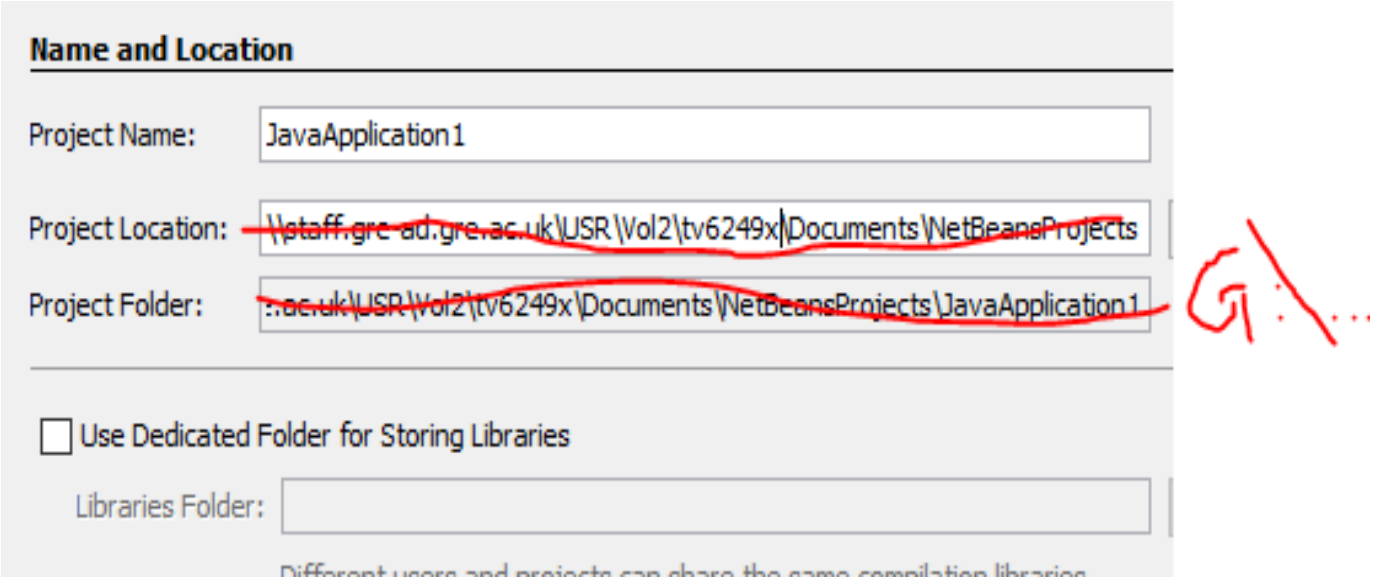
the UNIVERSITY of GREENWICH

COMP1618

Jaa Methods and File IO

Tutorial hints

- Use local storage G:\COMP1618\ or memory stick for lab tasks



the UNIVERSITY of GREENWICH

Outline

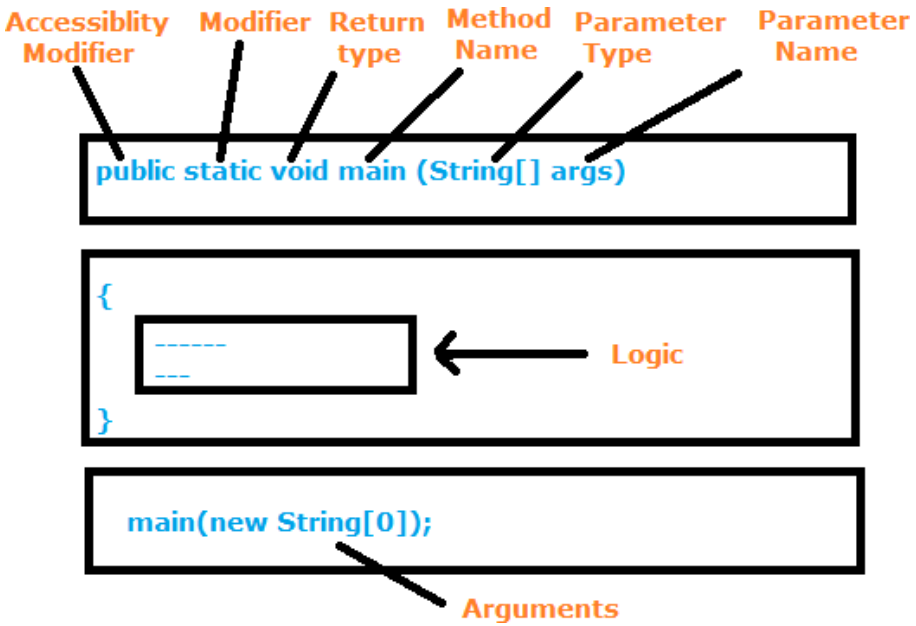
We will discusses the following main topics

