# School Grades

School teachers of grades 1 - 9 have to prepare their students' annual progress and achievement reports.

Write a program that helps teachers accomplish this task.

# Grades in School

```java
public class GradeIAverage {
    Run|Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int marksOfSubject = 0;
        int averageMarks = 0;
        int totalMarks = 0;
        // Enter the marks of each student in the semester
        System.out.println("Enter the marks of 5 subjects in semester 1 : ");
        for(int i = 0;i<5;i++)
        {
            marksOfSubject = sc.nextInt();
            //Marks in each subject must not be less than 0 or greater than 100
            if(marksOfSubject<0 || marksOfSubject>100)
            {
                System.out.println("Marks is less than 0 or greater than 100, Enter the mark again");
                marksOfSubject = sc.nextInt();
                continue;
            }

            // calculate the total marks
            totalMarks = totalMarks + marksOfSubject;
        }
        // calculate the average marks
        averageMarks = totalMarks/5;
        System.out.println("The total marks is : "+totalMarks);
        System.out.println("The average marks are : " + averageMarks);
        // Categorize the student based on grade obtained
        if(averageMarks>80)
            System.out.println("Grade is A");
        else if(averageMarks>=79 || averageMarks<=60)
            System.out.println("Grade is B");
        else if(averageMarks>=59 || averageMarks<=50)
            System.out.println("Grade is C");
        else if(averageMarks>=49 && averageMarks<=35)
            System.out.println("Grade is D");
        else
            System.out.println("Grade is F");

    }
}
```

## Think and Tell

- How can this code be optimized?
- Can we separate its functionality?
- Is it possible to reuse this code for another application?

# Implementing Modular Programming Using Functions

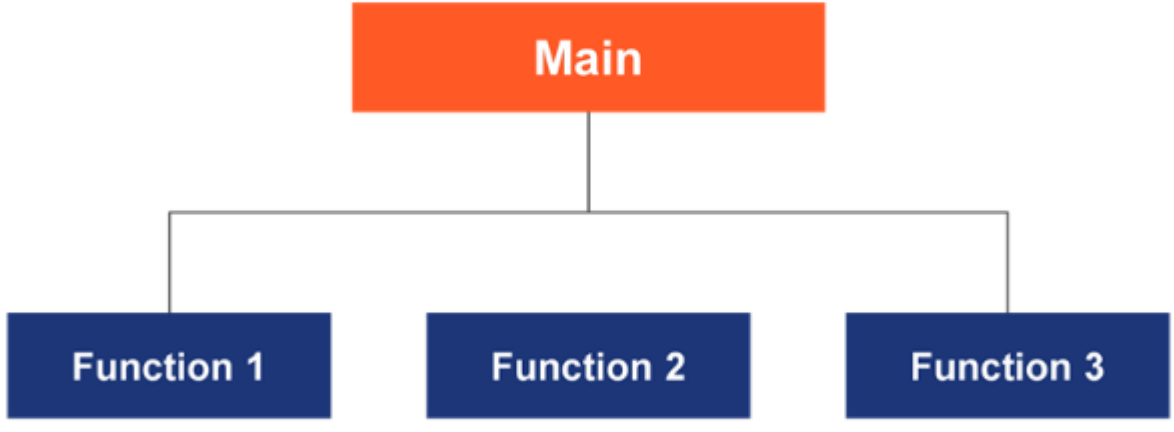
# Learning Objective

- Describe Functions
- Describe Method Structure
- Explain Method Call
- Local and Instance Variable

# Functions

# What Is Modular Programming?

Modular programming is about dividing a program into small, independent modules which have some specific functionality, instead of writing a program as one large block of code.



# What Is a Function?

- A function is a block of organized, reusable code that is used to perform a specific action.
- It has multiple lines of code inside.
- A function takes input parameters, processes them, and produces an output.


- **Note : A function is called a method in Java.**

# How Does a Function Process?

- Add two numbers, num1 and num2 and display the final output as a result.
- Num1 and num2 are input parameters for the function and are of type integer.
- The result is the output parameter of the type integer.

