Object Oriented Programming Java: Methods

Bikash Karmokar
Lecturer,
Department of CSE,
Bangladesh University

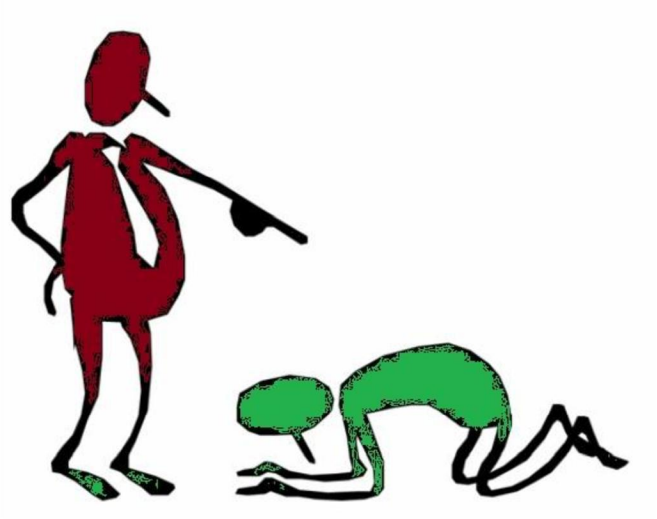
What do you need to know?

1. Data types
2. Operators
3. Conditional statements
4. Loops
5. At least 1D array

Boss

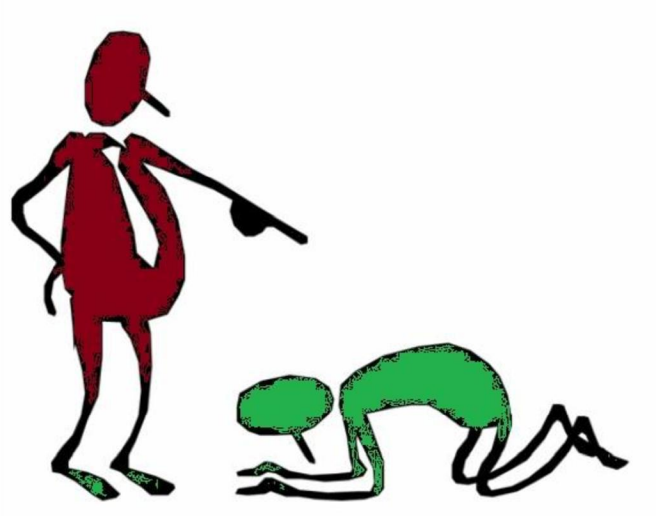
```
public static void main(String[ ] args)
{
    int a=10,b=5;

    int sum =  a+b;
    System.out.println(a+b);

    int difference =  a-b;
    System.out.println(a-b);

    int multiply =  a*b;
    System.out.println(a*b);

    int division =  a/b;
    System.out.println(a/b);

}
```

Boss



Assistant

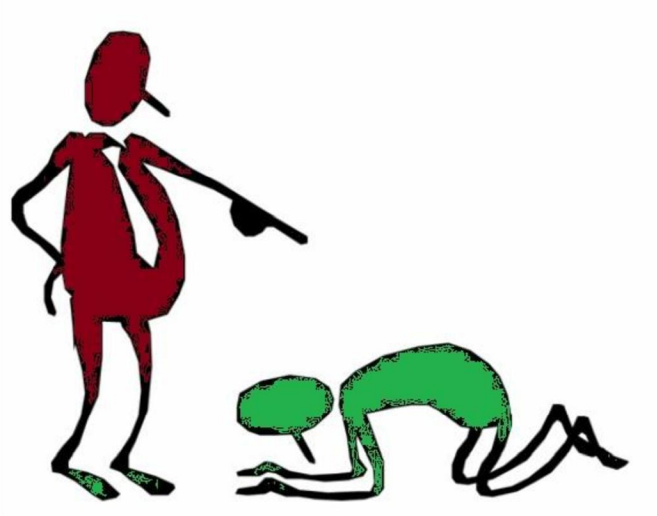
Boss



Assistant

```
int a=10,b=5;
int sum =  a+b;
System.out.println(sum);
```

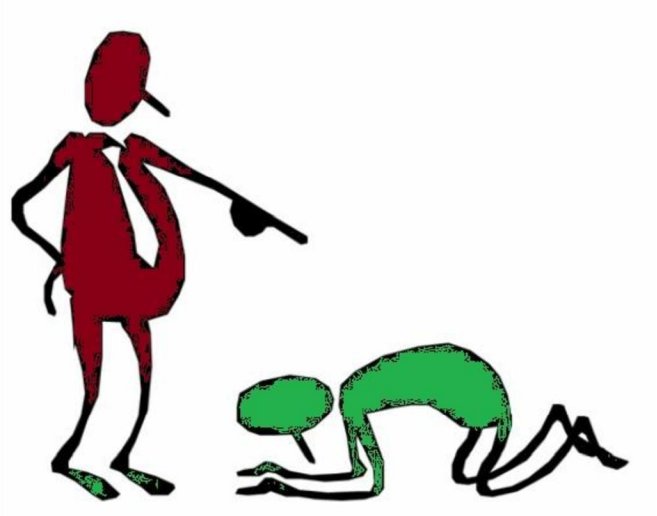
Boss



Assistant

```
add( )
{
    int a=10,b=5;
    int sum =  a+b;
    System.out.println(sum);
}
```

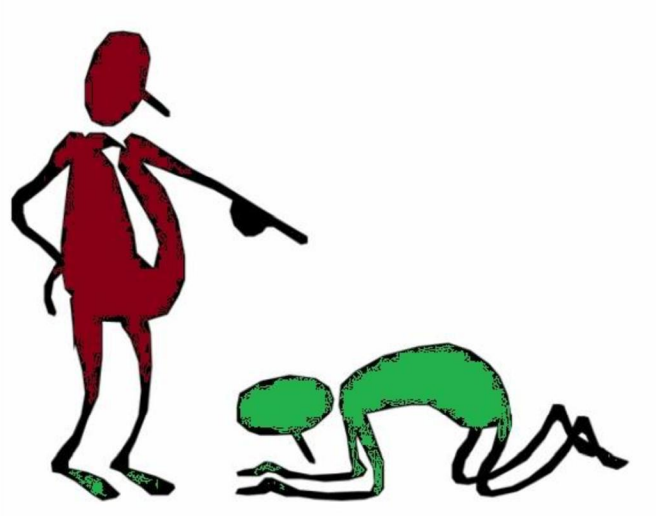
Boss



Assistant

```
void add( )
{
    int a=10,b=5;
    int sum =  a+b;
    System.out.println(sum);
}
```