- How to declare Java Methods
- Modifiers, Returning type, Passing values, Scope
- Java Library methods
- File I/O

What is a Java method?

```
1  /**
2   * First Java program, which says "Hello, world!"
3   */
4  public class Hello { // Save as "Hello.java"
5     public static void main(String[] args) {   // program entry point
6        System.out.println("Hello, world!");     // print message
7     }
8  }
```

method main()



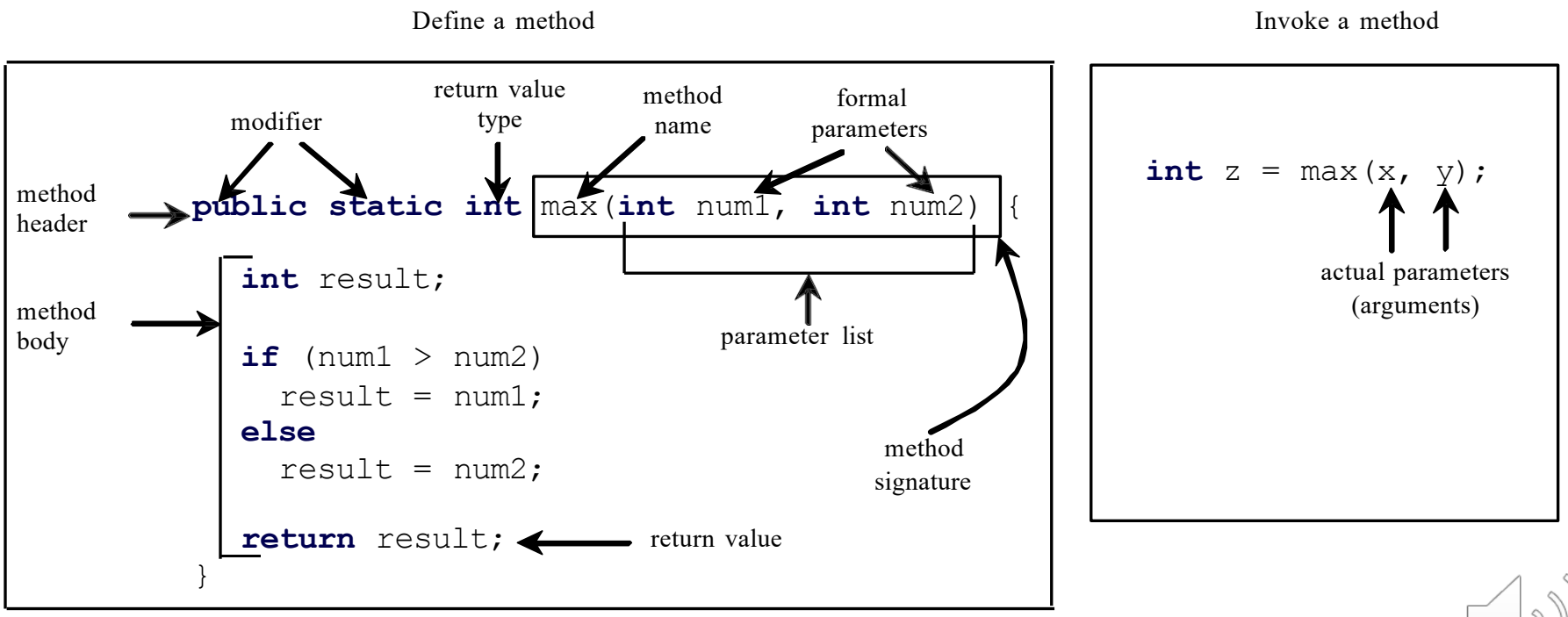
Header

Body

Caller

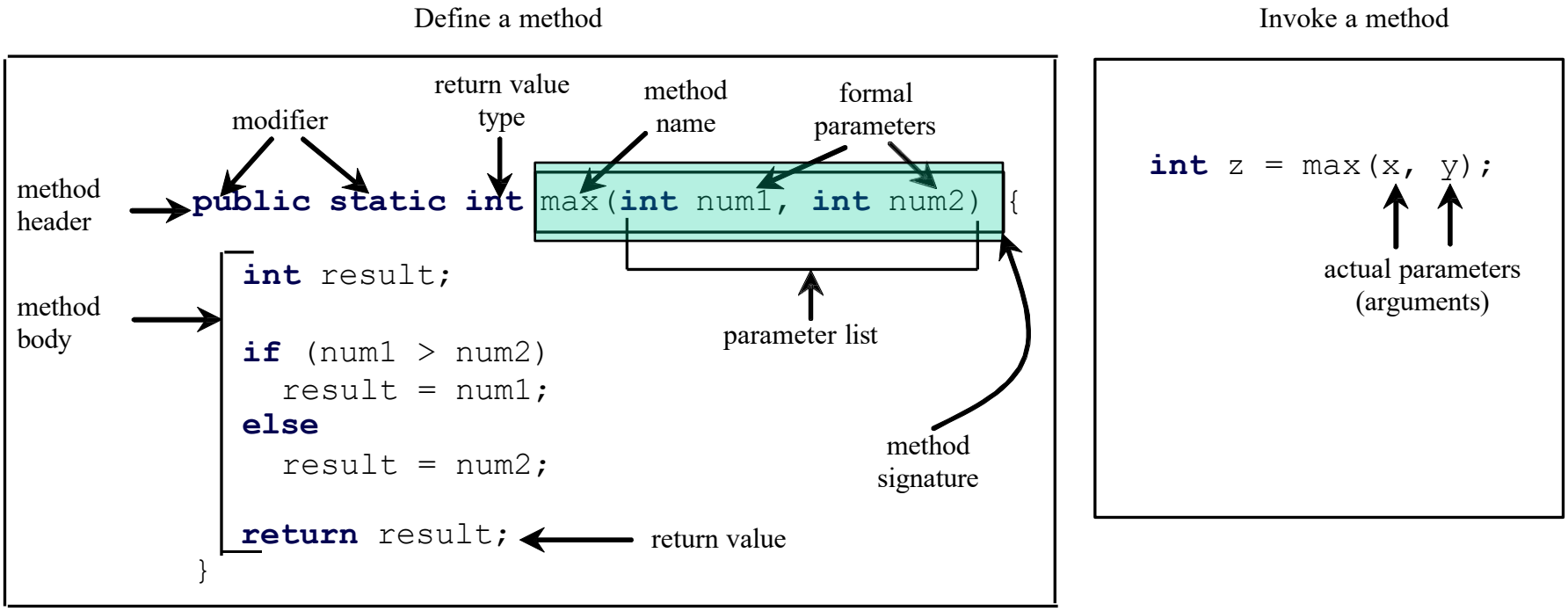
Defining Methods

A method is a collection of statements that are grouped together to perform an operation.



