Methods

Syntax:

modifiers return_type name (parameter list) exception list {
(1) return type, (2) name, (3) parameter list

body

} return type (4)

Ex:

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

return type ← int; name → add; parameter → (int a, int b)

100 lines { body

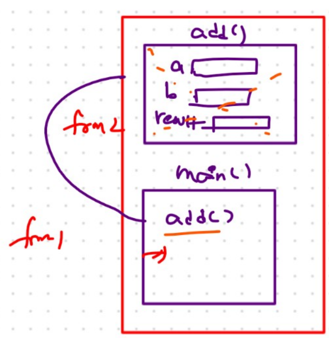
→ add(10,2)

→ add(20,10)

Notes

→ method is a set of instructions, helps us to do repeated tasks just defining once, & calling it many times as when we require

→ when ever a method starts, a region will be created for method execution inside RTS [i.e stackframe / activation record]




```
class {
    void add {
    }
    p s v main {
        add()
    }
```

O.S → Jvm

→ All the local variables of that particular method will be created inside the stack frame

→ In case if a method is not returning a data, the return type should be void

Method Overloading

(1) The process of having same method name for different method is called as "method overloading"

(2) we can overcome naming conflict in method overloading by

* no. of parameters
* data type <parameters>
* order of data type

Note:

* The variables that are accepted by the method is called "parameter"

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

* the variable constants which are passed during method call are known as "arguments"

```
add(10,20)
   arguments
```

Disadv of Type promotion over method Overloading

↑ compiler implicitly type cast lower type → higher type

Error: It shows error as

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
	The method add(float, int) is ambiguous for the type Calculator

	at TypePromotionProg.main(TypePromotionProg.java:18)
```

(1) In the above program when we pass two integer values then will be a "ambigous" situation because (int, int) can be converted to both (float, int) as well as (int, float) through implicit conversion i.e type promotion

Note:

* method Overloading is also called as

poly — many

morphism — form

(1) Compile Time polymorphism

(2) Early binding

(3) false polymorphism

(4) Virtual polymorphism

(5) Static binding