Boss

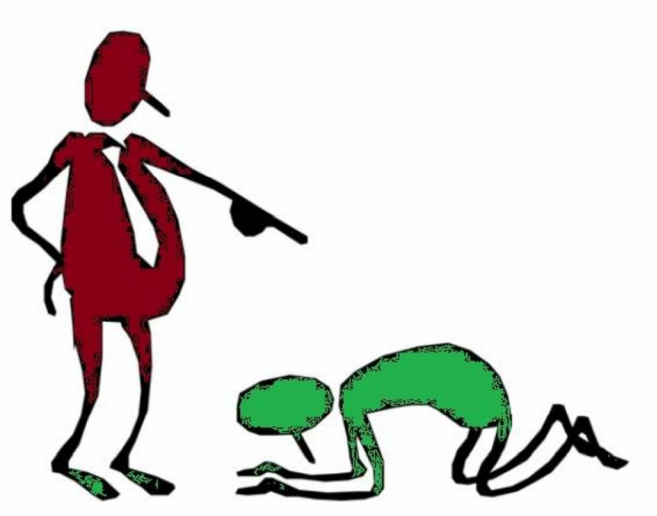


```
public static void main(String[] args)
{

}
```

Assistant

```
void add( )
{
    int a=10,b=5;
    int sum =  a+b;
    System.out.println(sum);
}
```

Boss

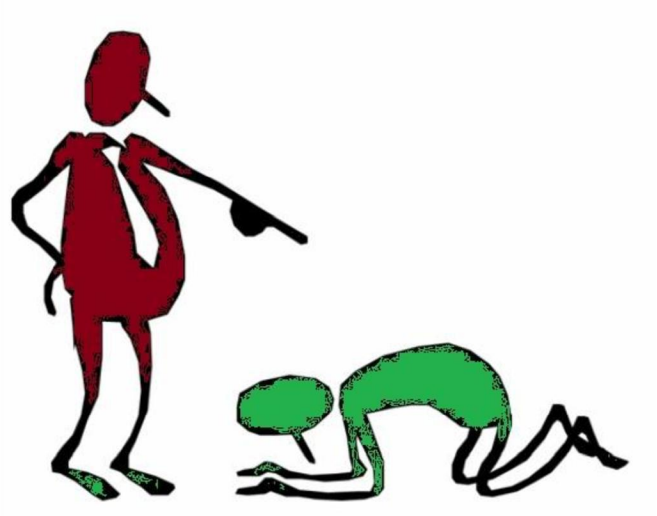


Assistant

```
public static void main(String[] args)
{
    add( );
}
```

```
void add( )
{
    int a=10,b=5;
    int sum =  a+b;
    System.out.println(sum);
}
```

Boss

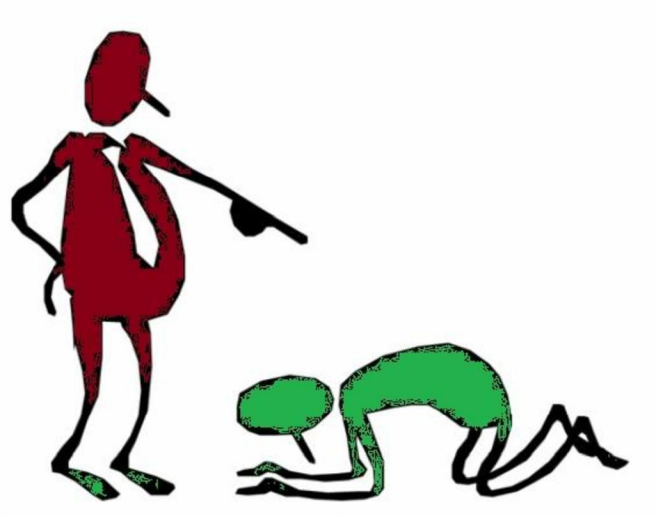


Assistant

```
public static void main(String[] args)
{

}
```

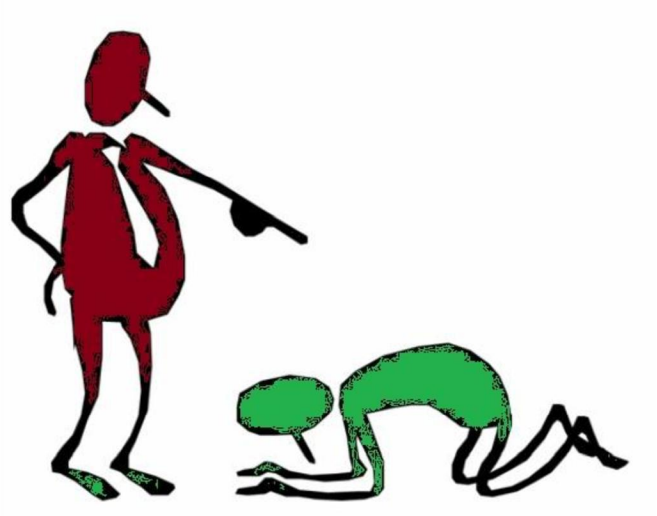
Boss

Assistant



```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss

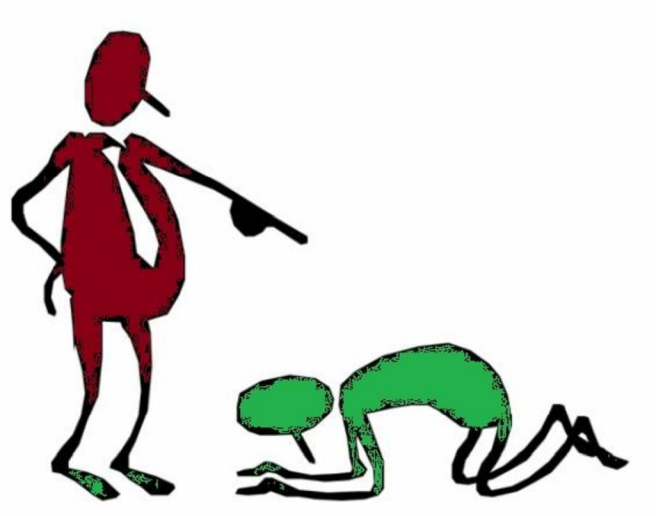


Assistant

add(**int a, int b**)

