Lesson:

Java Methods

PW SKILLS

Pre-Requisites

- Basic java syntax
- Concept of java class and objects.

List of Concepts Involved

- Introduction to methods
- Why are methods important in Java?
- How to declare methods
- Calling a method

Java is a language that promotes and supports the use of methods as far as possible to implement logic/solve a problem.

Topic : Methods in java

A method is a block of code /collection of statements/ a set of code, grouped together to perform a certain task or operation.

- A method runs only when it's called.
- We can pass data, as parameters, into a method (we will discuss more about it in the lecture)
- Methods are also known as functions.

Topic : Why are methods important in Java?

You might be wondering why do we need methods when we were already implementing our logic in a single simple program. Well, we have the answer!
Methods are important because:

- Methods allow and enable us to write code once and use it as many times as we want which means we are not required to write the same code again and again.
- Methods are often called time savers because they help you to reuse the existing code hence saving a lot of your time in re-typing at multiple places.
- A code created with a method is easy to read and dry run since the code in the method operates separately and can be called whenever needed.

Note: Method and function are the same; **'method**' is the object-oriented term for **'function**'.
Method, like a function, is a collection of instructions that perform a task, but a method is associated with object(s) but function is not.

For example:
Suppose you have to write a program that creates a rectangle and colors it. This can be solved by creating two methods

- a method to draw the rectangle
- a method to color the rectangle

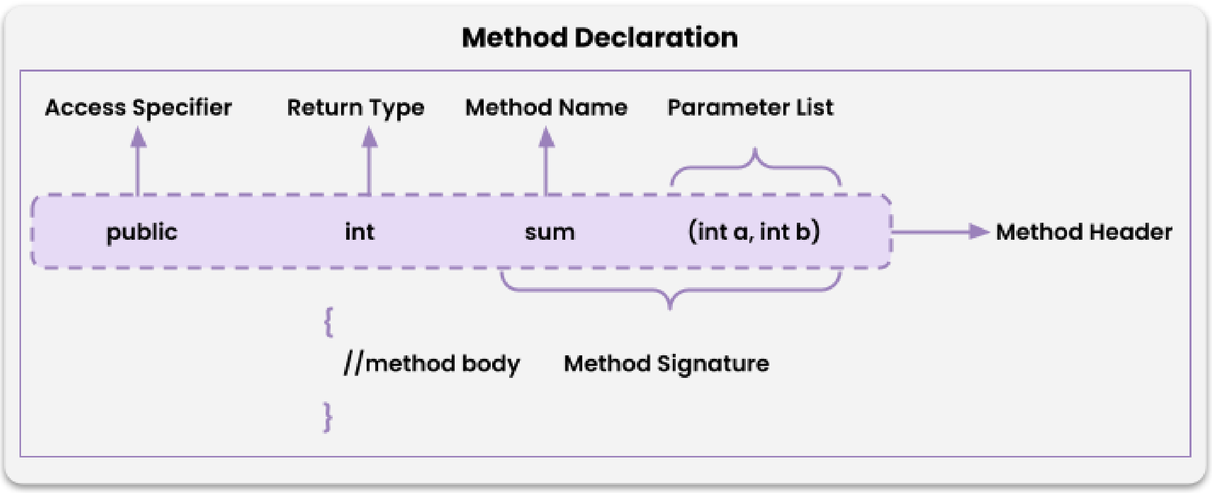
Dividing a complex problem into smaller chunks makes your program simple to understand and easy to reuse. In Java, there are two types of methods:

- **User-defined Methods:** These are the methods that you create based on the requirements.
- **Standard Library Methods:** These are built-in methods in Java that are readily available to use.

We would be using both of these in the topics and lectures ahead.

Topic: How to declare Methods

Method declaration provides information about the attributes, such as visibility, return type, name, and arguments of the method. It mainly consists of six components (in a method header), as shown in the following figure.



Let's discuss each of these in brief:

1. Method Signature:
In Java, a method signature is a part of the method declaration. It's a combination of the method name and parameter list.

2. Access Specifier:
It is the access type of the method. It specifies the visibility and accessibility of the method. You will have a clear idea once we discuss it with a program. But before that, let us explore the available options for access specifiers. In java, there are four types of access specifiers:

i) Private : The methods and/or data members declared as private are accessible only within the class where they are declared.
ii) Public : The methods that are declared public are accessible from everywhere in the program. There is no restriction on the scope of public data members.
iii) Protected : The methods declared as protected are accessible within the same package or subclasses in different packages.
iv) Default : When no access modifier is specified for a method – It is said to be having the default access modifier by default. Default access modifiers are accessible only within the same package.

That was too much of information to process, isn’t it ? But what if we derive analogies with something which we explore on a regular basis. How about social media ?! Facebook ?
That would make you learn the concept of public, private, potected in a better way.

So, let us understand this concept by a real life example of FACEBOOK post.