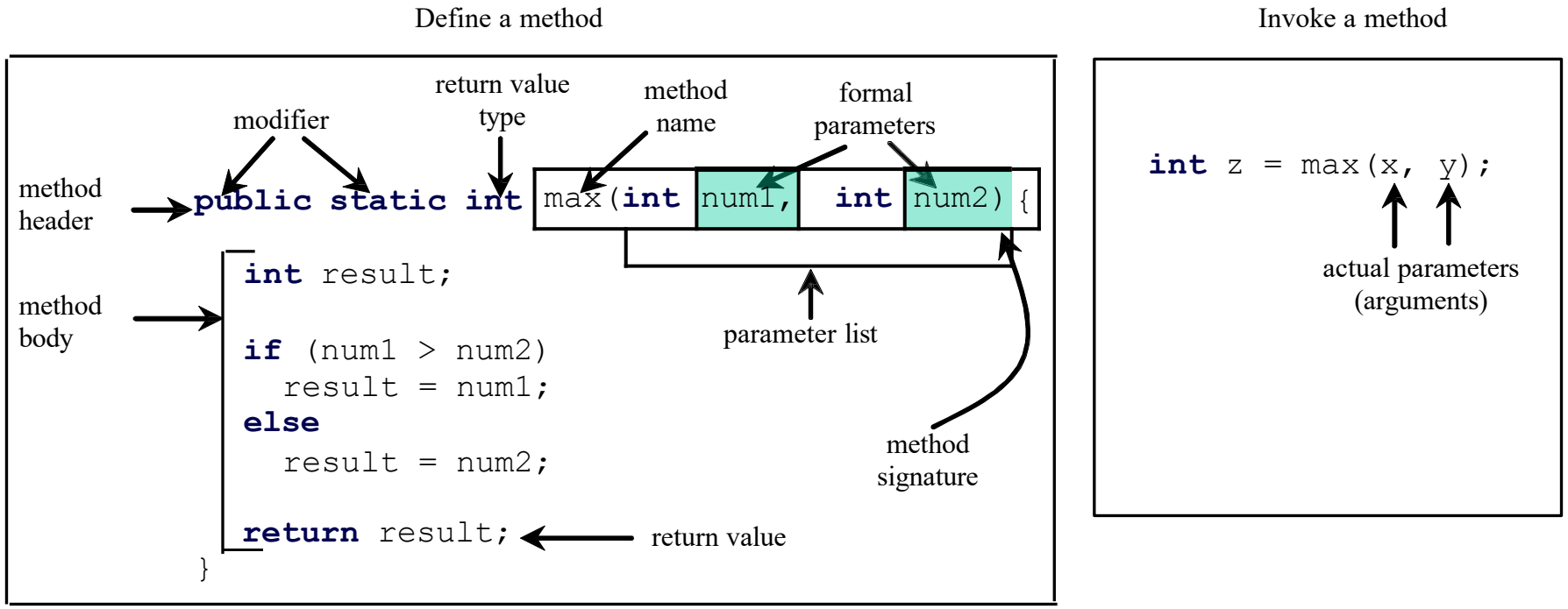
Method Signature

Method signature is the combination of the method name and the parameter list.



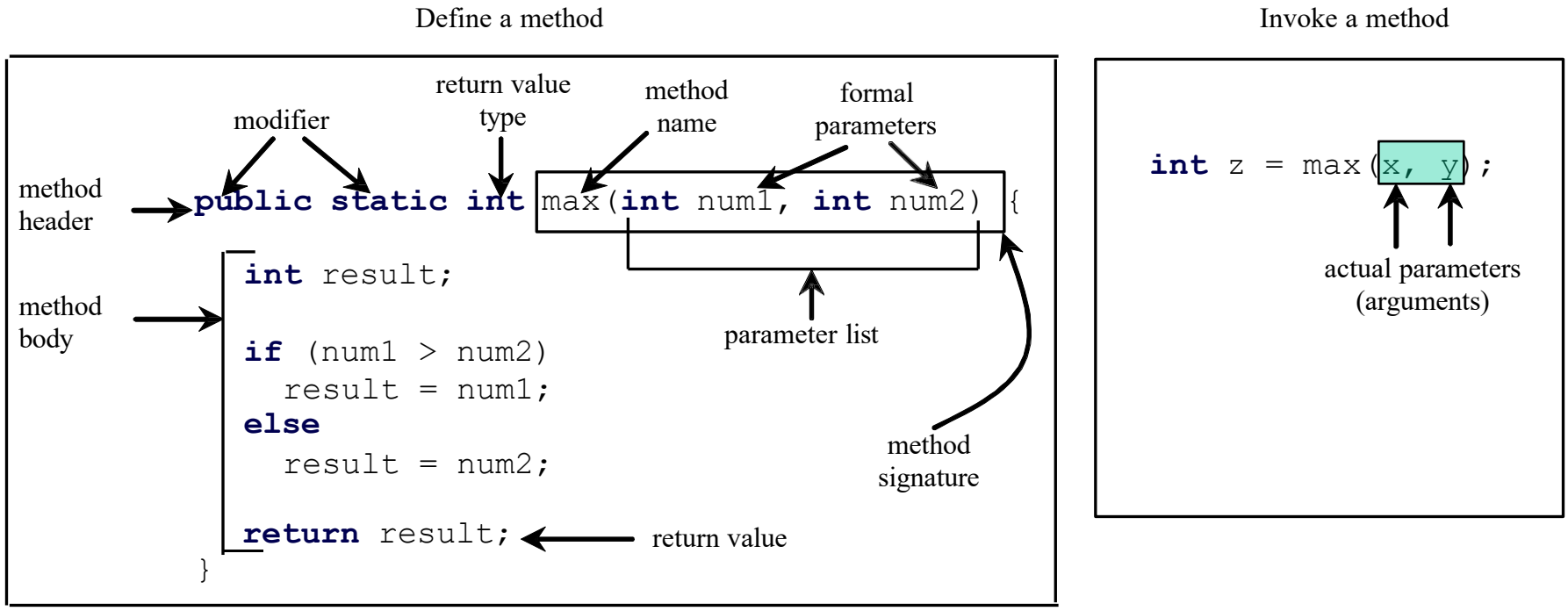
Formal Parameters

The variables defined in the method header are known as *formal parameters*.



