Methods/Functions

Methods/Functions

- A **method/function** is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- Methods are used to perform certain actions, and they are also known as **functions**.

Why use methods or functions?

- To reuse code: define the code once, and use it many times.

Java Methods

- A method must be declared within a class.
- It is defined with the name of the method, followed by parentheses **()**.
- Java provides some pre-defined methods.

Method.java > ...

```java
public class Method {
    // Method definition
    static void printHello() {
        System.out.println(x:"Hello, World!");
    }

    Run | Debug
    public static void main(String[] args) {
        // Calling the method
        printHello();
    }
}
```

- printHello() is the name of the method
- static means that the method belongs to the Main class and not an object of the Main class. You will learn more about objects and how to access methods through objects later in this tutorial.
- void means that this method does not have a return value. You will learn more about return values later in this chapter

Python Function

```python
# Function definition
def print_hello():
    print("Hello, World!")

# Calling the function
print_hello()
```

C++ Function

Function.cpp

Function.cpp > ...

```cpp
#include <iostream>
using namespace std;

// Function declaration
void printHello() {
    cout << "Hello, World!" << endl;
}

int main() {
    // Calling the function
    printHello();
    return 0;
}
```

- printHello() is the name of the function
- void means that the function does not have a return value. You will learn more about return values later in the next chapter
- inside the function (the body), add code that defines what the function should do

Parameters and Arguments

- Information can be passed to methods as parameter. Parameters act as variables inside the method.
- Parameters are specified after the method name, inside the parentheses.
- You can add as many parameters as you want, just separate them with a comma.

Java

```java
public class Method {
    // Method definition with parameters
    static void printMessage(String message, int x) {
        System.out.println(message + x);
    }

    Run | Debug
    public static void main(String[] args) {
        // Calling the method with an argument
        printMessage(message:"Hello, World!", x:5);
    }
}
```

Python

```python
# Function definition with parameters
def print_message(message,x):
    print(message,x)

# Calling the function with an argument
print_message("Hello, World!",5)
```

C++

```cpp
#include <iostream>
#include <string>
using namespace std;

// Function definition with parameters
void printMessage(string message, int x) {
    cout << message << x << endl;
}

int main() {
    // Calling the function with an argument
    printMessage("Hello, World!", 5);
    return 0;
}
```

Task….

- Create a method/function named calculateRectangleArea that takes two parameters, length and width.
- Create a method/function named printGreeting that takes a parameter name. Print a personalized greeting message, such as "Hello, [name]!".
- Create a method/function named findMax that takes three parameters, num1, num2, and num3.
- Create a method/function named concatenateStrings that takes two parameters, str1 and str2.