```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss



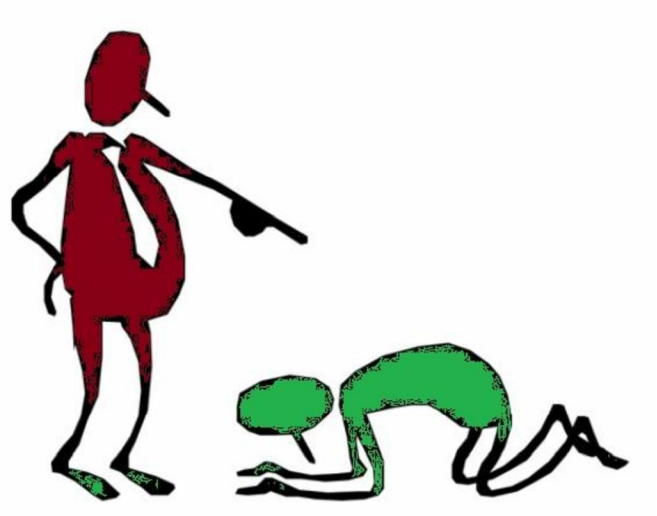
Assistant

add(**int a, int b**)

5 , 7

```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss

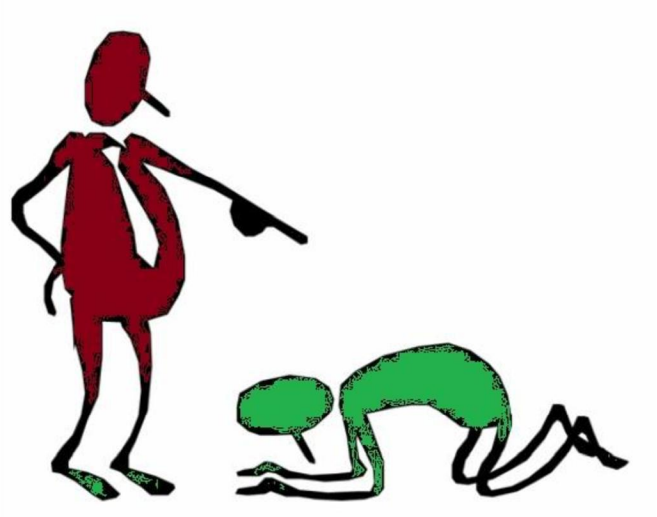


Assistant

add(**int a, int b**)

```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss



```
public static void main(String[] args)
{
    add(5,7);
}
```

Assistant

```
void add(int a, int b )
{

}
```

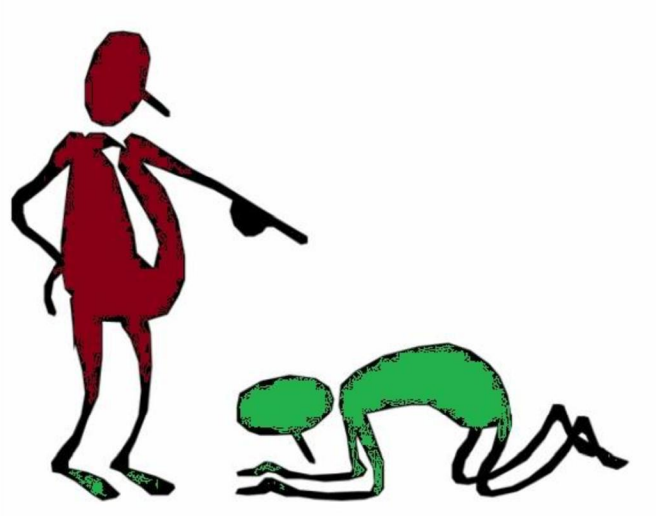
Boss

Assistant

```
public static void main(String[] args)
{
    add(5,7);
}
```

```
void add(int a, int b )
{
    int sum =  a+b;
    System.out.println(sum);
}
```