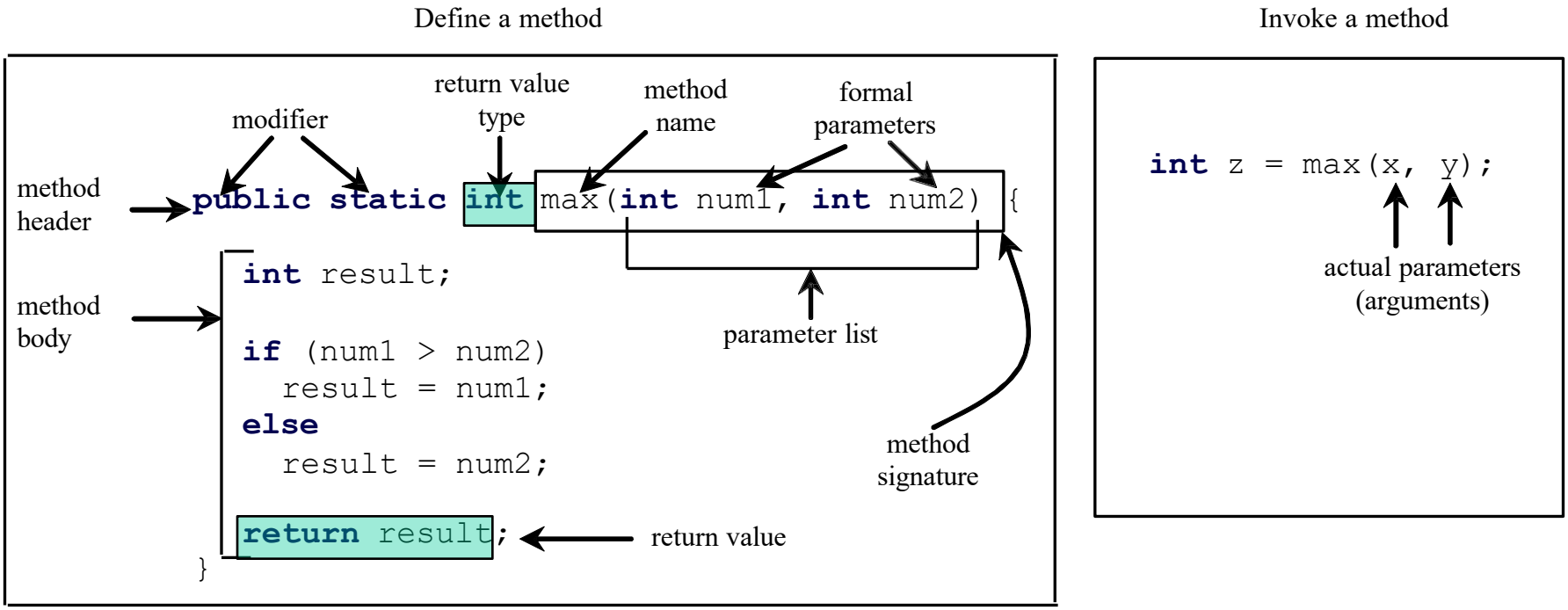
Actual Parameters

When a method is invoked, you pass a value to the parameter. This value is referred to as *actual parameter or argument*.



