B.Sc. In Software Development. Year 3.
Applications Programming.
Methods.

LIMERICK INSTITUTE
OF TECHNOLOGY
SCHOOL OF SCIENCE,
ENGINEERING & I.T.
Department of Information Technology

Creating A Method

- In general a method has the following structure.

```
modifer returnValueType methodName(list of parameters)
{
     //body of method
}
```

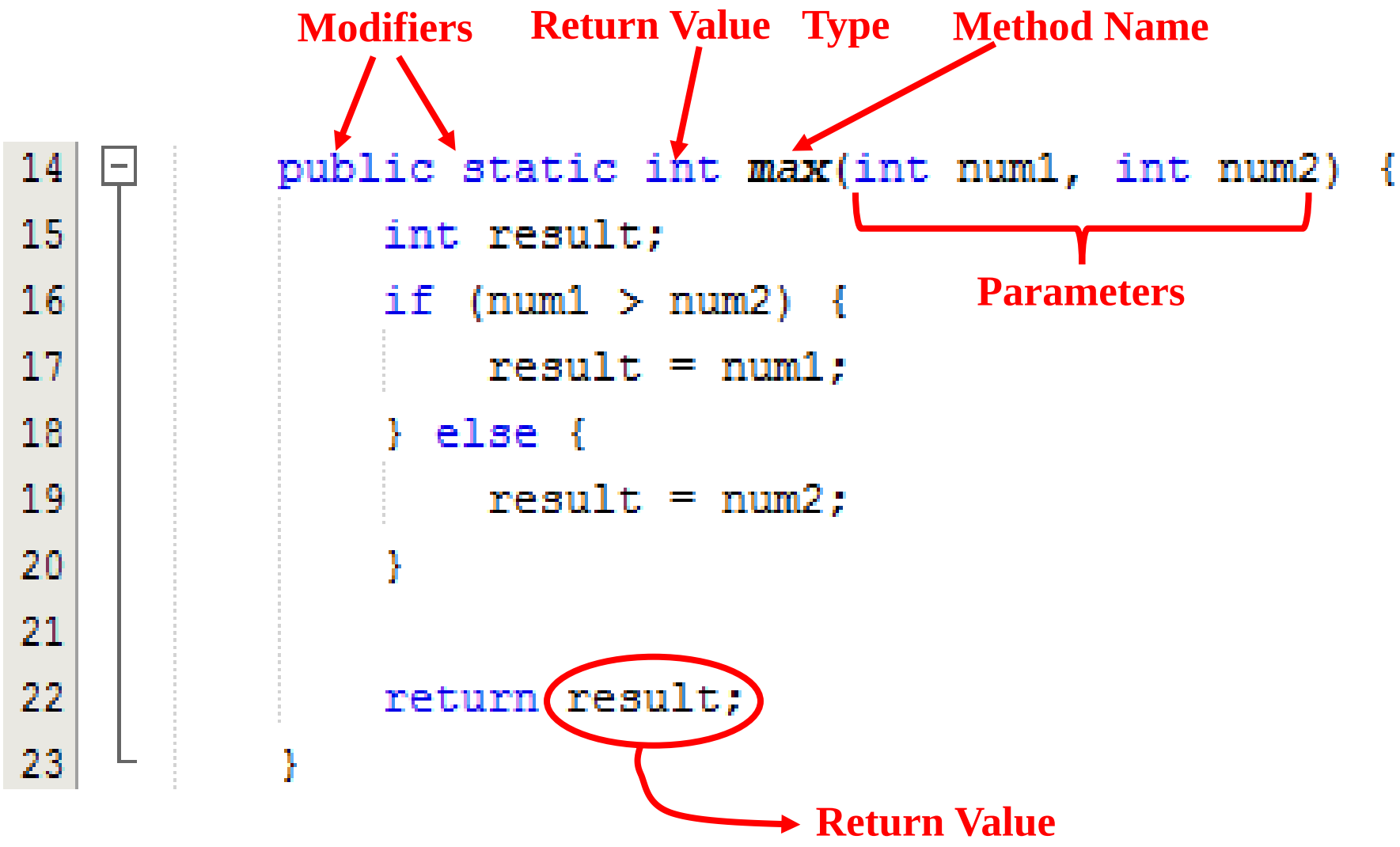
- Lets take a look at a method created to find which of two integers is the largest.

