Methods

By: Jason, Moh, Benjamin, Gordon, & Matthew

What are methods

A java method can be interpreted as a subprogram. It is a collection of statements that are grouped together to perform an operation.

Built-in vs User-defined Methods

Built-in:

Build-in methods are part of the compiler package, such as System.out.println ( ) and System.exit(0).

User-defined:

User-defined methods are created by you, the programmer. These methods take-on names that you assign to them and perform tasks that you create

Types of methods

Function(return)-Type: it calculates and return a value

```
Public static int calculate(int number){
```

Procedure-Type: executes some commands.

```
Public static void displayReverse(){
```

Example (syntax format)

```
Function(return)-type:

Public static return-type method-name(parameter 1){



Procedure-type method:

Public static void method-name(parameter 1){
```

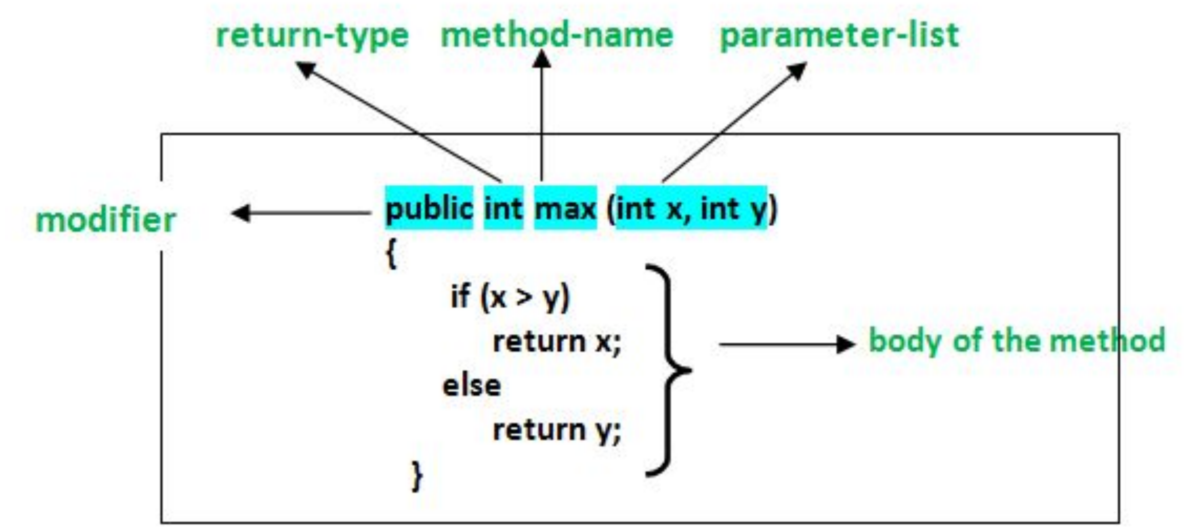
How to create a method (Method declaration)

In general, method declarations has five basic components (figure 2.) :

- **Modifier**-: Defines **access type** of the method i.e. from where it can be accessed in your application (For example: public).
- **The return type** : The data type of the value returned by the method or void if it does not return a value(Procedure and function type).
- **Method Name** : A specific name that identifies the method that can be used to invoke it later .
- **Parameter list** : Comma separated list of the input parameters are defined, preceded with their data type, within the enclosed parentheses. If there are no parameters, you must use empty parentheses ().

Method declaration continued

- **Method body :** It is enclosed between braces. The code that you need to be executed to perform your intended operations.



```
public int max (int x, int y)
{
      if (x > y)
          return x;
      else
          return y;
 }
```

Figure 2.

How to call a method (method invocation)

To invoke a method you need the method name with the parameter list defined between parentheses.

Example:

```
MethodName(parameter list);
```

Must have () (parentheses) when calling a method, even with no passing parameters.

Pass-By-Value

What happens: When a method is called, a copy of the value of each argument is passed to the method

In the second method: This copy can be changed inside the method, however such a change has no effect on the actual argument.

Pass-By-Value Continued

Main Method

```
int num = 10;
double decimal = 5.2;
NumberManeuvers(num, decimal);
System.out.println(“num = “ + num + “ and decimal = “ + decimal);
```

NumberManeuvers Method

```
public static void NumberManeuvers(int i, double j) {
    if (i == 10) {
       j = 6.2;
       i = 12;
    }
}
Output:     num = 10   and  decimal = 5.2
```

Pass-By-Reference

What happens: When an object (Array, String in arrays) is passed to a method, its memory location address (reference point) is used

The object: Arrays & Strings behave like objects

In the second method: When their memory location is passed to the method the object can be manipulated in the method resulting in actual changes to the object(Array, String)

Pass-By-Reference Continued

Main Method

```
int[] num = {1, 2, 3};
testingArray(num);
System.out.println(“num[0] = “ + num[0] + “, num[1] = “ + num[1] + “,
num[2] = “ + num[2]);
```

testingArray Method

```
public static void testingArray(int [] value) {
   value[0] = 4;
   value[1] = 5;
   value[2] = 6:
}
```

Benefits to methods

There are many advantages of using methods. Some of them are listed below:

- It makes the program well structured.
- Methods enhance the readability of the code.
- It provides an effective way for the user to reuse the existing code.
- Allows for easier debugging.
- Divide and conquer.
- Certain solutions require the use of methods.

Thank you for listening.