



Objects as a programming concept

IB Computer Science



*Content developed by
Dartford Grammar School
Computer Science Department*



HL Topics 1-7, D1-4



1: System design



2: Computer Organisation



3: Networks



4: Computational thinking



5: Abstract data structures



6: Resource management



7: Control



D: OOP

HL & SL D.3 Overview

D.3 Program development

D.3.1 Define the terms: class, identifier, primitive, instance variable, parameter variable, local variable

D.3.2 Define the terms: method, accessor, mutator, constructor, signature, return value

D.3.3 Define the terms: private, protected, public, extends, static

D.3.4 Describe the uses of the primitive data types and the reference class string

D.3.5 Construct code to implement assessment statements

D.3.6 Construct code examples related to selection statements

D.3.7 Construct code examples related to repetition statements

D.3.8 Construct code examples related to static arrays

D.3.9 Discuss the features of modern programming languages that enable internationalization

D.3.10 Discuss the ethical and moral obligations of programmers



1: System design

2: Computer Organisation



3: Networks

4: Computational thinking



5: Abstract data structures

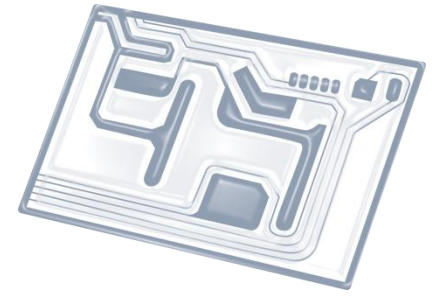
6: Resource management



7: Control

D: OOP





Topic D.3.2

Define the terms: **method**, **accessor**, **mutator**, **constructor**, **signature**, **return value**



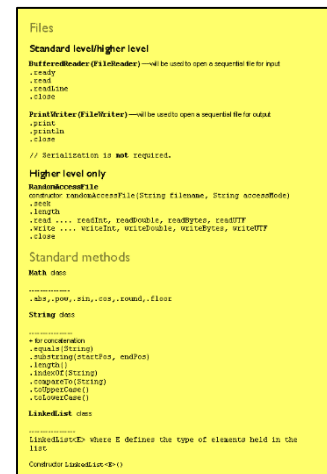
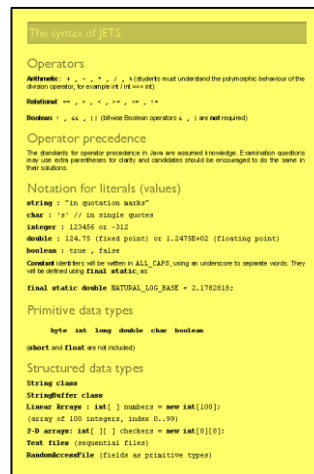
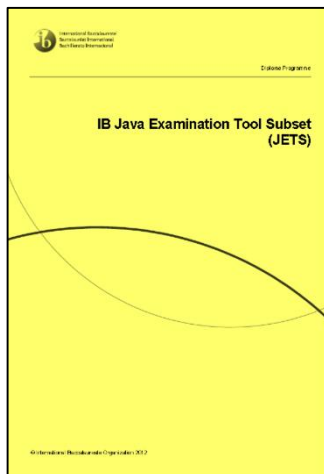
"MY DAD DOESN'T KNOW A LOT ABOUT COMPUTERS.
HE THINKS ISDN AND MP3 WERE THE ROBOTS ON 'STAR WARS'."



Exam note!

This curriculum point relates closely to the details published in the JETS booklet.

You will **NOT** get a copy of this booklet in the Paper 2 exam.

