What is a method?

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

```java
public int methodName(int a, int b) {
    // body
}
```

- method definition
  - `public` - modifier
  - `int` - return type
  - `methodName` - Name of the method
  - `int a, int b` – list of parameters
  - `// body` - method body

Method Example

Here is the method takes two parameters num1 and num2 and returns the minimum between the two.

```java
/** the snippet returns the minimum between two numbers */

public int minFunction(int n1, int n2) {
   int min;
   if (n1 > n2)
      min = n2;
   else
      min = n1;

   return min;
}
```

Method Calling

A method should be called to make use of it. There are two ways in which a method is called i.e., method returns a value or returning nothing (no return value).

The methods returning void is considered as call to a statement. Lets consider an example –

```
System.out.println("This is tutorialspoint.com!");
```

The method returning value can be understood by the following example –

```
int result = sum(6, 9);
```

Method Calling Example

```java
public class ExampleMinNumber {
   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = minFunction(a, b);
      System.out.println("Minimum Value = " + c);
   }
   /** returns the minimum of two numbers */
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;

      return min;
   }
}
```

This will produce the following result – `6`

The void Keyword

The void keyword allows us to create methods which do not return a value.

```
public class ExampleVoid {

    public static void main(String[] args) {
        methodRankPoints(255.7);
    }
    public static void methodRankPoints(double points) {
        if (points >= 202.5) {
            System.out.println("Rank:A1");
        }else if (points >= 122.4) {
            System.out.println("Rank:A2");
        }else {
            System.out.println("Rank:A3");
        }
    }
}
```

Encapsulation

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java −

- Declare the variables of a class as private.
- Provide public setter and getter methods to modify and view the variables values

Encapsulation Example

```java
public class EncapsulateMe {
   private String name;
   private int age;

   public int getAge() {
      return age;
   }

   public void setAge(int age) {
      this.age = age;
   }

   public String getName() {
      return name;
   }

   public void setName() {
      this.name = name;
   }
}
```

Benefits of Encapsulation

- The fields of a class can be made read-only or write-only.
- A class can have total control over what is stored in its fields.
- The users of a class do not know how the class stores its data. A class can change the data type of a field and users of the class do not need to change any of their code.

Does Java pass by reference or by value?

Java passes **everything** *by value*, and not *by reference* – make sure you remember that. And when we say everything, we mean everything – objects, arrays (which are objects in Java), primitive types (like ints and floats), etc. – these are **all passed by value** in Java.

The key with pass by value is that the method will not receive the actual variable that is being passed – but just a copy of the value being stored inside the variable.

Example of pass by value in Java

Suppose we have a method that is named “receiving” and it expects an integer to be passed to it:

```
  public static void receiving (int var) {
                 var = var + 2;
  }
```

What would be the output of this code :

```
public static void main(String [] args)
{
  int passing = 3;

  receiving (passing);

  System.out.println("The value of passing is: " + passing);
}
```

Are objects passed by reference in Java?

Everything is passed by value in Java. So, objects are not passed by reference in Java.

Let's be a little bit more specific by what we mean here:

Objects are passed by reference – meaning that a reference/memory address is passed when an object is assigned to another. – BUT (and this is what's important) that reference is actually passed by value. The reference is passed by value because a copy of the reference value is created and passed into the other object

Confused Yet

```
//create an object by passing in a name and age:
Person variable1 = new Person("Mary", 32);

Person variable2;

// Both variable2 and variable1 now both name the same object
variable2 = variable1;

/*this also changes variable1, since variable2 and variable1
   name the same exact object: */

variable2.setName("Jack");
variable2.setAge(22);

System.out.println(variable1.getName()+""+variable1.getAge();
```

Reference list

1. https://docs.oracle.com/javase/tutorial/java/javaOO/index.html
2. http://web.mit.edu/1.00/www/definitions.htm
3. https://www.tutorialspoint.com/java/java_methods.htm