In Java method overloading refers to the ability for a class to have more than one method with the same name as long as the arguments are not the same. There are 3 major ways to utilize method overloading and they are by changing the number of parameters, changing the data types of the parameters and lastly changing the sequence of the data type parameters. An example of each is listed below:

**NUMBER OF PARAMETERS:**

add(int, int)

add(int, inti, int)

**DATA TYPE:**

add (int, int)

add(int, float)

**SEQUENCE OF DATA TYPES OF PARAMTERS:**

add(int, float)

add(float, int)

When method overloading it is extremely important to make sure that the casting of the data types is correct. The major benefits to method overloading is the flexibility that it provides by allowing the use of one method with multiple or varying data types. This can create a more seamless code base as one might see in a program that uses mathematical methods such as adding or multiplying. It allows for the method to be able to possibly take in multiple additional parameters without having to build an entirely new addition method.

The way to think about void vs non-void methods is to look at it as void methods don’t return a value back to the method. Hence the term void. This is different in the case of non-void methods which do return a value to the method itself. There are many non-void methods. It is also very important to remember that the return value matches the return type of the method.

A method can return a **literal value:**

return 500;

return true;

A method can return the contents of a **variable:**

return sum;

return name;

A method can return the results of a **mathematical expression:**

return 20 + 5 + 10;

return (num1 + num2) / 2;

A method can return the result of a **boolean expression**:

return num > num1;

return num != 0 && num < 100;