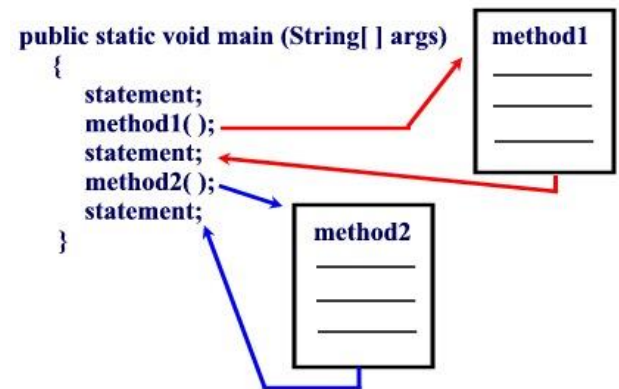


Definition: **method**

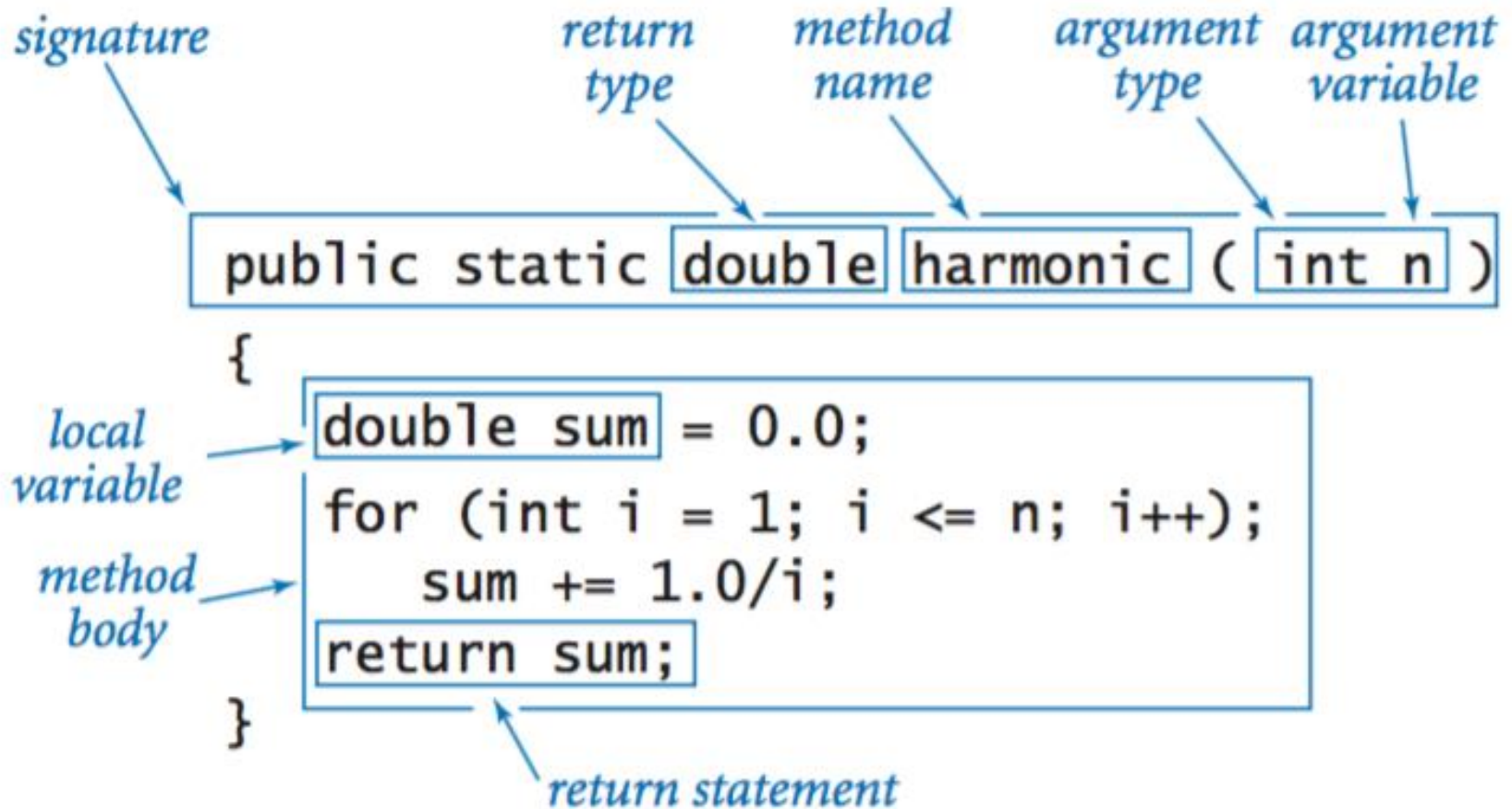
A **method** is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the **method's** name.

