Methods

Camille Duquesne
1ère IB

Any questions/difficulties you would like to share about last session’s content ?

B2 Programming

B2.1.1 Construct and trace programs using a range of global and local variables of various data types.

B2.2.2 Construct programs that apply arrays and Lists.

B2.3.3 Construct programs that utilize looping structures to perform repeated actions.
• Types of loops, including counted loops and conditional loops, and appropriate use of each type
• Conditional statements within loops, using Boolean and/or relational operators to govern the loop's execution

B2.3.1 Construct programs that implement the correct sequence of code instructions to meet program objectives.
• The impact of instruction order on program functionality
• Ways to avoid errors, such as infinite loops, deadlock, incorrect output

What is a method ?

A **method/function/procedure** is a **reusable structured block of code** that fulfills a **single specific purpose**.

We have already encountered methods from the build-in library so far such as:

`System.out.println()` or `Math.pow()`

Let’s see how we can define our own methods !

What are advantages of using methods/sub procedures ?

The advantage of methods is that they are **reusable** and therefore **prevent code repetition** in our programs, as they can be reused in multiple parts of a project.

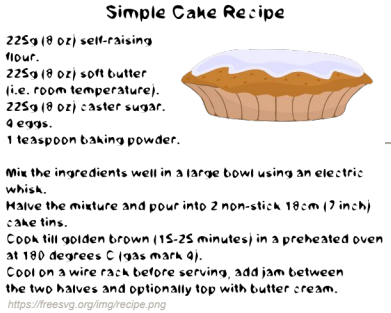
Furthermore, defining methods helps us to **organize** our program in **smaller and very modular structures**. This allows for **easier debugging, maintenance, change implementation and testing**. This is particularly useful when coding in teams as methods are (in most cases) independent structures.

How does that apply to our `Math.pow()` method ?

Defining vs Calling a method

Defining a function

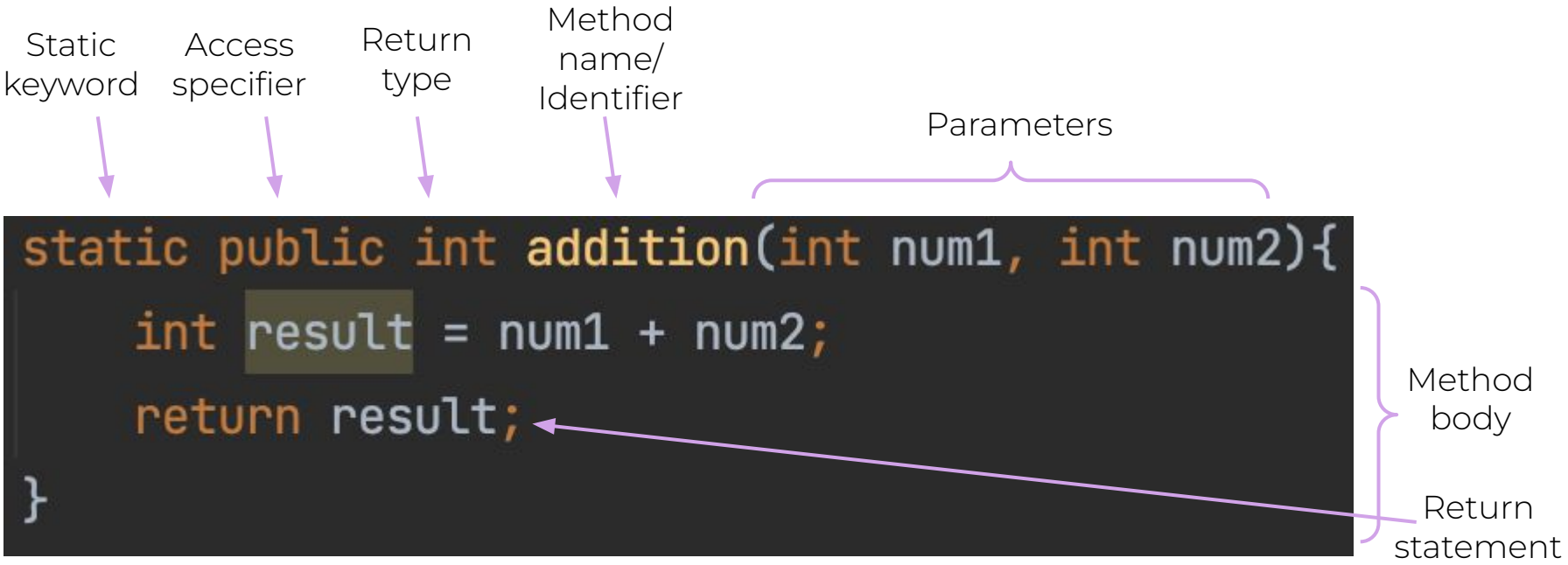
Theoretical and general definition of how it's supposed to work



Calling a function

Using the definition to really build an object.