Suppose, you plan to upload a Facebook Status. Have you ever noticed, you can do it in 4 possible ways :

a. If you make your status visible for “Public”, anyone and everyone can see this status all across the facebook. This draws a parallel with ***Public Access Specifiers*** that we have just learnt.
b. If you make this status visible "only for me”, the status will only be visible in and from your account. This is similar to ***Private Access Specifiers.***
c. If you make this status visible for “Friends or Friends of friends”, your status will be only visible to your friends and your friend’s immediate friends which means although many people can see the status now but still not everyone present on Facebook will be able to view it. They atleast need to have a common connection between them and you. This is similar to the behaviour of ***Protected Access Specifiers***.
d. If you make this status visible for "Friends", then your status will be only visible to your immediate friends. It will not be visible to your friend’s friend or everyone signed up on Facebook.This is similar to the working of ***Default Access Specifiers***.

So, it all made sense now, right ?!

3.Return type : It is used to specify the data type of the value returned from a method.

4.Method Name : Every method has a name and is invoked/called by it. Name should be given in such a way that it corresponds to the functionality of the method. Eg. add () method should have addition functionality.

5.Parameters : Parameters act as normal variables inside the method. They are specified after the method name, inside the parentheses. You may add as many parameters as you want as per requirement, syntactically, you just have to separate them with a comma.

6.Method Body: It contains all the actions to be performed i.e. the code for which method has been made. It is enclosed within the pair of curly braces.

Creating a method alone won’t help, we also have to make sure that it is being called at the required places.
But how do we call a method ?!

Topic: Calling a Method

- To call a method in Java, you have to write the method name followed by two parentheses () and a semicolon; . Here, if there are any values to be passed to the method, the parameters will have to be passed within these parantheses.
- To call the method you have to know the name of the method, number of parameters required, their data types and the return type of the method to collect the returned value by the method.

In the following example, welcome() method is used to print a text (the action), when it is called:

Example 1
```
public class Main {
  static  void welcome() {
    System.out.println("Welcome to Physics Wallah");
  }

  public static void main(String[] args) {
    welcome();
  }
}
```

Output: Welcome to Physics Wallah

Note: The keyword static means that the particular method/attribute can be accessed without creating an object of a class.We will learn about static keyword in upcoming lectures.
Explanation:
In the example above, we have created a method welcome(). It takes no parameters.
In the line,
`welcome();`
we have called method welcome() by passing no arguments . Since, the method is not returning any value, it's return type is void.

Example 2. Java Program to add two numbers using methods
```
class Main {

  // create a method
  public int addition(int p, int q) {
    int add = p + q;
    // return value
    return add;
  }

  public static void main(String[] args) {

    int p = 40;
    int q = 60;

    // create an object of Main
    Main obj = new Main();
    // calling method
    int answer = obj.addition(p,q);
    System.out.println("Sum is: " + answer);
  }
}
```

Output: Sum is 100

Explanation:
In the example above, we have created a method addition(). The method takes two parameters p and q. We have also created an object “obj” of class Main.

In the line `Main obj = new Main();`

Every time an object is created using a **new**() keyword, at least one constructor (it could be the default constructor) is invoked to assign initial values to the **data members** of the same class. (We will study constructors in detail in the forthcoming lectures.)
In the line,

`int answer = obj.addition(p, q);`

Here, we have called the method by passing two arguments p and q. Since, the method is returning integer value, we have stored the value in the answer variable of int type.
Note- If the method does not return any value, we use the void keyword as the return type of the method.

Example 3.
```
   class Main {

  // create a method
  public void addition(int p, int q) {
    int add = p + q;
    // print value
    System.out.println("Sum is: " + add);
  }

  public static void main(String[] args) {

    int p = 40;
    int q = 60;

    // create an object of Main
    Main obj = new Main();
    // calling method
    obj.addition(p,q);

  }
}
```

Output: Sum is 100
Explanation:
In the example above, we have created a method named addition(). The method takes two parameters p and q.

In the line, `obj.addition(p, q);`

we have called the method by passing two arguments p and q. Since, the addition method is not returning any value the return type is void.

Standard Library Methods
The standard library methods are built-in methods in Java that are readily available for use. These standard libraries come along with the Java Class Library (JCL) in a Java archive (*.jar) file with JVM and JRE.

For example,

- print() is a method of java.io.PrintSteam. The print("...") method prints the string inside quotation marks.
- sqrt() is a method of Math class. It returns the square root of a number.
- floor() is a method of Math class. It returns the rounded-off nearest integer which is less than the given value.

Here's an example:

```
public class Main {
  public static void main(String[] args) {
    // using the sqrt() method
    System.out.print("Square root of 16 is: " + Math.sqrt(16));
    }
}
```

Output:

```
Square root of 16 is: 4.0
```

You will come to know about more of these standard library functions once we start with problem solving and explore more questions.

That is all for the introduction of methods. We will learn more in the forthcoming lectures !
Till then,keep learning !!

Upcoming Class Teasers

- Java Methods
- Concept of scope of variables