A **method** can be described as a **subprogram** that acts on data and often returns a value.

Each **method** has its own name (identifier).



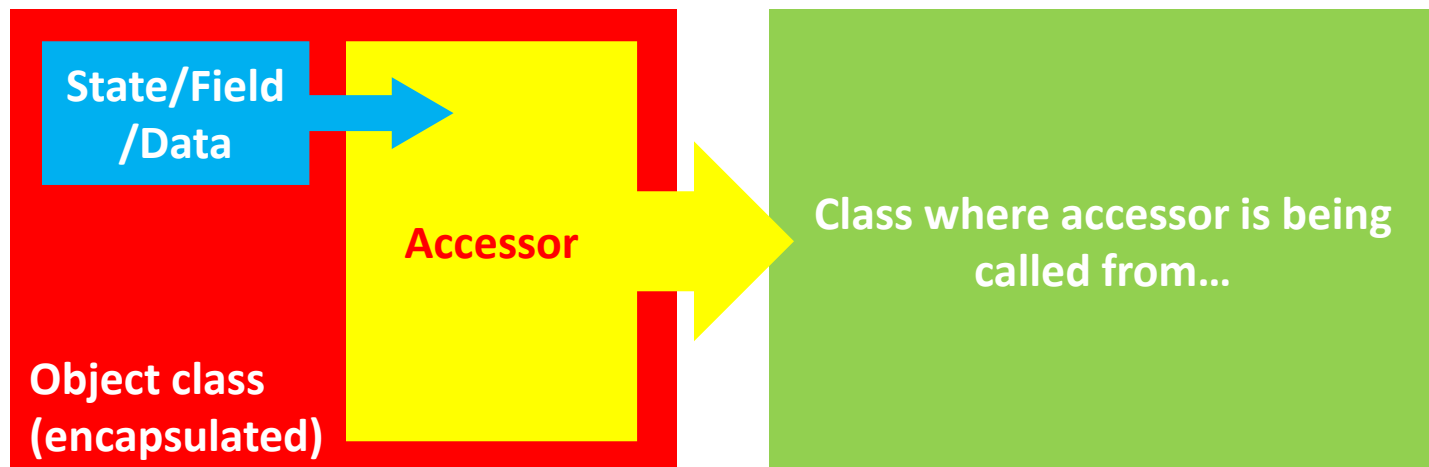
Example: method



Definition: **accessor**

An **accessor** is a type of method used in Java OO programming that which returns the value of a private instance (class) variable.

It is also known as a **getter** method.



Example: accessor

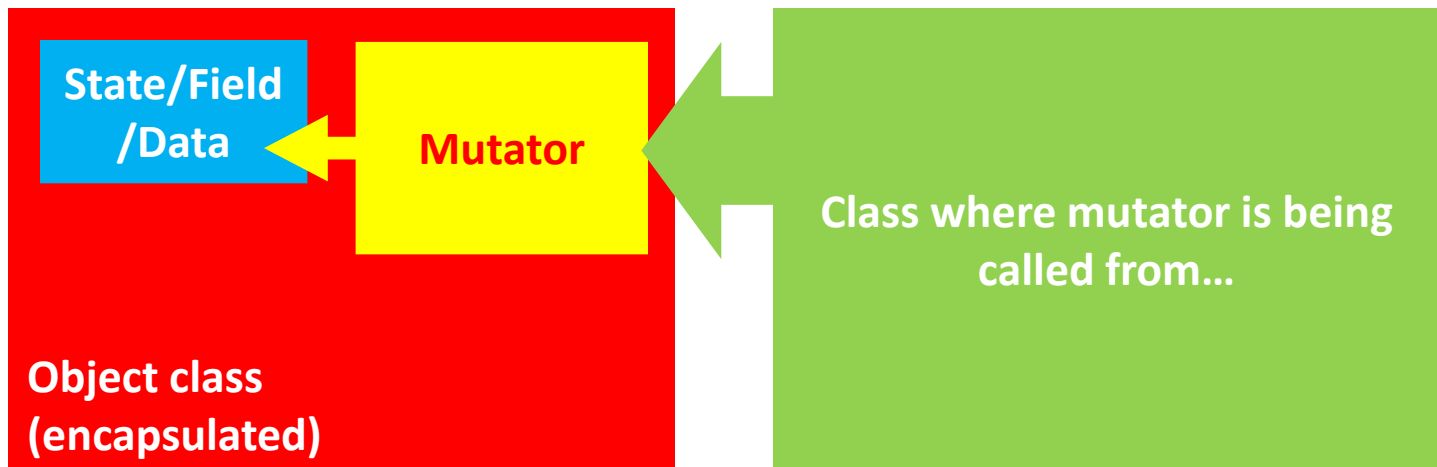
```
public class Student {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String newName) {  
        name = newName;  
    }  
}
```