Boss

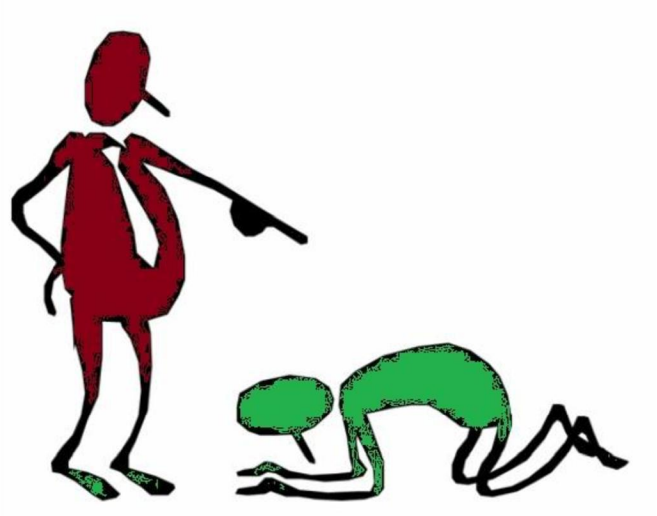


Assistant

```
public static void main(String[] args)
{

}
```

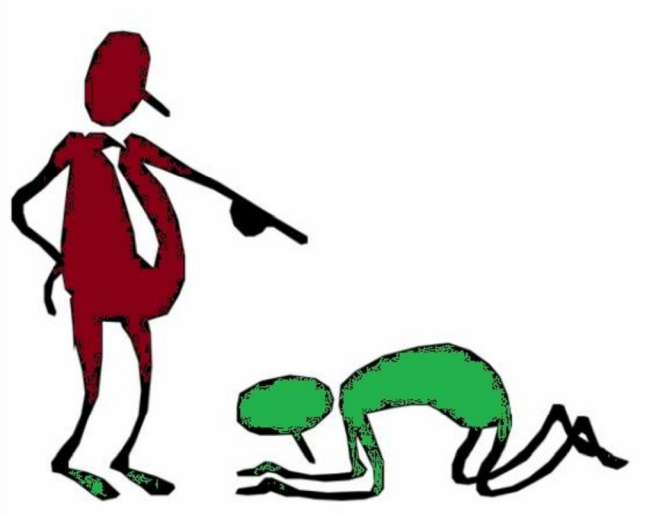
Boss

Assistant



```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss

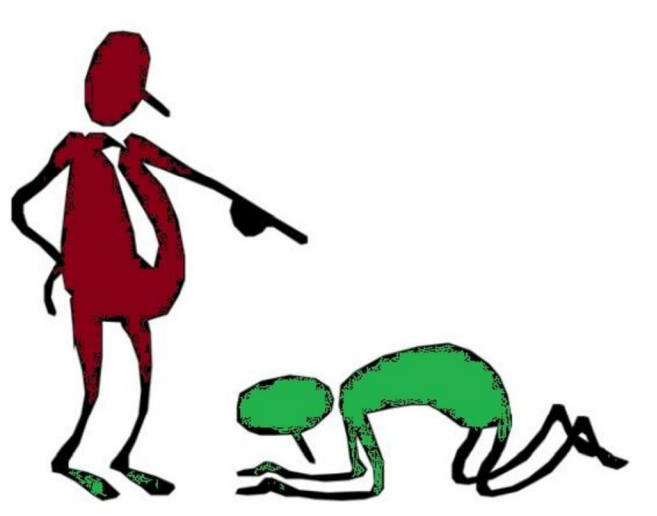


Assistant

add(**int a, int b**)

```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss



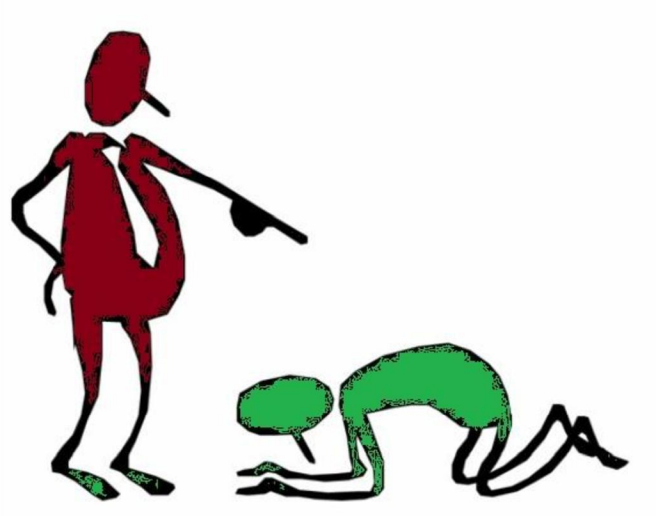
Assistant

add(**int a, int b**)

5 , 7

```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss

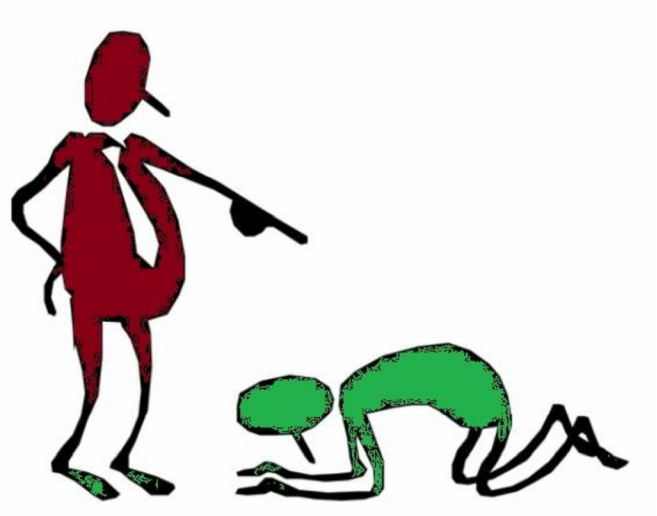


Assistant

add(**int a, int b**)

```
public static void main(String[] args)
{
    add(5,7);
}
```

Boss

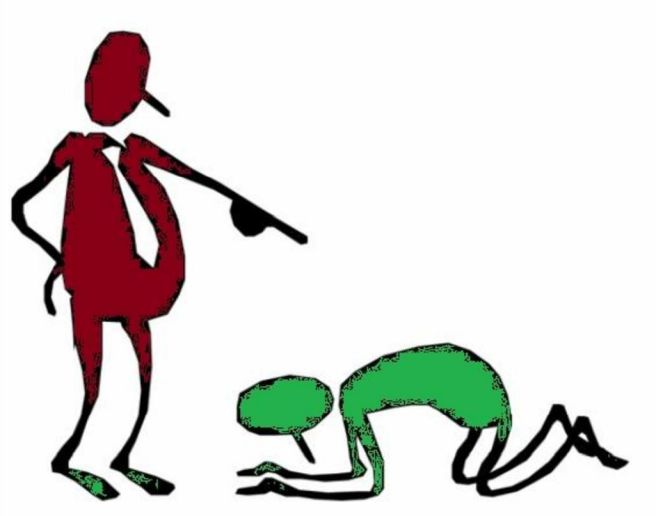


```
public static void main(String[] args)
{
    add(5,7);
}
```

Assistant

```
    add(int a, int b )
{

}
```

Boss

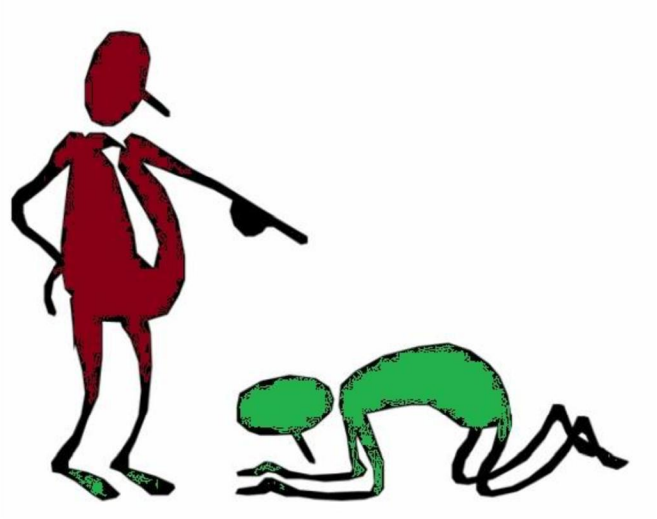


```
public static void main(String[] args)
{
    add(5,7);
}
```

Assistant

```
    add(int a, int b )
{
    int sum =  a+b;

