Functions / Methods

* Function / Methods :-
  - A method in java is a block of code that perform a specific task
  - It is executed only when it is called.
  - Helps in code resuability and modular programming (defined code once & use it many times)

* Syantax :-

```
~~return typ~~
returnType methodName (parameter) {
    // method body or code.
    return value; (if return type is
                   not void)
}
```

returnType :- Type of value returned (int, string, void, etc)

methodName :- Name of the method.
- it should be meaningful.

parameter :- inputs. (can be 0 or more, seperated by ',' (comma))

return :- Used to return a value.
(except for void method)

Date
Page

★ Static & Non Static Method

- Static Method
  - Belong To class
  - Can be called directly without creating an object.
  - Defined using static keyword.
  - Eg:-

```
public class Myclass {
    static void show() {
        sout("Static");
    }
    psvm (string[] args) {
        Myclass.show();
    }
}
```

- Non-static Method
  - Belongs to an object
  - Needs an object before calling
  - No static keyword
  - Eg:-

```
public class Myclass {
    void show() {
        sout("Non static");
    }
    psvm (string [] args) {
        Myclass obj = new Myclass()
        obj.show();
    }}
```

- Calling function:-
  - Static

```
MethodName (Parameter);
```

  - NonStatic
    - Create an Object
    - Calling

```
ObjectName.MethodName()
          Parameters
```

Return-Type :-

- In java, the return type of a method specifies what kind of value of method will return to caller.
- A return type may be premitive type like int float or :void (return nothing).

Notes :-

- Type of data returned by a method must be compatible with return type sepecified by method
- Eg:- if return type is int then we can not return character / string.
- The variable receiving the value returned by a method must be compatable with return type specified for method.

★ Pass by Value :-

- When we pass a premitive variable (like int, float, etc) a copy of actual value is passed its method.
- So any change inside the method does not affect original variables.
- Here only value are pased original stays safe.

Date
Page

* Pass by value of Reference variable

- The value being passed is a copy of the reference (memory address) to the object.
- But it still don't passes the actual object itself.
- Changes made in that reference affect the original object.
- Datatypes - String, array, objects, ~~etc.~~

Notes:-

- String is a reference type but acts like a primitive in many way due to immutability i.e. once created, it cannot be changes. Any modification create a new string object.

[redacted]

★ Scopes :- Scope is the part of programm where variable or function can be accessed or used in ~~your~~ code.

- Types of scope :-

1) Block Scope :-

- Variables initialized out side the block can be updated inside the block & can't be initialised inside the block.

```
psvm () {
    int a = 10;
    {
        a = 20;
        int b=20;
    }
    b = 30; X
}
```

- variable initialised inside the block can't be updated outside the block

- ~~vabi~~ variable initialised inside the block is ~~accesab~~ ~~acce~~ accessible within that block & can be intialised again outside the block.

2) function Scope :-

- variable declared inside a method / function scope can't be accessed outside the method

3) Loop Scope :-

- Variable declared in for, while loop exist only withing the loop body.

4) Class Scope :-

- Variable declared inside class but outside any method/ function.
- Also called instance variable if not marked static.

Shadowing :-

Shadowing in Java is practive of using variables in overlapping scopes with same name where the variable in low-level scope override the variable of high-level scope. Here the variable at high-level is shadowed by low level scope variable.

Eg:-

```
public class shadowing {
    static int x = 10;
    psvm() {
        sout(x); //10
        x = 20;
        sout(x); //20
    }
}
```

Variable Arguments :-

- Variable Arguments is used to take a variable number of argument. A method that takes variable number of argument is varargs method.

Syantax :-

```
Static void example (int ... a){
                    // body ;
}
```

- We can also used variable arguments with multiple arguments but variable argument should be in last.

Syantax :-

```
Static void example ( int a, int b, int ... z){
                    // body ;
}
```

Function Overloading :—

Defining multiple methods with same name, but with different parameter (either in number, type or order of parameter) in the same class.

Rules of Overloading :—

- Number of parameter
- Type of parameter
- Order of parameter

Example :—

```
static void demo (string a) {
        sout (a);
    }
Static void demo (int z) {
        sout (z);
    }
```

- At compile time, it decides which function to run.

Armstrong Numbers :—

Suppose there is number → 153

$153 \rightarrow (1)^3 + (5)^3 + (3)^3 = 1 + 125 + 27$

$= 153$ Ans