} accessor

Definition: **mutator**

A **mutator method** is a method used to control changes to a encapsulated instance (class) variable/state.

They are also widely known as **setter** methods.

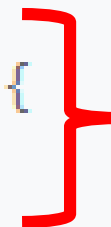


Example: mutator

```
public class Student {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  

```

```
    public void setName(String newName) {  
        name = newName;  
    }  
}
```

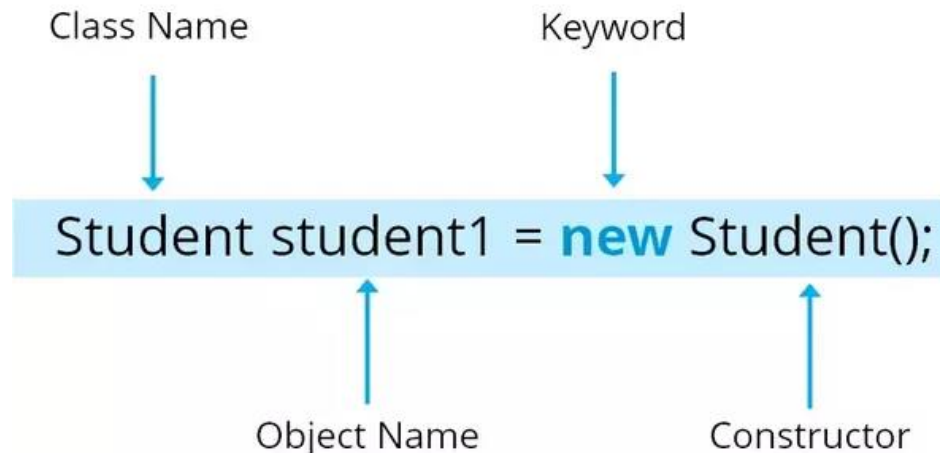


mutator

Definition: **constructor**

A **constructor method** is an instance **method** (defined inside a class) that is invoked when an object of that class is created (by using the **new** keyword)

Object creation rule in Java: When an object is created, one of the **constructor method** in the class must be invoked (to initialize the instance variables in the object)



Example: constructor

```
public class StudentResults {
```

```
    private String Full_Name;
```

```
    private String Exam_Name;
```

```
    private String Exam_Score;
```

```
    private String Exam_Grade;
```

```
    StudentResults () {
```

```
}
```

```
}
```



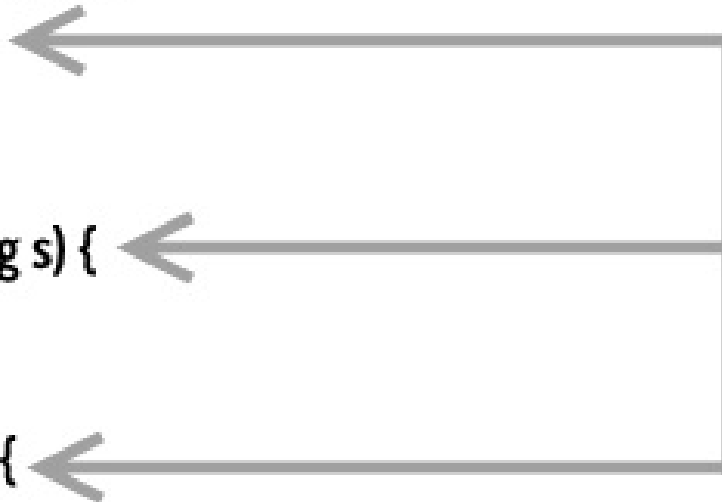
Constructor

Same name as class

No return type

Example 1: Overloaded constructors

```
public class Demo {  
    Demo() {  
        ..  
    }  
    Demo(String s) {  
        ...  
    }  
    Demo(int i) {  
        ...  
    }  
    .....  
}
```