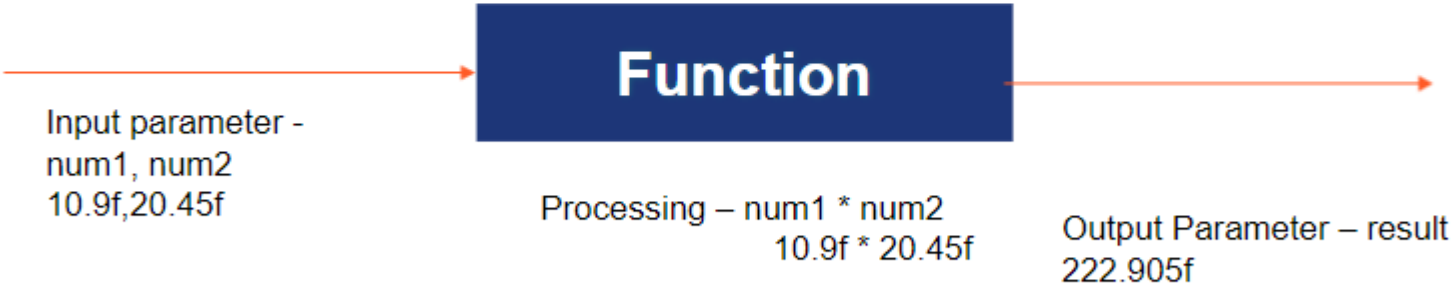

Input parameter - num1, num2
10,20

Processing – num1 + num2
10 + 20

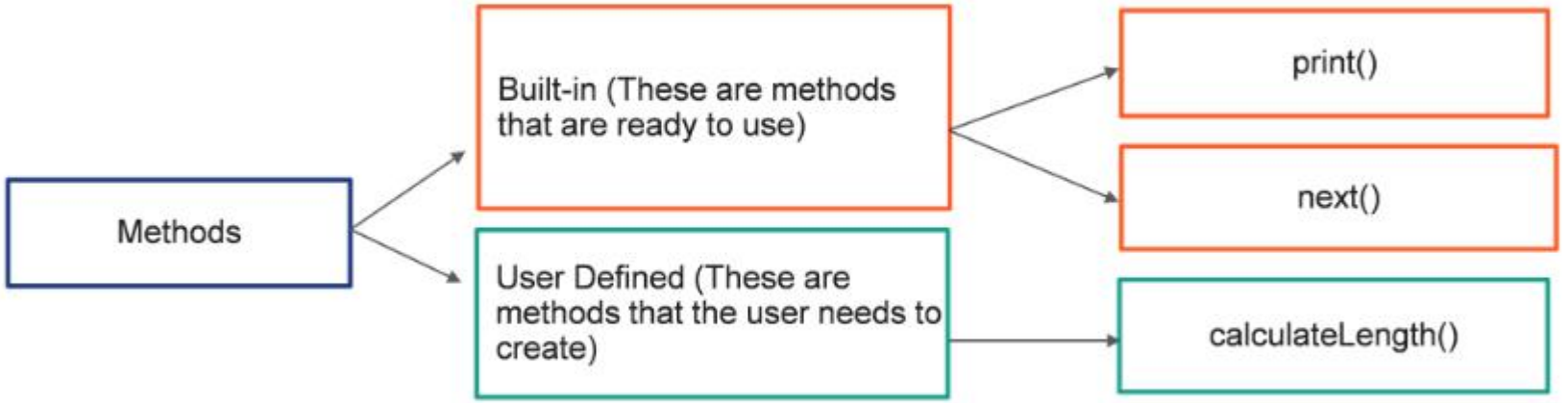
Output Parameter – result 30

# How Does a Function Process? (contd.)

- Multiply two numbers num1 and num2 and display the final output as result.
- Num1 and num2 are input parameters for the function and are of type float.
- Result is the output parameter of type float.



# Types of Methods in Java



# Method Structure

# Structure of a Method

```
<modifier> <return-type> <method-name>(<parameter-list)
{
    <method body>
}
```

- **Modifier:** The `modifier` tells the compiler how to call the method.
- **Return type:** The return type is the data type of the value the method returns i.e., the output. A method may or may not return a value. If the method does not return a value, the keyword used is **`void`** instead of the data type.
- **Method name:** The actual name of the method.
- **Parameter list:** A parameter is like a placeholder for input values to the method and is optional.
- **Method body:** The method body contains a collection of statements that define what the method does or the logic for the code goes here.