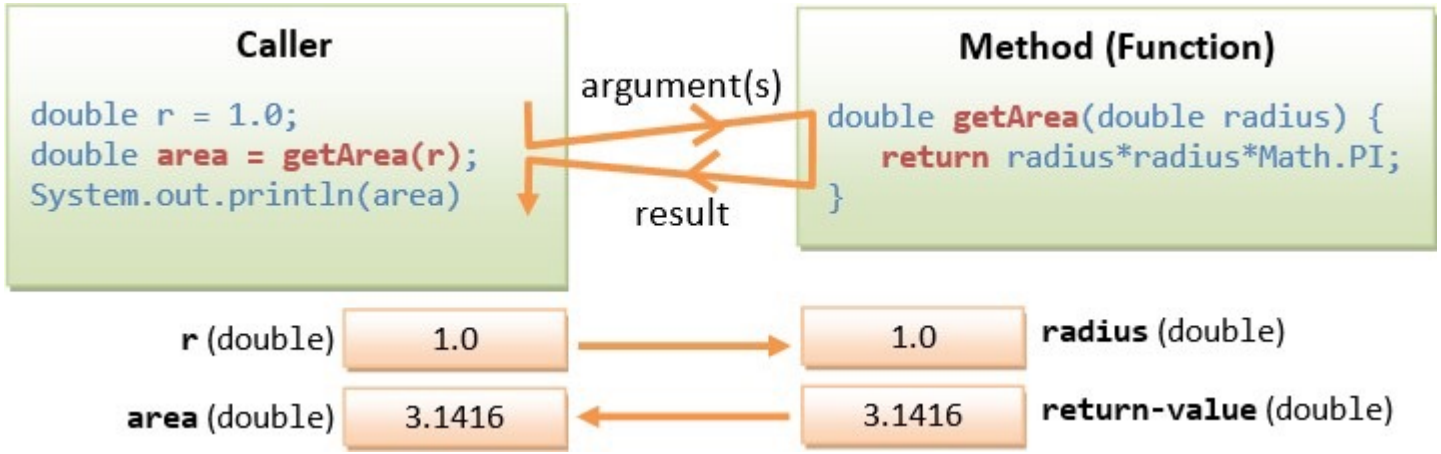
Return Value Type

A method may return a value. The returnValueType is the data type of the value the method returns. If the method does not return a value, the returnValueType is the keyword void. For example, the returnValueType in the main method is void.



Using Methods

1. A caller invokes a method and pass arguments to the method.
2. The method:
   - Receives the argument passed by the caller
   - Performs the operations in method's body
   - Return a result back to the caller
3. The caller receives the result and continues its operation



the UNIVERSITY of GREENWICH

Modifiers

The keywords **`public`**, **`private`** and **`static`** are method *modifiers.*

- **`public`** methods may be called from outside the classes in which they are defined.
- **`private`** methods may only be called from other methods defined in the same class.

the UNIVERSITY of GREENWICH

Modifiers

- `static` methods *belong to the class itself*, and not to the objects created by the class. `static` methods may be `public` or `private`.
- Non-`static` (or *dynamic*) methods have access to the inner state of class objects. (This will become clearer in the next section).

Defining a Return type

```
public static int sum(int num1, int num2)
{
 int result;
 result = num1 + num2;
 return result;
}
```

Return type

This expression must be of the same data type as the return type

The `return` statement causes the method to end execution and it returns a value back to the statement that called the method.

the UNIVERSITY of GREENWICH

`void` Methods

- A `void` method is one that simply performs a task and then terminates.

```
System.out.println("Hi!");
```

the UNIVERSITY of GREENWICH

Passing Arguments to a Method

- Values that are sent into a method are called arguments.

```
System.out.println("Hello");
number = Integer.parseInt(str);
```

- The data type of an argument in a method call must correspond to the variable declaration in the parentheses of the method declaration. The parameter is the variable that holds the value being passed into a method.

Passing a value to the `displayValue` Method

```
displayValue(5);
```

The argument 5 is copied into the parameter variable **`num`**.

```
public static void displayValue(int num)
{
     System.out.println("The value is " + num);
}
```

The method will display `The value is 5`

Passing Multiple values

The argument 5 is copied into the **num1** parameter.

The argument 10 is copied into the **num2** parameter.

```
showSum(5, 10);
```

NOTE: Order matters!

```
public static void showSum(double num1, double num2)
{
   double sum;      //to hold the sum
   sum = num1 + num2;
   System.out.println("The sum is " + sum);
}
```

Trace Method Invocation

return max(i, j) and assign the return value to k

```
public static void main(Strin    args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

the UNIVERSITY of GREENWICH

Local Variables - Scope

- A local variable is declared inside a method and is not accessible to statements outside the method.

- A method's local variables exist only while the method is executing. When the method ends, the local variables and parameter variables are destroyed and any values stored are lost.

the UNIVERSITY of GREENWICH

Java Libraries

- The Java Development Kit comes with many libraries which contain classes and methods for you to use

- These are classes and methods that other people have written to solve common tasks

- Some of these include:
  - a Math library (java.lang.Math)
  - String library (java.lang.String)
  - Graphics library (java.awt.* and javax.swing.*)
  - Networking library (java.net)

the UNIVERSITY of GREENWICH

Library methods

- Some methods include
  - `System.out.println(...);`
  - `Date now = new Date();`

