return statement in a void method? **You can use a return statement in a void method, but this is only used to exit the method, not common practice.** Does the return statement in the following method cause syntax errors? **Yes, it will cause a syntax error, a return statement returns one value, in rare cases it may return more than one, but not here.**

Public static void xMETHOD(double x, double y){

System.out.println(x + y);

Returns x + y;

}

4.8 Write Method headers for the following methods. (not the bodies)

a. return a sales commission, given the sales amount and commission rate

(a) **public static double salesCommission(double sales amount, double commission)**

b. Display the calendar for a month, given the month and year.

(b)**public static void displayCalendar(month, year)**

c. return a square root of a number

**(c)public static double squareRoot(double value)**

d. Test weather a number is even and returning true if it is

**(d) public static boolean isNumberEven(int, number)**

e. Display a message a specified number of times

**(e) public static void displayMessage(String message, int numberOfTimes)**

f. Return the monthly payment, given the loan amount, number of years, and annual interest rate

**(f) public static double monthlyPayment(double loan amount, int numberOfYears, double annualInterestRate)**

g. Return the corresponding uppercase letter, given a lowercase letter

**(g) public static char convertToUppercase(char, letter)**

5.3 What is pass by value? What is the result of the program on page 216 (a), (b), (c) and (d)?

Pass by value happens when we invoke another method with the actual parameters. We pass a copy of the values we would like to perform calculations on, to the formal parameters at which point our method body or (black box) will take over and perform the calculations and produce one output or (value) for us called a return value. When we are done invoking our method, we pop the tops off our call stacks and the formal parameters are returned to their original values awaiting the next invoking or (call).

1. **0**
2. **2**

**2 4**

**2 4 8**

**2 4 8 16**

**2 4 8 16 32**

**2 4 8 16 32 64**

1. **Before the call, variable times is 3**

**n = 3**

**Welcome to Java!**

**n = 2**

**Welcome to Java!**

**n = 1**

**Welcome to Java!**

**After the call, variable times is 3**

1. **1**

**2 1**

**2 1**

**4 2 1**

**i is 5**

8.3 Given two method definitions

Public static double m(double x, double y)

Public static double m(int x, double y)

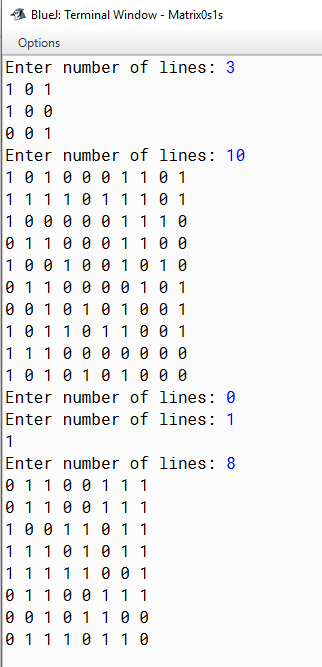
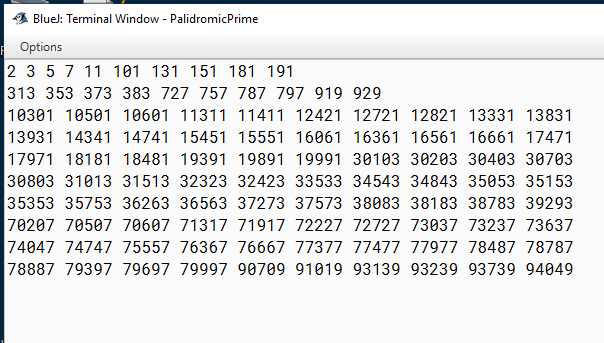
Tell which of the two methods is invoked for:

1. Double z = m (4, 5); **public static double m (int x, double y)**
2. Double z = m (4, 5.4); **public static double m (int x, double y)**
3. Double z = m (4.5, 5.4) **public static double m (double x, double y)**

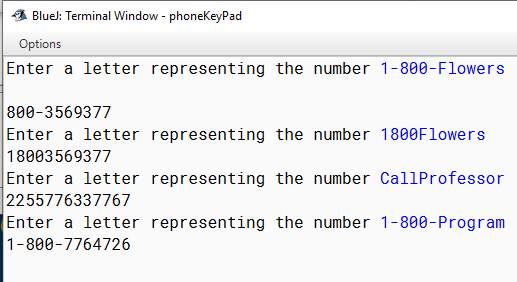
9.2 What is the scope of a local variable?

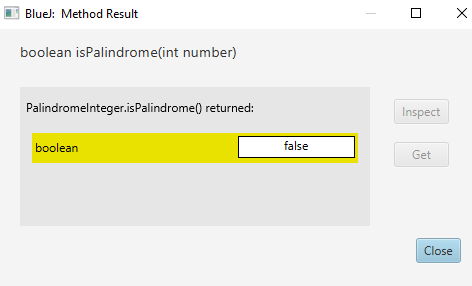
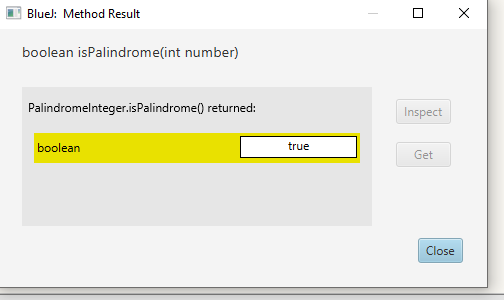
The scope of a variable is the part of a program where we can reference or use our value, thus **the scope of a local variable starts when the variable is declared to the end of the block containing the variable**. If a variable is declared in the middle of a block it would start upon declaration lasting until the end of the block. Parameters help us have a good variable scope by declaring the variables in the beginning of the method so they are good throughout the entirety of the method.

**Print Matrix Result: Palindromic Prime Result:**



Phone Key Pad Result:



**Palindrome Integer Result:**