***Note*** *- Modifier will be discussed in detail later in the course*

# Method With Return Type

```
public int calculateArea(int length, int width){
    int area = length * width;
    return area;
}
```

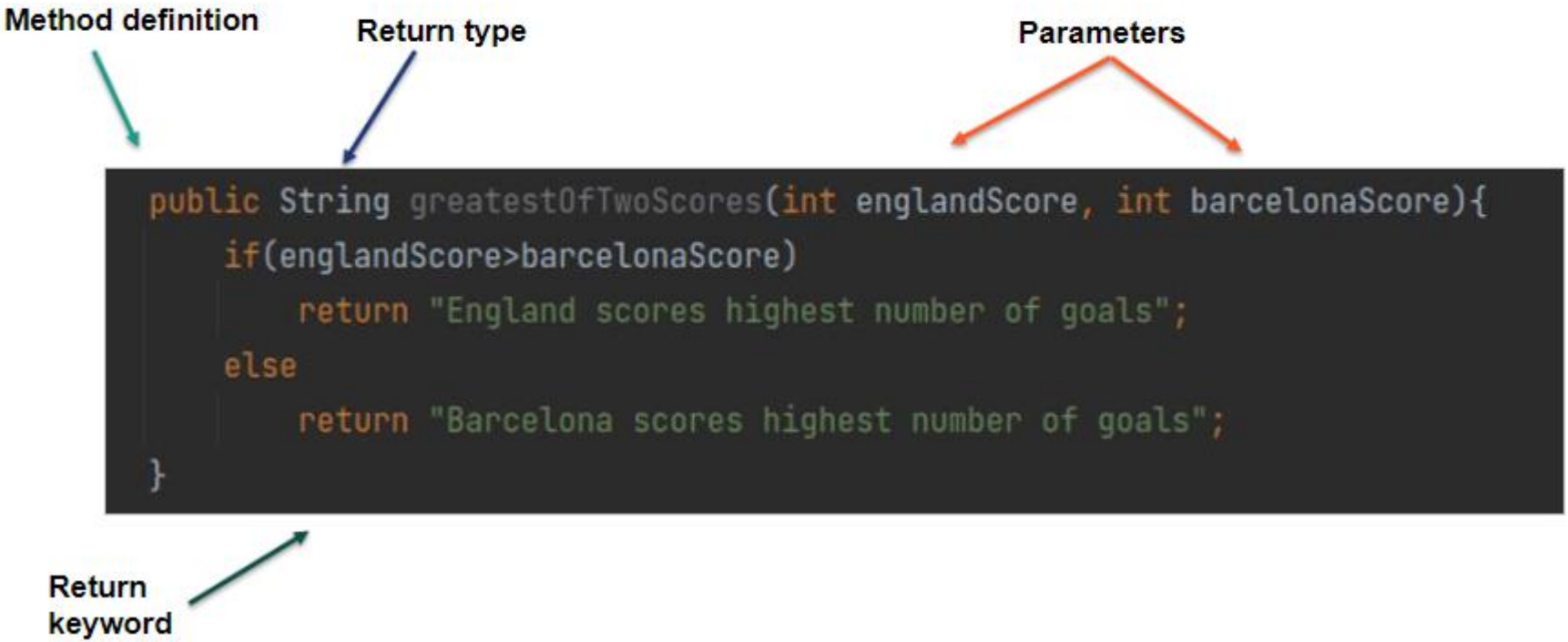
- The modifier is `public`.
- The return type can be any datatype but, in this method, `calculateArea` it is `int`.
- The method name is `calculateArea`.
- There are two parameters `length` and `width` which are of type `int`.
- The method body consists of the logic to calculate area.
- If a method returns a value, the return key word is used.
- Since the `calculateArea` method returns an `int` type, only `int` type variables or values can be returned from the method.

# Method Without Return Type

```
public void display(){
    System.out.println("Good Morning");
}
```

- The modifier is `public`.
- The return type is `void` which means the method will not return any value.
- The method name is `display`.
- The method parameters are optional therefore the display method does not take any parameters.
- The method body just prints a message on to the console.
- Since the method is not returning any value, the return keyword is not required.

# Method Structure



## Display Hello

Write a program having two methods.

First display the method which will return void and display "Happy Learning".

Secondly, create the main method and call the display method from it.