```
int thisYear = now.getYear() + 1900;
```

the UNIVERSITY of GREENWICH

Library methods

- When calling a **`static`** method we always use

  `ClassName.methodName(arguments)`

```
double positiveNumber = Math.abs(-10);
double twoCubed = Math.pow(2,3);
double someTrigThingy = Math.sin(Math.PI);
```

the UNIVERSITY of GREENWICH

Library methods

- When calling a dynamic method we always use

```
object.methodName(arguments)
```

- For example:

```
object1.method1("test")
```

the UNIVERSITY of GREENWICH

Pro's and Con's of Methods

- **Advantages of Methods**
  - Code reusability
  - Reduces code duplication
  - Easier debugging
  - Problems are decomposed
  - Hides tricky logic
  - Easier to read and understand

- **Disadvantages of Methods**
  - It takes initially a little more time to set them up

File Input and Output

the UNIVERSITY of GREENWICH

Writing Text To a File

- To open a file for text output you create an instance of the `PrintWriter` class.

```
PrintWriter outputFile = new PrintWriter("StudentData.txt");
```

Pass the name of the file that you wish to open as an argument to the `PrintWriter` constructor.

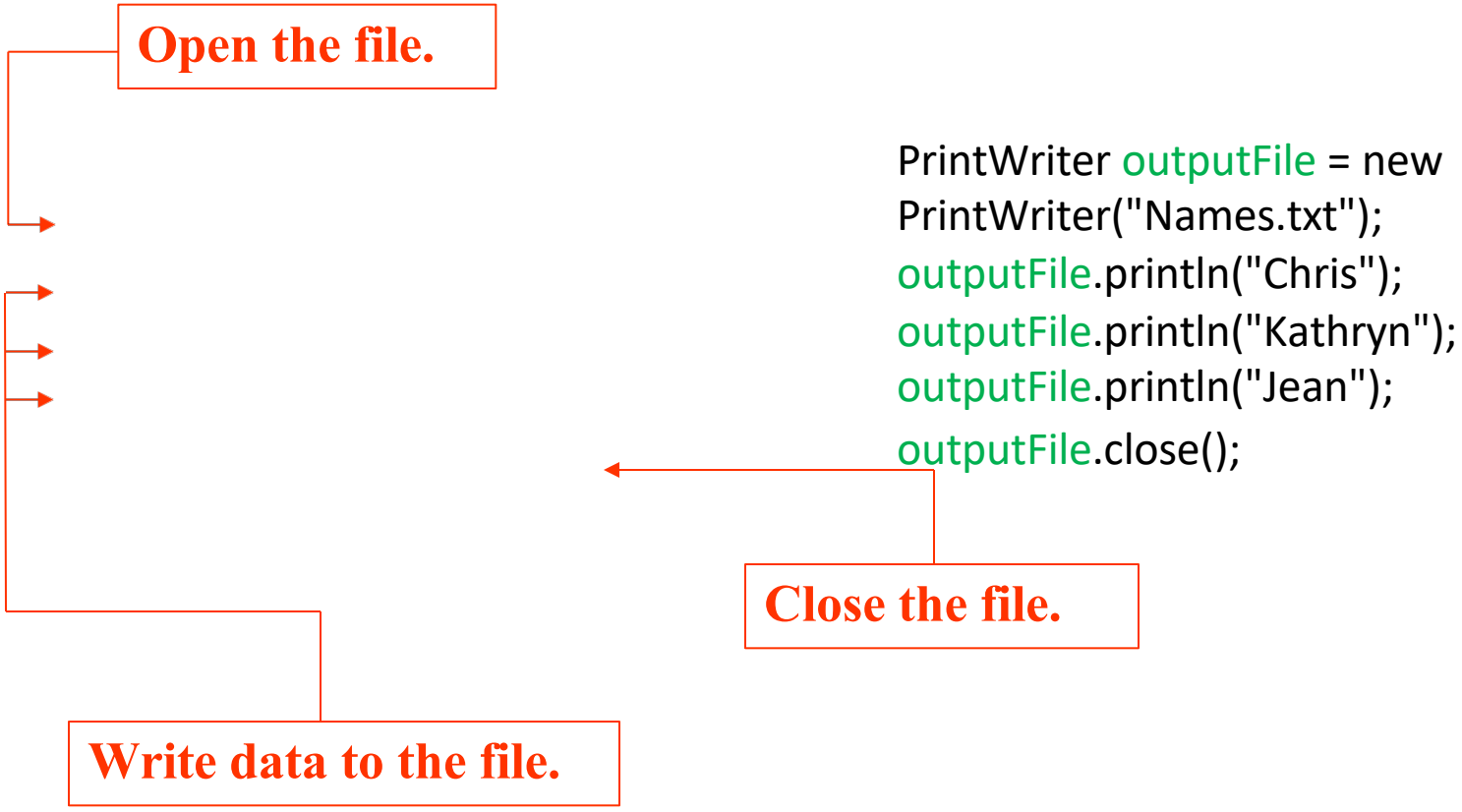
Warning: if the file already exists, it will be erased and replaced with a new file.

the UNIVERSITY of GREENWICH

The `PrintWriter` Class

- The `PrintWriter` class allows you to write data to a file using the `print` and `println` methods, as you have been using to display data on the screen.
- Just as with the `System.out` object, the `println` method of the `PrintWriter` class will place a newline character after the written data.
- The `print` method writes data without writing the newline character.

The `PrintWriter` Class



The `PrintWriter` Class

- To use the `PrintWriter` class, put the following `import` statement at the top of the source file:

