7. Java Methods In Depth | Different Types of Methods with Examples

Access Specifiers:

- Public : can be access through anywhere
- Private: can be access by methods of same class only
- Protected: can be access by ① subclasses anywhere
  ② Non-subclass in same package
- Default: if we don't mention anything, ~~by~~ default access specifier is used by Java.
  Accessed by classes in the same package

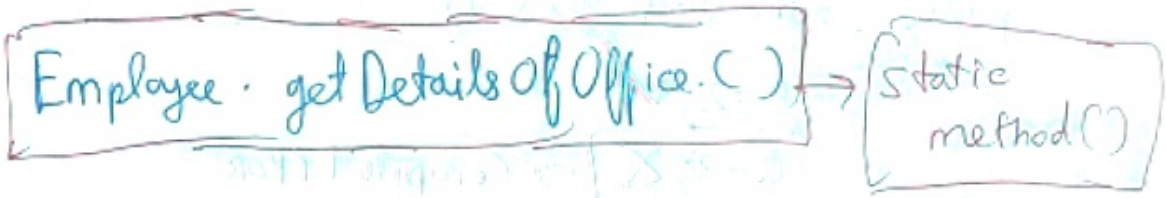
(Types of Method)

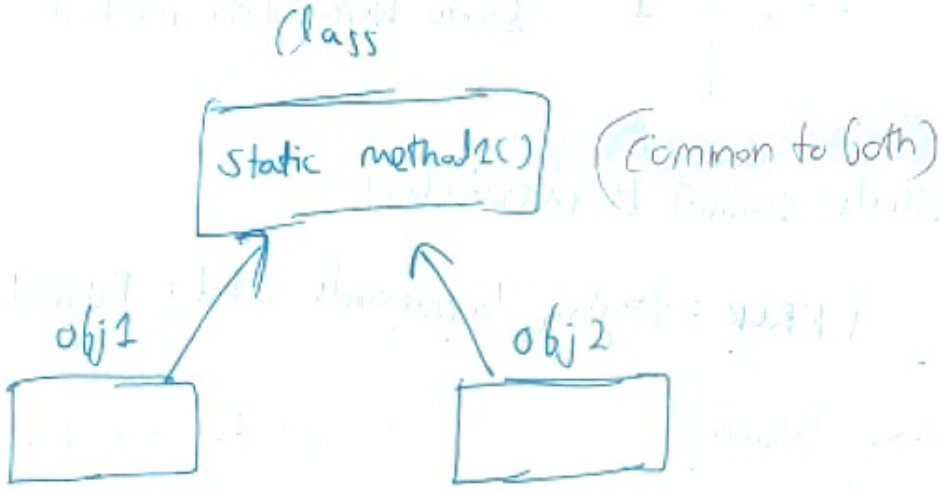
~~Static~~ [System Defined Method]:

Methods built in already like `Math.sqrt()`

[Static Method Type]

* Methods are associated with the class. Instance of the class cannot be used to call the object.





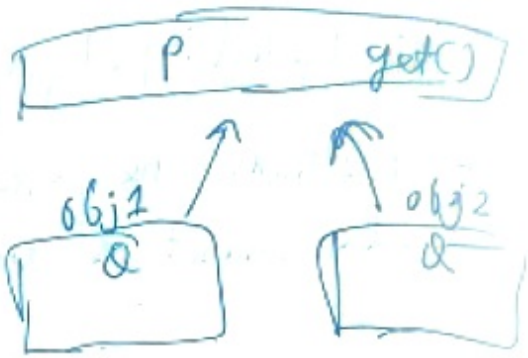
* Can be called with className

* Static method cannot access Non-static Instance variables and methods.

  ↳ Because the method is static, so it is shared among multiple instances. But it cannot have variables which are not common to class, but ~~exist~~ exclusive for Objects.

```
class A{
    static int P;
     int Q;
    public static void get(){
        P=2; ✓
        Q=3; ✗ → Compile ERROR
    }          Since Non-Static method.
```

(Static Block)



* Static methods cannot be overridden

Example — ERROR: trying to override Static Method.

```
public class Person{

    public static void profession();

}
```

```
public class Doctor extends
Person{

✗ @Override
    public static void profession()

}
```

(Why?) Because overriding is dynamic binding and static block memory allocation is compile time binding

V.V. Imp

When class extends another class. Java hides the static block of the parent class to the child class.

```
class Person{
public static void Profession()
}
```

```
class Doctor extends Person{
public static void Profession()
}
```

Static code hidden to child class

Q) When to declare method Static?

A) (1) Methods which do not modify the state of the object can be declared Static

(2) Methods which do not use any instance variables or modify them and compute on arguments only.

(Case I)

```
Static P= q+m;
static int ( int a, int b){
    return a+b;
}.
```

only Computing not changing

```
static P = 10;
static int ( int a, int b){
    P += (a+b);
    return P;
}
```

This should be avoided as multiple threads may cause inconsistency issue.

[Final Method Type].

* Cannot be overridden.

```
class A{
    public final void go(){
    }
}
```

```
class B extends A{
    public void go(){
    }
}
```

compile time error as go is a final method.

[Abstract Method Type].

* Defined in abstract Classes
* Only method declaration is done
* Implementation done in child classes

[Variable ARGUMENTS] (VAR ARGS): [Interesting].

Lets say you have a method sum (int a, int b).

But you don't know how many arguments the user going to send. In that case

```
int sum ( int ... variable ) {
    int output = 0;
    for ( int var : variable ) {
        output += var;
    }
    return output;
}
```

```
call sum (1, 2);
call sum (1, 2, 3);
call sum (1, 2, 3, 4);
```