}
```

Boss



Assistant

```
public static void main(String[] args)
{
    add(5,7);
}
```

```
int  add(int a, int b )
{
    int sum =  a+b;

    return sum;

}
```

Boss

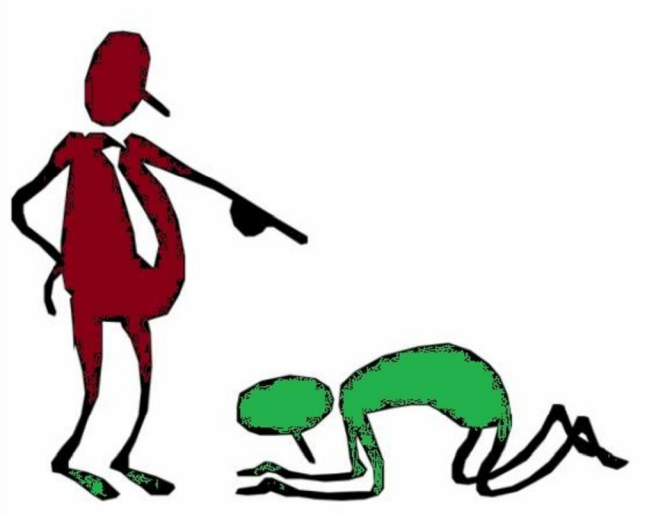
Assistant

```
public static void main(String[] args)
{
    add(5,7);
}
```

```
int  add(int a, int b )
{
    int sum =  a+b;

    return sum;

}
```

12

Boss



Assistant

```
public static void main(String[] args)
{
    add(5,7);
}
```

12

```
int  add(int a, int b )
{
    int sum =  a+b;

    return sum;

}
```

Boss

Assistant

```
public static void main(String[] args)
{
 int total=add(5,7);
}
```

12

```
int  add(int a, int b )
{
    int sum =  a+b;

    return sum;

}
```

Types of methods

Bikash Karmokar, Lecturer, CSE,
Bangladesh University

Types of methods

1. Dumb method.
--Does the same thing every time

2. Clever Method.
--Take input from Boss

Types of methods

3. Smart method.
--Take input from Boss and return result to the Boss

1.
```
void add( )
{
    int a=10,b=5;
    int  sum = a+b;
    System.out.println(sum);
}
```

2.
```
void add( int a, int b)
{
    int  sum = a+b;
    System.out.println(sum);
}
```

3.
```
int add( int a, int b)
{
    int  sum = a+b;
    return sum;
}
```

Steps in using a method

Bikash Karmokar, Lecturer, CSE,
Bangladesh University

Steps in using a method

Step 1: Define the method

Steps in using a method

Step 1: Define the method

Step 2: Use the method

Step 1: Define the method

Specify name

add ( )

Step 1: Define the method

Specify name

Specify what the method does : method body

```
add  (                    )
{

}
```

Step 1: Define the method

Specify name

Specify what the method does : method body

Specify what data the method needs : parameters

```
add ( int a, int b )
{

}
```

Step 1: Define the method

Specify name

Specify what the method does : method body

Specify what data the method needs : parameters

What type of data and what variable does the method return? return type and data

```
int add ( int a, int b )
{
    return a+b;
}
```

Step 1: Define the method

Specify name

Specify what the method does : method body

Specify what data the method needs : parameters

What type of data and what variable does the method return? return type and data

Can everyone see the method? Does it have restricted access? : modifier

```
public int add ( int a, int b )
{
    return a+b;
}
```

Step 2: Use the method

```
public int add ( int a, int b )
{
    int sum = a+b;
    return sum;
}
```

Step 2: Use the method

```
public int add ( int a, int b )
{
    int sum = a+b;
    return sum;
}
```

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = add(x,y);
}
```

When should I make a method?

Ask yourself these questions:

Is that a single operation you want to do? Ex: Finding Average

Does it need several steps for doing that operation? Ex: Take values then calculate average.

The structure of a method

Modifier Return value Method name

Method header → public static int max (int num1, int num2)

{

Method body → ...

...

Return value → return answer;

}

Parameter list

The method header specifies the modifier, return value type, method name and parameters of the method.

The optional modifier tells the compiler how to call the method

The variables defined in the method header are known as formal parameters or simply parameters.

The return value type is the data type of the value the method return.

The structure of a method

Modifier Return value Method name

Method header → public static int max (int num1, int num2)

{

Method body → ...

...

Return value → return answer;

}

Parameter list

When a method is invoked, you pass a value to the parameter.

The parameter list refers to the type, order and number of the parameters of a method.

The method name and the parameter list together constitute the method signature.

Parameter are optional.