```
import java.io.*;
```

- See example: FileWriteDemo.java

the UNIVERSITY of GREENWICH

Exceptions

- When something unexpected happens in a Java program, an *exception* is thrown.
- The method that is executing when the exception is thrown must either handle the exception or pass it up the line.
- Handling the exception will be discussed later.
- To pass it up the line, the method needs a `throws` clause in the method header.

the UNIVERSITY of GREENWICH

Exceptions

- To insert a `throws` clause in a method header, simply add the word *throws* and the name of the expected exception.
- `PrintWriter` objects can throw an `IOException`, so we write the `throws` clause like this:

```
public static void main(String[] args) throws IOException
```

Appending Text to a File

- To avoid erasing a file that already exists, create a `FileWriter` object in this manner:

```
FileWriter fw =
      new FileWriter("names.txt", true);//append
```

- Then, create a `PrintWriter` object in this manner:

```
PrintWriter pw = new PrintWriter(fw);
```

the UNIVERSITY of GREENWICH

Specifying a File Location

- On a Windows computer, paths contain backslash (\) characters.
- Remember, if the backslash is used in a string literal, it is the escape character so you must use two of them:

```
PrintWriter outFile =
      new PrintWriter("A:\\PriceList.txt");
```

the UNIVERSITY of GREENWICH

Specifying a File Location

- This is only necessary if the backslash is in a string literal.
- If the backslash is in a `String` object then it will be handled properly.
- Fortunately, Java allows Unix style filenames using the forward slash (`/`) to separate directories:

```
PrintWriter outFile = new
    PrintWriter("/home/rharrison/names.txt");
```

Reading Data From a File

- You use the `File` class and the `Scanner` class to read data from a file:

Pass the name of the file as an argument to the `File` class constructor.

```
File myFile = new File("Customers.txt");
Scanner inputFile = new Scanner(myFile);
```

Pass the `File` object as an argument to the `Scanner` class constructor.

the UNIVERSITY of GREENWICH

Reading Data From a File

```
Scanner keyboard = new Scanner(System.in);
System.out.print("Enter the filename: ");
String filename = keyboard.nextLine();
File file = new File(filename);
Scanner inputFile = new Scanner(file);
```

- The lines above:
  - Creates an instance of the `Scanner` class to read from the keyboard
  - Prompt the user for a filename
  - Get the filename from the user
  - Create an instance of the `File` class to represent the file
  - Create an instance of the `Scanner` class that reads from the file

Reading Data From a File

- Once an instance of `Scanner` is created, data can be read using the same methods that you have used to read keyboard input (`nextLine`, `nextInt`, `nextDouble`, etc).

```
// Open the file.
File file = new File("Names.txt");
Scanner inputFile = new Scanner(file);
// Read a line from the file.
String str = inputFile.nextLine();
// Close the file.
inputFile.close();
```

the UNIVERSITY of GREENWICH

Exceptions

- The `Scanner` class can throw an `IOException` when a `File` object is passed to its constructor.
- So, we put a `throws IOException` clause in the header of the method that instantiates the `Scanner` class.
- See Example: ReadFirstLine.java

the UNIVERSITY of GREENWICH

Summary

- We have covered :
  - Java Methods
  - Parameters, Variable, Scope
  - Library methods
  - File I/O