DEMO

# Method Call

```java
public class MethodDemo {
    1 usage
    public int calculateArea(int length, int breadth) {
        int area = length * breadth;
        return area;
    }

    1 usage
    public void display() {
        System.out.println("Good Morning");
    }

    public static void main(String[] args) {
        MethodDemo methodDemo = new MethodDemo();
        methodDemo.calculateArea( length: 10,  breadth: 20);
        methodDemo.display();

    }
```

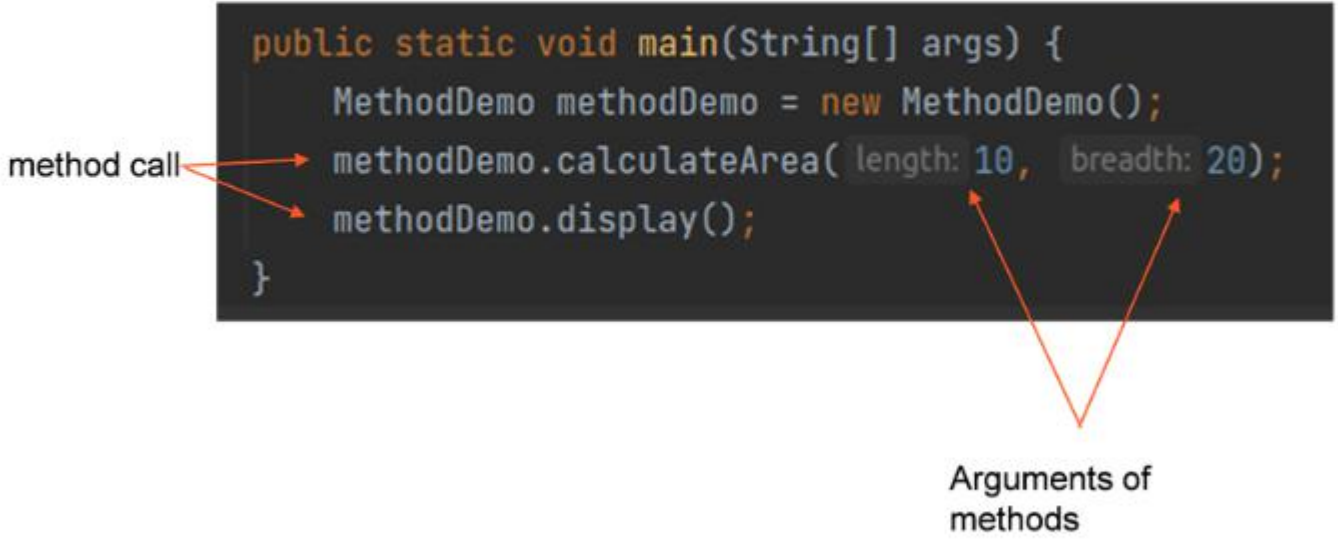
# Methods Inside the Class

- MethodDemo is a class consisting of three methods.
- The **userDefined** methods that **calculateArea**() and **display**() are called from the main method so they are executed.
- The dot operator(.) is used to call the method from the main method
- Inside the main method, an object of MethodDemo class must be created
- The methods are then called using the dot operator(.)

***Note*** *- More about objects and classes will be discuss in upcoming courses*

# Arguments to a Method

- Values are passed to a method during the method call. These values are called 'arguments to the method'.