The Method Callstack

Bikash Karmokar, Lecturer, CSE,
Bangladesh University

The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```

The Method Callstack

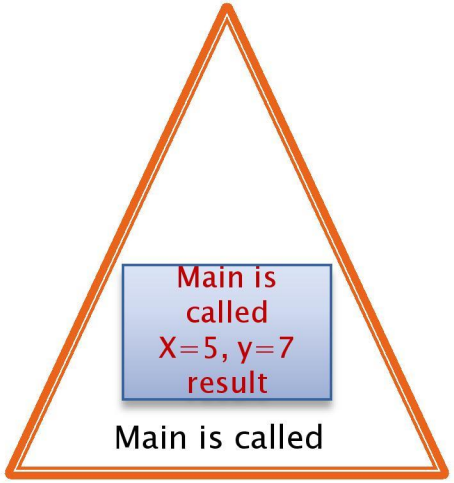
```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```

The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

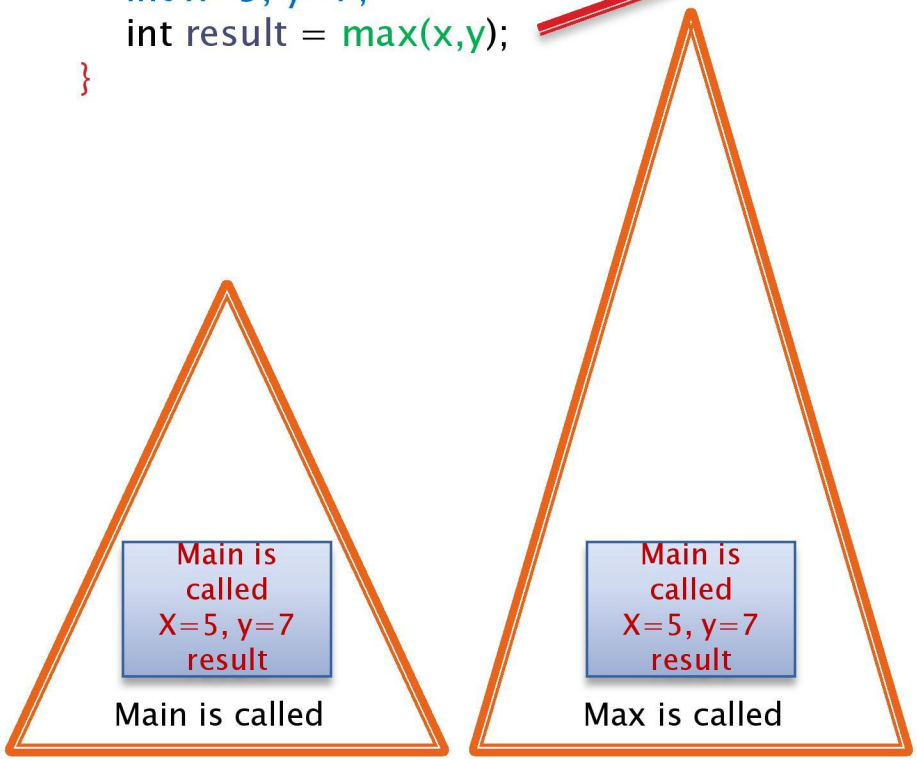
```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```



The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

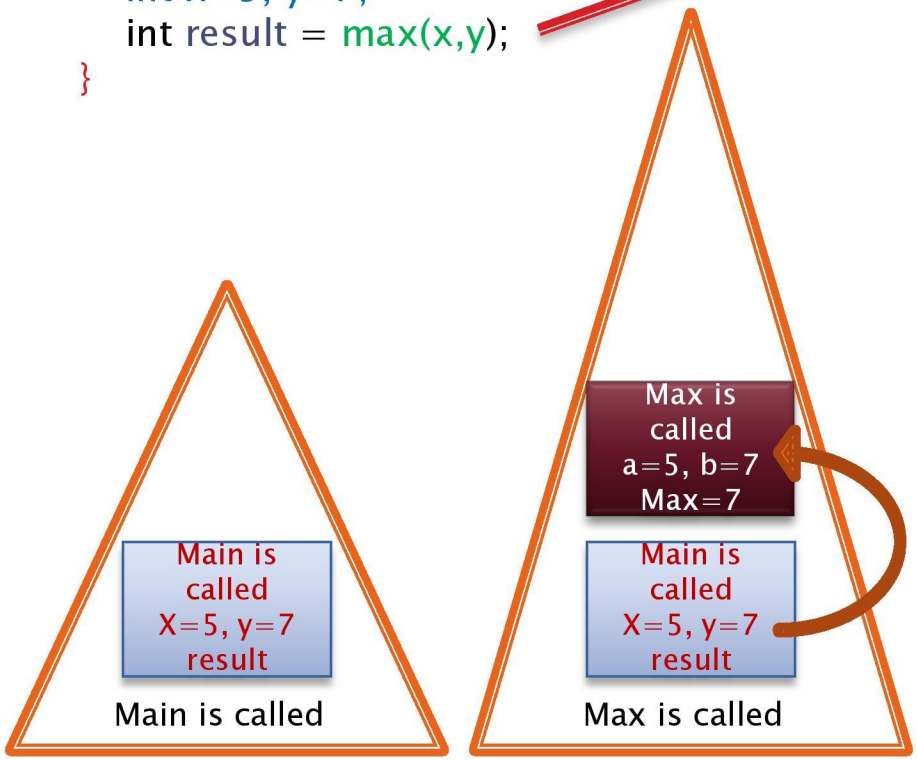
```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```



The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

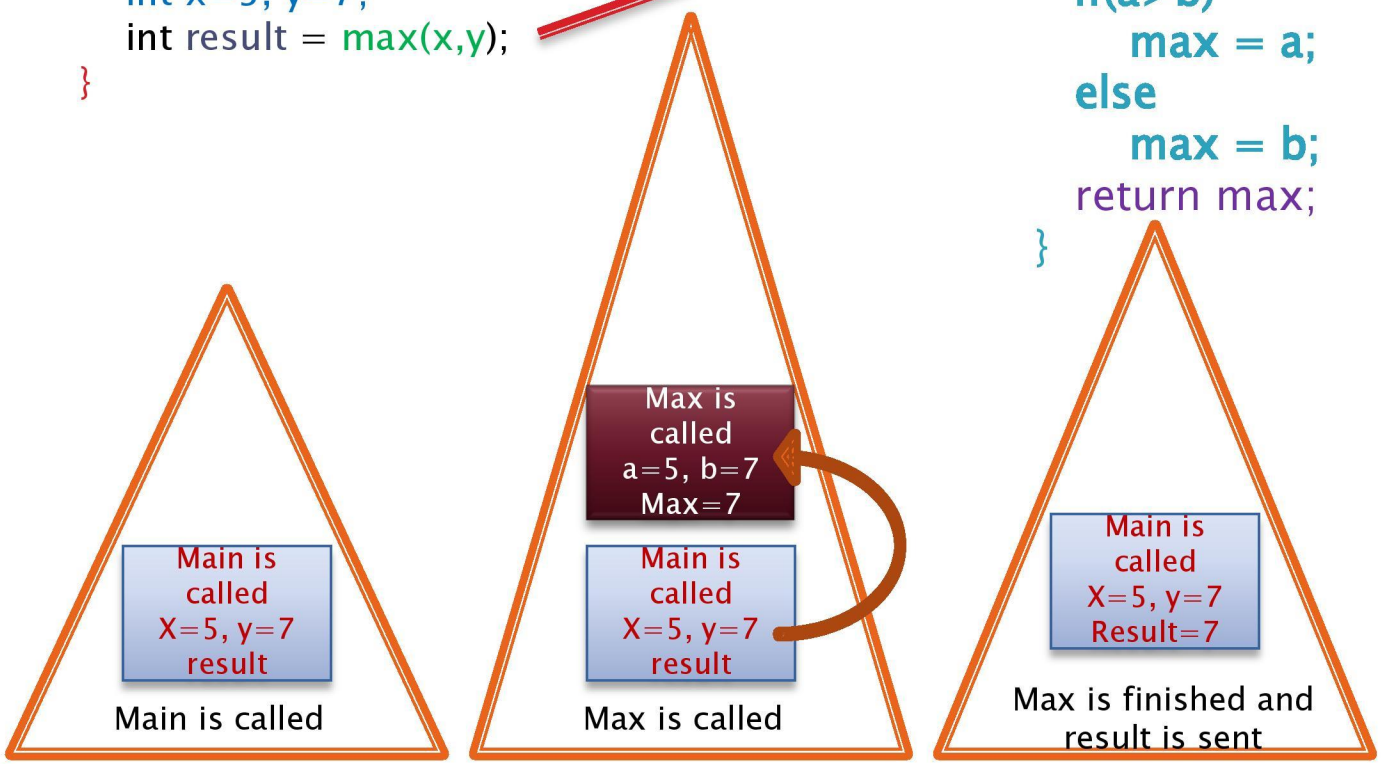
```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```