- This method named max, has two `int` parameters, num1 and num2, the larger of which is returned by the method.

Creating A Method

```
14  public static int max(int num1, int num2) {
15      int result;
16      if (num1 > num2) {
17          result = num1;
18      } else {
19          result = num2;
20      }
21
22      return result;
23  }
```

Creating A Method



Creating A Method

- All methods except constructors require a returnValueType.
- The parameters defined in the method header are known as formal parameters.
- When a method is invoked, its formal parameters are replaced by variables or data.
    - These are known as actual parameters.
- The method body contains a list of statements that define what the method does.
- A return statement using the keyword *return* is required for a non-void method to return a result.

Calling A Method

- To use a method you have to call or invoke it.
- There are two ways to call a method.
- The choice is based on whether the method returns a value or not.
- If the method returns a value, a call to the method is usually treated as a value.
- For example, the following code calls max(3,4) and assigns the result of the method to the variable *larger*.

```
int larger = max(3,4);
```

Calling A Method

```
System.out.println(max(3,4));
```

- This code prints the return value of the method call *max(3,4);*
- When a program calls a method, program control is transferred to the called method.
- A called method returns control to the caller when its return statement is executed or when its method-ending brace is reached.
- The following code give the complete listing a program which determines which of two numbers is the largest.

Creating A Method

```
7   public class TestMax {
8
9       public static void main(String[] args) {
10          int i = 5;
11          int j = 2;
12          int k = max(i, j);
13          System.out.println("The larger of the numbers " + i + " and " + j + " is " + k);
14
15      }//end main
16
17      static int max(int num1, int num2) {
18          if (num1 > num2) {
19              return num1;
20          } else {
21              return num2;
22          }
23      }//end max
24
25  }//end class
```

Passing Parameters By Value

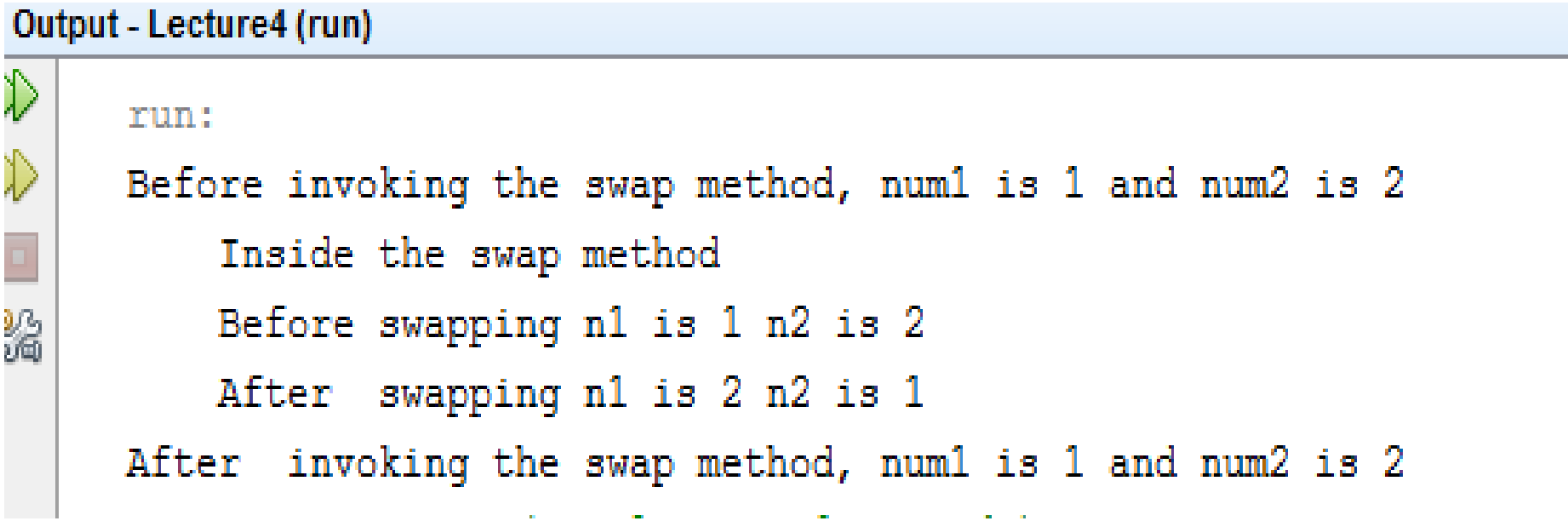
- When you invoke a method with parameters, a copy of the value of the actual parameter is passed to the method.
- This is referred to as pass by value.
- The actual variable outside the method is not affected, regardless of the changes made to the formal parameter inside the method.