How do we define a Static method ?



What is the method name/identifier ?

It is a **unique** name that is used to **define the method**. It must be **representative** of the functionality of the method (usually an action/verb).

A method is **called through its name**.

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

What are parameters ?

Parameters are the **input** of a function. They are written between the parentheses and separated by a comma. Always include the **data type and variable name.**

If the method has no parameter, leave the parentheses blank.

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

What is the method's body?

The **method's body** is everything between the curly braces. It contains **all the actions/statements** that will be performed to obtain the final result. It should do what the method name is telling !

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

What is return + return type ?

The **return** statement allows the function to **output** data. No return means no output !

The **return type** is a **data type** that the method **returns** (=data type of the variable after the return keyword). If the method does not return anything, we use **void** keyword.

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

Procedure vs. function/method

A function, triggers and action and returns a result

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

A procedure, triggers and action but does not return anything

```
static public void addition(int num1, int num2){
    int result = num1 + num2;
    System.out.println(result);
}
```

What is the access specifier ?

Access specifier: It specifies the **visibility** of the method.
There are four types of access specifier: public, private, protected, default.
We will only use **public** for now.

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

Public: *The method is accessible by all classes*

Private: *The method is accessible only in the classes in which it is defined.*

Protected: *The method is accessible within the same package or subclasses in a different package.*

Default: *When we do not use any access specifier is precised, protected is used.*

What is the static keyword ?

In Java, **static** means that the particular member **belongs to a class rather than to an instance of that class** (= rather than by creating a variable of that type).

Therefore you can access the member that is prefixed with a static keyword, through the name of the class.

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

Can you spot all terms ?

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

How do we call our method ?

```java
public class Main {
    public static void main(String[] args) {
        int firstAddition = Main.addition(1,2);
        System.out.println(firstAddition);
        int secondAddition = addition(14,6);
        System.out.println(secondAddition);
    }
    2 usages
    static public int addition(int num1, int num2){
        int result = num1 + num2;
        return result;
    }
}
```

Methods are called using their unique identifier (name) !

```
3
20
```

The main() method

The main() is the starting point to start the execution of a Java program. Without the main() method, Java will not execute the program.

Now we are able to understand every word of the definition of the main method:

```
public static void main(String[] args) {
    //Do Stuff
}
```

Scope of variables

The **scope** of the variable defines its **lifetime**, meaning where it can be used and where it can’t. The code block in which a variable is defined, defines its scope.

There are two possible scopes:

- Local variable
- Global variable

Local Variable

The variable **`i`** is defined in the for loop and will only exist within the loop. If I try to print **`i`** after my loop, will get an error message, as the variable **`i`** has stopped existing.

```
int[] grades = { 16,23,31,34,15 };
for (int i = 0; i < grades.length; i++) {
    System.out.println(grades[i]);
}
```

Same thing for my variable **`result`** in the second example. It only exists within the function definition of addition().

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

Global Variable

A global variable can be used **anywhere** throughout the program.
Global variables are less common in java as it is an Object Oriented Languge by design. We can still have global variables by using the **static** keyword in java.

```java
public class Main {
    2 usages
    static int a = 3;

    public static void main(String[] args) {
        System.out.println(a);
    }
    static public int addition(int num1, int num2){
        int result = num1 + num2;
        System.out.println(Main.a);
        return result;
    }
}
```

Example

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(result);
    }
    static public int addition(int num1, int num2){
        int result = num1 + num2;
        return result;
    }
}
```

Are num1 and num2 local variables or global variables ?

Why don’t we use global variables everywhere ?

In the vast majority of cases, it is a **very bad practice to use a global variable** in a function instead of passing this variable as an argument (and thus making it local).

Indeed this makes your functions **more difficult to test** and induces **more risks of bugs**. For example if your global variable changes name you would have to change all your function definitions to keep your code working. This is not the case if the variable is passed as an argument.

Therefore **you should pass as arguments the variables that you are going to use in a function**.

What are inputs ? What are outputs ?

Input is any **data/information** that a computer system (or program) **receives/takes in**. Input can come from various sources, such as user interactions, sensors, files, or other computer systems. Inputs are typically provided to perform calculations, make decisions, or generate responses.

Output is the **result/information** generated by a computer system (or program) **after transforming/processing data**. Output can be the result of transformed input and can also be used for further computation. Outputs can take various forms, including text, numbers, graphics, sound, or any other form of data representation that conveys meaningful information to the user or other parts of a computer system.

Practice !

Identify what the inputs and outputs are for a method that calculates the area of a rectangle.

Identify what the inputs and outputs are for an app that scans QR codes

Identify what the inputs and outputs are when playing Zelda breath of the wild

Can you recognize the inputs and outputs in the code:

```java
class Main {
    public static void main(String[] args) {
        double[] studentGrades = { 85.5, 90.0, 78.5, 92.0, 88.5 };
        double avg = calculateAverage(studentGrades);
        System.out.println("Average Grade: " + avg);
    }
    1 usage
    public static double calculateAverage(double[] grades) {
        double sum = 0.0;
        for (double grade : grades) {
            sum += grade;
        }
        double average = sum / grades.length;
        return average;
    }
}
```

What are the parameters ? arguments ?

Parameter: Parameters are variables defined in a method's declaration.

Arguments: Arguments are the actual values that are passed to a function or method when it is called.

What are pre-conditions of an algorithm ?

The "pre-conditions" of an algorithm refers to a **set of conditions that must be satisfied before the algorithm can be executed.** Pre-conditions ensure that the input is suitable for the algorithm to operate correctly and produce meaningful results.

If a precondition is not met, it is the responsibility of the the programmer to ensure that the algorithm is not invoked with invalid input.

Preconditions are typically documented as part of the algorithm's specification or usage guidelines.

Practice !

Identify what the pre-conditions are for a method that calculates the area of a rectangle.

Identify what the pre-conditions are for an app that scans QR codes

Identify what the pre-conditions are when playing Zelda breath of the wild

What are the preconditions in the following method ?

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns a string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

Parameters:

`beginIndex` - the beginning index, inclusive.

`endIndex` - the ending index, exclusive.

Returns:

the specified substring.

Throws:

`IndexOutOfBoundsException` - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

What are post conditions of an algorithm ?

Post-conditions refer to the **conditions** that are expected to **hold true after the algorithm has been executed**. They specify the desired outcomes and properties of the algorithm's results.

Post-conditions are essential for verifying the correctness and validity of the algorithm's execution.

Practice !

Identify what the post-conditions are for a method that calculates the area of a rectangle.

Identify what the post-conditions are for an app that scans QR codes

Identify what the post-conditions are when moving your character in a video game

What are the post-conditions of the following method ?

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns a string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

Parameters:

`beginIndex` - the beginning index, inclusive.

`endIndex` - the ending index, exclusive.

Returns:

the specified substring.

Throws:

`IndexOutOfBoundsException` - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

Python/Java comparison

What is the advantage of java compared to python in regards to pre-conditions and post-conditions ?

Clean code & good practices

In general it is important to keep in mind for whom we write our code. The main ones concerned will be our users, our colleagues and ourselves. Moreover, if your code is published online, anyone can read it. **Your code is meant to be read and reread** and for your code to last in time it is very important that it is readable.

To name **your variables, methods, classes and packages correctly, they must be named after what they represent, their value or their purpose.**

What does this function do ?

```
static public int stuff(int a, int[] b){
    int c = 0;
    for (int i = 0; i < b.length; i++) {
        if (a == b[i]) {
            c = c + 1;
        }
    }
    return c;
}
```

What does this function do ?

```
static public int countCoins(int specificCoin, int[] walletChange){
    int numCoins = 0;
    for (int coin: walletChange){
        if (coin == specificCoin) {
            numCoins += 1;
        }
    }
    return numCoins;
}
```

Nested functions

We are also allowed to make a call to a method inside the definition of another method.

However definitions of methods should remain on the same level and not be nested.

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(half(2,3));
    }
    1 usage
    static public int addition(int num1, int num2){
        int result = num1 + num2;
        return result;
    }
    1 usage
    static public double half(int num1, int num2) {
        double result = 0.5 * addition(num1, num2);
        return result;
    }
}
```

Modularization

Functions are a great way to provide modularity in system development. **Modularity** refers to the act of designing your system in **separate independent modules**. Each module handles a **specific functionality** and modules work cohesively when combined

Modularity is very advantageous as it **reduces development time.** Since the system is designed/organized more logically, developers can work on each module independently and modules can be developed in **parallel**.