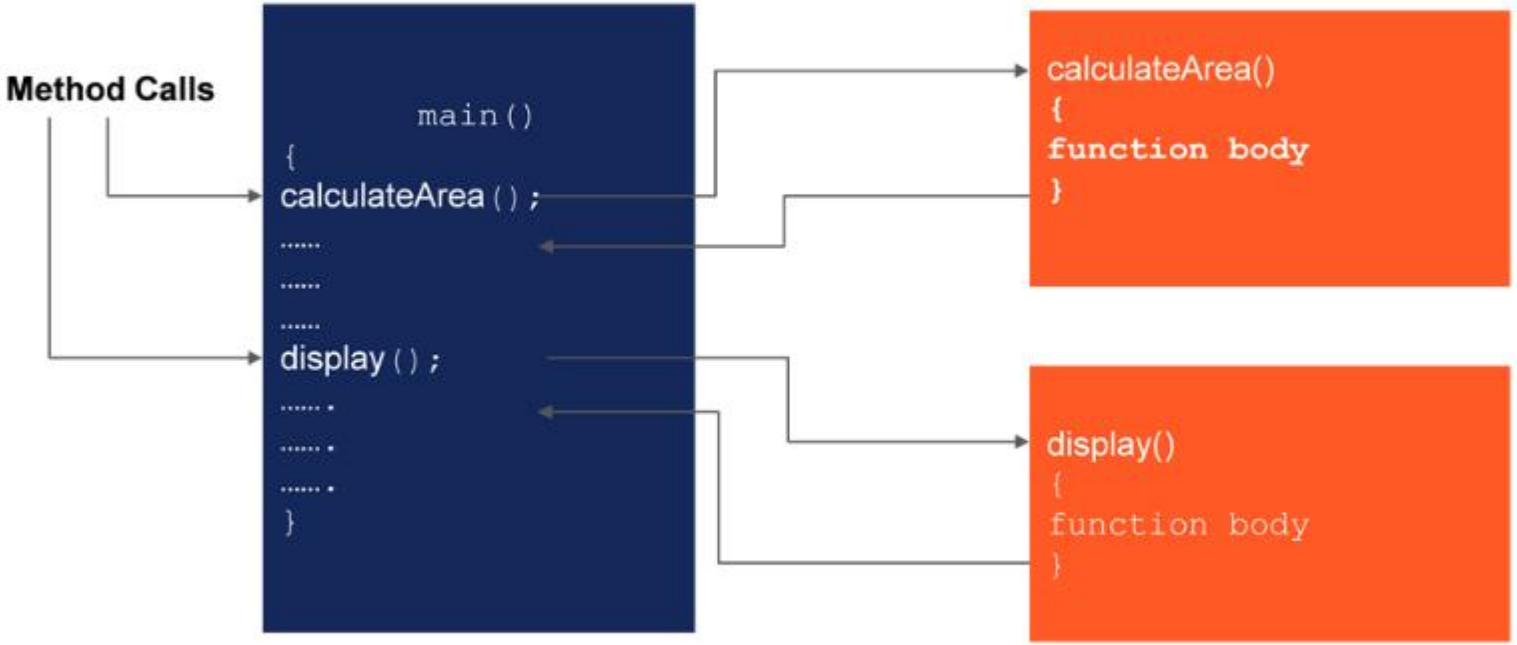


# Return Type of Method

- The `calculateArea` method returns an int type value.
- When calling `calculateArea` method from the main method, create a variable of type int in this case `int area`.
- The `area` will hold the value that is returned from `calculateArea`.

```
public static void main(String[] args) {
    MethodDemo methodDemo = new MethodDemo();
    int area =  methodDemo.calculateArea( length: 10,  breadth: 20);
    System.out.println(area);
}
```

# Method Call


Method Calls
main()
{
calculateArea();
.....
.....
.....
display();
.....
.....
.....
}
calculateArea()
{
function body
}
display()
{
function body
}

# Quick Check

Methods in Java are a combination of:

1. Return type
2. Method name
3. Parameters list
4. All the above

# Quick Check: Solution

Methods in Java are a combination of -

1. Return type
2. Method name
3. Parameters list
4. **All the above**

# Quick Check

What will be the return type of a method that does not return any value?

1. int
2. float
3. void
4. String

# Quick Check: Solution

What will be the return type of a method that does not return any value?

1. int
2. float
3. **void**
4. String

# Advantages of Modular Programming Using Methods

- Improves readability as it reduces the length of the programs.
- Simplifies understanding and debugging of the code.
- Allows reusability of the code.
- Makes management of large applications easier.

# Grades - Annual Performance Report

In a school, the teachers of grades 1- 9 have to prepare the annual performance report of their students.

Write a program that helps teachers to give an average of marks from the total marks.

1. Create two functions. One function will calculate the total marks and the other function will calculate the average marks.
2. Call both methods from the main method.

**DEMO**

# Local and Instance Variable

# Local and Instance Variable


- There are two types of Variables: Local and Instance
  - Local variables are declared inside the method, inside the block, or inside the loop.
  - Local variables need to be initialized at the time they are declared.
  - Instance variables are declared inside the class and not inside any of the methods.

***Note*** – *Instance variables will be discussed in detail in later course*