Passing Parameters By Value

- Consult *TestPassByValue.java* (in a NetBeans project called Lecture4) demonstrates this point.

- This program creates a method for swapping 2 variables.
  - The swap method is invoked by passing two actual parameters.
  - It should be noted that after the method has been invoked the actual parameters are not changed.

Passing Parameters By Value



Passing Parameters By Value

- What happens if you change the formal parameters name *n1* in swap to *num1*.

- What effect will this have?
  - None.

- This is because it doesn't matter whether the formal parameter and the actual parameter have the same name.

- The formal parameter is a local variable in the method with its own memory space.

- The local variable is allocated when the method is invoked and it disappears when the method is returned to its caller.

Overloading Methods.

- The *max* method that was used earlier works only with the `int` data type.
- But what if you need to find which of two floating-point numbers has the maximum value?
- The solution is to create another method with the same name but different parameters. For example:

```
double max(double num1, double num2) {
 if (num1 > num2)
        return num1;
 else
        return num2;
}
```

Overloading Methods.

- If you call *max* with `int` parameters, the max method that expects `int` parameters will be invoked.

- If you call *max* with `double` parameters, the max method that expects double parameters will be invoked.
  - This is method overloading.
  - This is when two methods have the same name but have different signatures.

- It is the job of the Java compiler to determine which method to use based on its signature.

Overloading Methods.

- In the program *TestMethodOverloading.java (located in a project called Lecture4)* three methods are created.

- The first finds the maximum integer, the second the maximum double and the third finds the largest of three doubles.

- All three methods (though slightly different) have the same name.
  - The methods are said to be overloaded.

Overloading Methods.

- When calling *max(3,4)*, the *max* method for finding the largest of two integers is invoked.

- When calling *max(3.0,5.4),* the *max* method for finding the largest of two doubles is invoked.

- When calling *max(3.0,5.4,10.14),* the *max* method for finding the largest of three doubles is invoked.

- Can you invoke the max method with an int value and a double value such as *max(2,2.5)*?
  - The answer is yes.

- So which method is invoked?

Overloading Methods.

- The answer to that question is that the method for finding the maximum for two *double*s is invoked.

- The actual parameter 2 is automatically converted into a *double* value and passed to this method.

- Why so (you may be wondering) is the method *max(double, double)* not invoked for the call *max(3,4)?*

- Both *max(double, double)* and *max(int, int)* are possible matches for *max(3,4).*

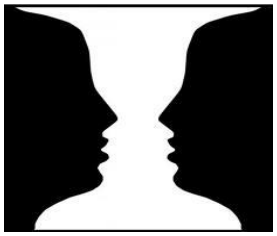
- The Java compiler finds the most specific method for a method invocation.

- Since the method *max(int, int)* is more specific than *max(double, double)*, *max(int, int)* is used to invoke *max(3,4).*

Overloading Methods.

- Overloading methods can make programs clearer and more readable.
- Methods that perform closely related tasks should be given the same name.
- The overloaded methods must differ in their parameter profiles.
- You cannot overload methods based on different modifiers or return types.

Ambiguous Invocation



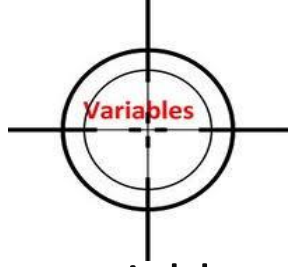
- Sometimes there may be two or more possible matches for an invocation of a method, but the compiler cannot determine the most specific match.

- This is referred to an ambiguous invocation which is a compile error.

- The following code is an example of a program which will generate an ambiguous compile error.