Three overloaded
constructors -
They must have
different
Parameters list

Example 2: Overloaded constructors

```
public class Student{
```

```
String name;  
String surName;  
int age;
```

```
public Student()  
{  
    name ="jon";  
    surName = "doe";  
    age =0;  
}
```

The constructor

Its job is to ensure all instance variables are initialized.

when it executes
Student jen ;
jen = new Student() ;

```
public Student(String n,String sN)  
{  
    name =n ;  
    surName =sName;  
    age =0;  
}
```

Overloaded constructor

when it executes
Student jen ;
jen = new Student("Jen","Smith") ;

```
public Student(String n,String sN int a)  
{  
    name = n ;  
    surName = sName;  
    age = a ;  
}
```

Overloaded constructor

when it executes
Student jen ;
jen = new Student("Jen","Smith", 18) ;

```
public void setAge(to)  
{  
    age =to;  
}
```

```
}
```

Difference: Constructor vs Method

Difference between Method and Constructor

	Method	Constructor
1	Method can be any user defined name	Constructor must be class name
2	Method should have return type	It should not have any return type (even void)
3	Method should be called explicitly either with object reference or class reference	It will be called automatically whenever object is created
4	Method is not provided by compiler in any case.	The java compiler provides a default constructor if we do not have any constructor.

