⊛ Syntax of function :

```
[Access Specifier] [Access Modifier] <return type/void> <function name> (parameter list)
{
 Body of function
}

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

* Access Specifier

(a) Public : Public data and methods can be accessed everywhere within as wells as outside the class in which they were declared.

It can be inherited.

(b) Protected : Protected data and methods can be accessed everywhere within their class, other classes in their package and sub classes in other packages.

It can be inherited.

Packages → class → methods/data members

(c) default (friendly / Package)

(d) Private : Private data & member methods can only be accessed in the class in which they they were declared.

It is the most secure access specifiers.

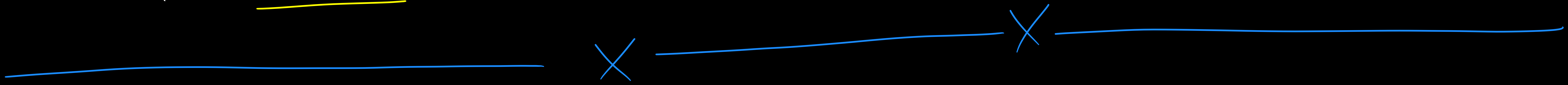
It cannot be inherited.

* Access Modifiers :

(i) **Static** : Any static method or variable is of class type and it can be directly called or accessed within any function in the same class without help of any object.

A non static function or variable is required to be called through object from any static function of the class.

(ii) final



Eg:

```
class fun
{
 public static void calc()
 {
   int a = 10;
   int b = 20;
   Sopln(a+b); → 30
 }
 public void acc()
 {
   calc();
 }
 public static void main(string args[])
 {
   Fun obj = new fun();
   obj.acc();
 }
}
```

* Formal parameters ⇒ The parameters, appear in the function definition statement are called formal parameters.

Function definition statement

Eg:

```
public static void sum(int a, int b)   → 17  → 16
{ int s = a+b; → 33
    sopln(s);
}
```

→ 33

Formal Parameters

* Actual Parameter ⇒ The parameters which appear in function calling statements are called actual parameter.

Eg:

```
public static void main (String args[ ])
{
    int x = 17; ✓
    int y = 16; ✓
    sum (x, y); → 33
    sum (10, 20); → 30
}
```

Actual Parameters

No H.W Today