Each module can also be independently **tested** and allows to catch bugs early on.

For the same reasons using modularity makes the code more **maintainable** overtime. Indeed if changes are needed, they can be made to individual modules without impacting the entire system.

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        calculus(2,3);
    }

    1 usage
    static public int calculus(int num1, int num2) {
        return num1 * num2 + num1;
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(calculus(2,3));
    }

    1 usage
    static public int calculus(int num1, int num2) {
        return num1 * num2 + num1;
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        int result = calculus(2,3);
        System.out.println(result);
    }

    1 usage
    static public int calculus(int num1, int num2) {
        System.out.println("hello");
        return num1 * num2 + num1;
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        int result = calculus(2,3);
        //System.out.println(result);
    }

    1 usage
    static public int calculus(int num1, int num2) {
        System.out.println("hello");
        return num1 * num2 + num1;
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        //int result = calculus(2,3);
        //System.out.println(result);
    }

    static public int calculus(int num1, int num2) {
        System.out.println("hello");
        return num1 * num2 + num1;
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        int result = calculus(2,3);
        System.out.println(result);
    }

    1 usage
    static public double calculus(double num1, double num2) {
        return num1 * num2 + num1;
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        int result = calculus(2,3);
        System.out.println(result);
    }

    1 usage
    static public void calculus(double num1, double num2) {
        System.out.println(num1 * num2 + num1);
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        calculus(2.0, 4.0);
    }

    1 usage
    static public void calculus(double num1, double num2) {
        System.out.println(num1 * num2 + num1);
    }

}
```

What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        int a = 1;
        calculus(2.0, 4.0);
        System.out.println(a);
    }

    1 usage
    static public void calculus(double num1, double num2) {
        int a = 2;
        System.out.println(num1 * num2 + num1);
    }

}
```

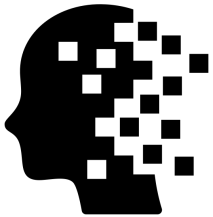
What does this code output ?

```java
public class Main {
    public static void main(String[] args) {
        int a = 1;
        calculus(2.0, 4.0);
    }

    1 usage
    static public void calculus(double num1, double num2) {
        System.out.println(num1 * num2 + num1);
        System.out.println(a);
    }

}
```

Pause & Recall



Created by Victoruler
from Noun Project

Close your eyes and try to recall as many things as possible that were covered during this lesson.

Alternatively, you can keep your eyes open and write down as many things you remember on a piece of paper.

This will help strengthen your memory of key concepts 💪

Exercise 1 *(GenAI 🟠 Orange)*

Write a method calculateArea() that takes the length and width of a rectangle as parameters and returns its area. In the main method, take user input for the length and width, call the calculateArea method, and display the result.

Exercise 2 *(GenAI 🟠 Orange)*

Write a function that adds all the numbers in an array and returns the sum

Exercise 3 *(GenAI 🟠 Orange)*

Write a function that takes a name as argument and prints "Hello" + your name. No return statement should be used

Exercise 4 *(GenAI 🟠 Orange)*

Write a function to find the maximum and minimum value of an array.

Exercise 5 *(GenAI 🟠 Orange)*

Write a function that takes as input from the user: a String and a number n. The script should return the String with the n^{th} character removed from the string.

Exercise 6 *(GenAI 🟠 Orange)*

Write a function that performs the sum of two 2D matrices of the same size.

Exercise 7 *(GenAI 🟠Orange)*

Recode the String function replace()
https://docs.oracle.com/en/java/javase/24/docs/api/java.base/java/lang/String.html#replace(char,char)

Then code an alternative function called replaceAll() that replaces all occurrence of the char you want to replace

What are the main differences in your algorithmic logic between both functions ?

Exercise 8 *(GenAI 🟠 Orange)*

Recode the String function substring()

https://docs.oracle.com/en/java/javase/24/docs/api/java.base/java/lang/String.html#substring(int,int)

Exercise 9 *(GenAI 🟠 Orange)*

Create a method reverseString() that takes a string as a parameter and returns its reverse.

Exercise 10 *(GenAI 🟠 Orange)*

Write a function that generates a random password. The password must have a random length between 7 and 10 characters. Each character must be chosen at random and should have at least one lowercase, one uppercase, one digit and one special character character. Your function will not take any parameters. It will return the randomly generated password as the only result.

Homework

Finish all the exercises

Create flashcards for the following terms:

Procedure
Function
Maintainable code
Modularity
Parameters
Arguments
Return
Scope of variables: local, global
Input
Output
Return type
Method identifier