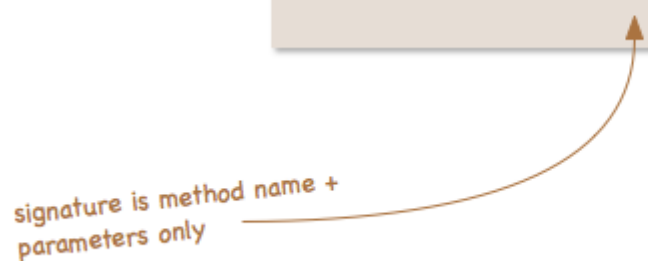


Definition: **signature**

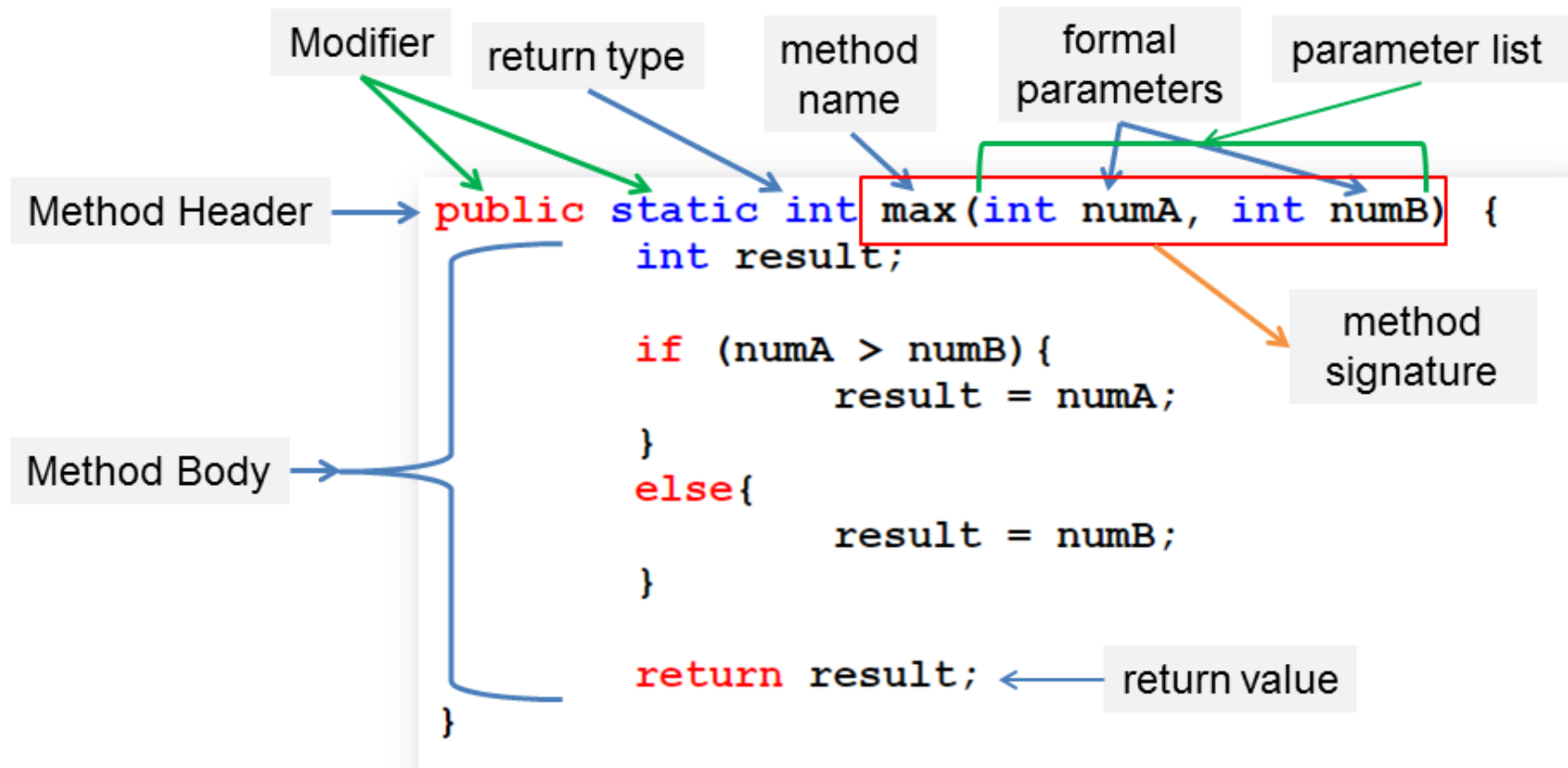
- A **method signature** is part of the method declaration. It is the **combination** of the method **name** and the **parameter list**.
- The reason for the emphasis on just the method name and parameter list is because of **overloading** methods that have the same name but accept different parameters.

```
public double getMyFundsFromBank(String bankName)
```

signature is method name +
parameters only

A curved arrow points from the text 'signature is method name + parameters only' to the 'getMyFundsFromBank(String bankName)' part of the code snippet above.

Example: signature



Definition: **return value**

- **return** is a reserved keyword in Java; it cannot be used as an identifier.
- It is used to exit from a method, with or without a value.
- **return** can be used with methods in two ways:
 - A. **Methods returning a value** : For methods that define a return type, return statement must be immediately followed by return value.
 - B. **Methods not returning a value** : For methods that don't return a value, return statement can be skipped.

Example: return-type method

```
// Java program to illustrate usage
// of return keyword

class A {

    // Since return type of RR method is double
    // so this method should return double value
    double RR(double a, double b)
    {
        double sum = 0;
        sum = (a + b) / 2.0;
        // return statement below:
        return sum;
    }
    public static void main(String[] args)
    {
        System.out.println(new A().RR(5.5, 6.5));
    }
}
```

Output:

6.0

Example: non return-type method

```
// Java program to illustrate no return
// keyword needed inside void method

class GFG {

    // Since return type of RR method is
    // void so this method shouldn't return any value
    void RR(int a, int b)
    {
        int sum = 0;
        sum = (a + b) / 10;
        System.out.println(sum);
        // no return statement in this method
    }

    public static void main(String[] args)
    {
        new GFG().RR(5, 5);
    }
}
```

Output:

1

Procedures vs Functions

- Methods are also known as Procedures or Functions:
 - **Procedures**: don't return any value (void).
 - **Functions**: return a value
- No method can return more than one value at a time in Java.

```
modifier returnType nameOfMethod (parameter List)
{
    // method body
    return variable/value that matches return type
}
```