The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

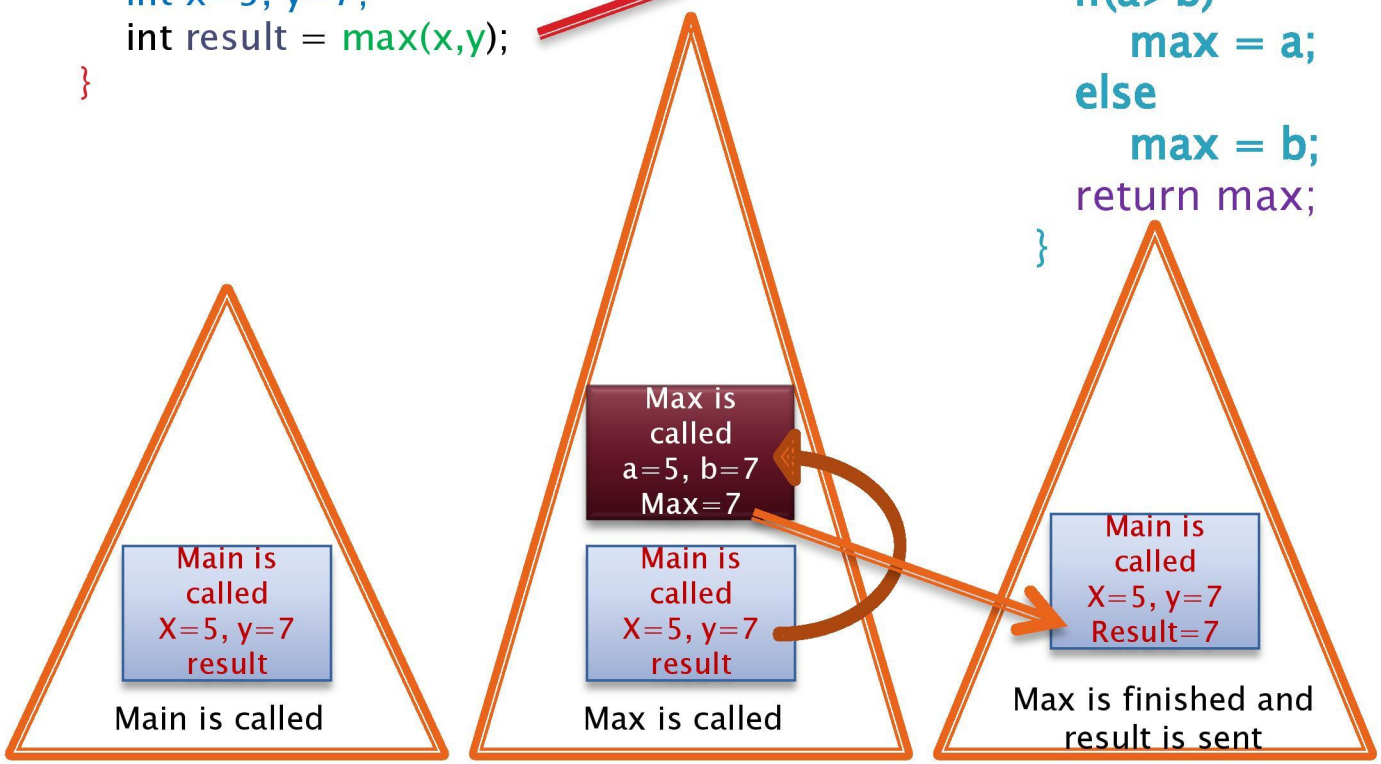
```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```



The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

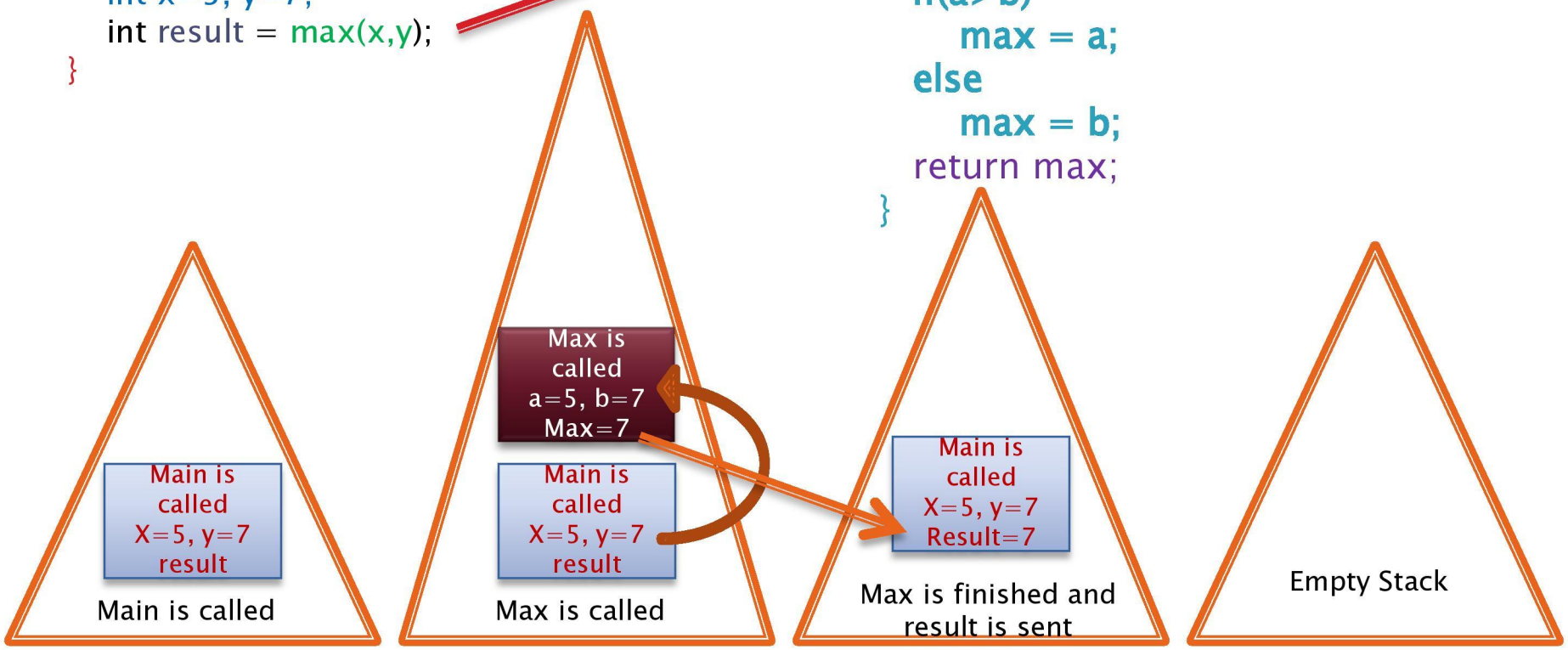
```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```