Ambiguous Invocation

```
 7   public class AmbigiousOverloading {
 8
 9       public static void main(String args[]) {
          System.out.println(max(1, 2));
11      }
12
13      public double max(int num1, double num2) {
14          if (num1 > num2) {
15              return num1;
16          } else {
17              return num2;
18          }
19      }
20
21      public int max(double num1, int num2) {
22          if (num1 > num2) {
              return num1;
24          } else {
25              return num2;
26          }
27      }
28
29  }//end class
```



The Scope Of Local Variables

- The scope of a variable is the part of the program where the variable can be referenced.
- A variable defined inside a method is referred to an a local variable.
- The scope of a variable starts from its declaration and continues to the end of the block that contains the variable.
- A local variable must be declared before it can be used.
- You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks.

The Scope Of Local Variables

- The following code is correct:

```
29      public void correctMethod() {
30          int x = 1;
31          int y = 1;
32
33          for (int i = 1; i < 10; i++) { //i is declared.
34              x += i;
35          }
36
37          for (int i = 1; i < 10; i++) { //i is declared.
38              y += i;
39          }
40      }
```

The Scope Of Local Variables

- The following code is incorrect as x is declared in the for loop body block, which is nested inside the method body block where another x is declared.

```
42      public void incorrectMethod() {
43          int x = 1;
44          int y = 1;
45
46          for (int i = 1; i < 10; i++) { //i is declared.
              int x = 0;
48            x += i;
49          }
50      }
```

The Scope Of Local Variables

- If a variable is declared as a method parameter, it cannot be re-declared inside the method.
- The scope of a method parameter covers the entire method.
- A variable declared in the initial action part of a for loop header has its scope in the entire loop.
- But a variable declared inside a for loop body has its scope limited in the loop body from its declaration and to the end of the block that contains the variable.

The Scope Of Local Variables

- Finally, do not declare a variable inside a block and then use it outside the block.
- The following is a common mistake.

```
for (int i =1; i <10; i++)  { //i is declared.
}
int j = i;
```

Constructors

- A constructor method is used to create an instance of an object.

Exercise 1

- Write a program accepts as input the number of hours (as an double) a user has parked in a city centre car park. Thus is to be passed to a method that will return the total charge for the car based on the following:
- The car park charges a €2.00 minimum fee to park for up to three hours.
- The garage charges an additional €0.50 per **hour for each hour or part thereof** in excess of three hours.
- The maximum charge for any given 24 hour period is €10.00.
- Assume that no car parks for no longer than 24 hours at a time.

Exercise 1

- Once the method has retuned the charge payable, it must be displayed.

- You can test your method using the following:
- There is a €2.00 charge for a 3 hour stay in the car park.
- There is a €3.50 charge for a 5.5 hour stay in the car park.
- There is a €4.50 charge for an 8 hour stay in the car park.
- There is a €5.00 charge for an 8.75 hour stay in the car park.
- There is a €9 charge for a 17 hour stay in the car park.

- You may find use for Math.ceil() in your solution. This method returns (as a double) the next whole number.

Exercise 2

- A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is 9009 = 91 × 99.

- Write a program to determine the largest palindrome made from the product of two 3-digit numbers.

- Use at least one method (other than main) in your solution. Consider having a method to determine if a given number is palindromic or not.

- Hint: the answer is *906609*

Exercise 3

- 2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.

- Write a program to compute the smallest positive number that is *evenly divisible* by all of the numbers from 1 to 20?

- Use at least one method (other than main) in your solution.

- Hint: the answer is *232792560*

Exercise 4

- In the Euro Zone the currency is made up of euro (€), and cent (c), and there are eight coins in general circulation:
- 1c, 2c, 5c, 10c, 20c, 50c, €1 (100c) and €2 (200c).
- It is possible to make €2 in the following way:
- 1×€1 + 1×50c + 2×20c + 1×5c + 1×2c + 3×1c
- How many different ways can €2 be made using any number of coins?
- Use at least one method (other than main) in your solution.
- Hint: the answer is *73682.*

References

Joel Murach 2015, *Murach's Beginning Java with NetBeans* Mike Murach & Associates.
ISBN-13 9781890774844 (Link)

Paul Deitel 2017, *Java How To Program (Early Objects) (11th Edition).* Prentice Hall.
ISBN-13 9780134800271 (Link)

Daniel Liang 2017, *Introduction to Java Programming and Data Structures, Comprehensive Version (11th Edition). Pearson.*
ISBN-13 978-0134670942 (Link)