The Method Callstack

```
public static void main(String[ ] args)
{
    int x=5, y=7;
    int result = max(x,y);
}
```

```
public static int max ( int a, int b)
{
    int max;
    if(a>b)
        max = a;
    else
        max = b;
    return max;
}
```



Method Overloading

Bikash Karmokar, Lecturer, CSE,
Bangladesh University

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

```
public  class Maths
{
    public int max(int a, int b)
    {
        int max;
        if(a>b)
            max=a;
        else
            max=b;
         return max;
    }
}
```

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

```
public  class Maths
{
    public int max(int a, int b)
    {
        int max;
        if(a>b)
            max=a;
        else
            max=b;
         return max;
    }
}
```

```
public int max(double a, double b)
{
    int max;
    if(a>b)
        max=a;
    else
        max=b;
     return max;
}
```

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

max(3, 5)

```
public  class Maths
{
    public int max(int a, int b)
    {
        int max;
        if(a>b)
            max=a;
        else
            max=b;
         return max;
    }
}
```

```
public int max(double a, double b)
{
    int max;
    if(a>b)
        max=a;
    else
        max=b;
     return max;
}
```

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

max(3, 5)

```
public  class Maths
{
    public int max(int a, int b)
    {
        int max;
        if(a>b)
            max=a;
        else
            max=b;
         return max;
    }
}
```

```
public int max(double a, double b)
{
    int max;
    if(a>b)
        max=a;
    else
        max=b;
     return max;
}
```

max(3.0, 5.0)

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

max(3, 5)

```
public  class Maths
{
    public int max(int a, int b)
    {
        int max;
        if(a>b)
            max=a;
        else
            max=b;
         return max;
    }
}
```

```
public int max(double a, double b)
{
    int max;
    if(a>b)
        max=a;
    else
        max=b;
     return max;
}
```

max(3.0, 5.0)

max(3, 5.0)

What is method overloading?

Method within the same class that have the same name but slightly different characteristics (signature) like different number of parameters or different type of parameters that perform related operations are called **overloaded method**.

max(3, 5)

```
public  class Maths
{
    public int max(int a, int b)
    {
        int max;
        if(a>b)
            max=a;
        else
            max=b;
         return max;
    }
}
```

```
public int max(double a, double b)
{
    int max;
    if(a>b)
        max=a;
    else
        max=b;
     return max;
}
```

max(3.0, 5.0)

max(3, 5.0)

What is method overloading?

- When you create two or more methods with the same name within the same class, you are overloading the method name.
- Different method definitions have something different about their parameter lists.
- Java distinguishes overloaded methods according to the number of parameters and the types of the parameters.
- When java code is written to call a method, the compiler checks it the first argument has the same type as the first parameter, the second argument has the same type as the second parameter.
- If there is no match, java will try some simple type conversion [only smaller to bigger data type] such as casting int to double, to see whether that produces a match.
- It can be applied to void methods, to methods that return a value, to static methods, to non-static methods and also to constructors.

Method overloading: remembrance

- The parameter list must be different for overloaded method.
- You can't use return type or access modifier alone to create overloaded methods.
- Overloaded methods causing ambiguous invocation lead to compiler errors.

Ambiguous overloading

Ambiguous overloading

```
public int max(int a, double b)
{
        ..........
        ..........
}
```

Ambiguous overloading

```
public int max(int a, double b)
{
        ..........
        ..........
}


public int max(double a, int b)
{
        ........
        ........
}
```

Ambiguous overloading

```
public int max(int a, double b)
{
        ..........
        ..........
}
```

max(3, 5.0)

```
public int max(double a, int b)
{
        ........
        ........
}
```

Ambiguous overloading

```
public int max(int a, double b)
{
        .........
        .........
}
```

max(3, 5.0)

max(3.0, 5)

```
public int max(double a, int b)
{
        ........
        ........
}
```

Ambiguous overloading

```
public int max(int a, double b)
{
        .........
        .........
}
```

max(3, 5.0)

max(3.0, 5)

```
public int max(double a, int b)
{
        ........
        ........
}
```

max(3, 5)

Ambiguous overloading

```
public int max(int a, double b)
{
        .........
        .........
}
```

max(3, 5.0)

max(3.0, 5)

```
public int max(double a, int b)
{
        ........
        ........
}
```

max(3, 5)

Thank You