SystemVerilog for Verification

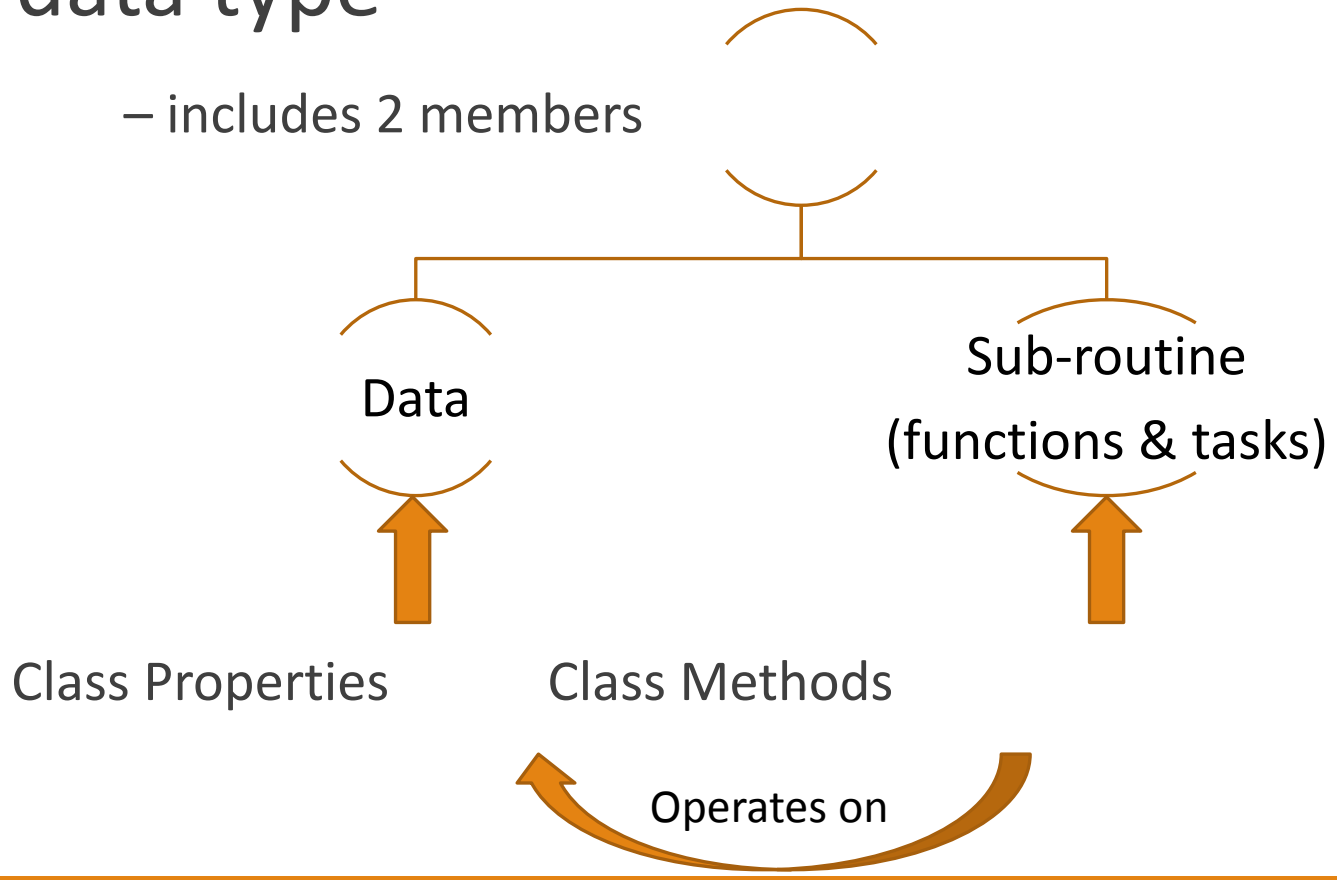
CLASS & OOPS – PART I

Agenda

- ✓ Class Basics
- ✓ Class Format
- ✓ Class Object
- ✓ Class Constructor
- ✓ Class v/s Structure
- ✓ Static Property
- ✓ Static Method

Class Basics

- class is a **data type**
  - includes 2 members



Class Format

```
class your_class_name;

    // Here : declare class properties

    // Here : define class constructor function (also used to initialize class properties)

    // Here : define functions and tasks

endclass : your_class_name
```

Class Object

- Class defines data-type. Object is instance of that class.

Dynamic

Step 1 – declare object → home h;

Step 2 – create object → h = new();

RuN TiMe MeMoRy AlLoCaTiOn

If step 2 ignored – object is uninitialized with value = null

```
if(h == null)    h=new();
```

- ‘ . ’ to access class properties and methods → h.light = 1; h.open_electricity();

Function based	Object based
Status = door_status(h);	Status = h.door_status();

Class Constructor

- Class has default built-in constructor ‘new’ method.
  - So, even if not declared, new will be there inside class by default.
- Initializer values can be overwritten by constructor.

```
class abc;
    int a = 1;
    function new()
        a = 2;
    endfunction
endclass
```

Class vs Structure

Struct	Class
typedef struct { int a; } abc; abc abc1;	**class abc { int a; } abc abc1;**
	Objects created & destroyed dynamically
Can't contain methods	contain methods → manipulates own property
Only give functionality of encapsulation	With encapsulation - inheritance, polymorphism

Looks Similar

Differences ?

Static

Memory Assigned

Dynamic

Not Yet memory assigned

Abc1 = new(); ← now assigned

Class : Static Property

- Till Now – *'instance class property'* – don't exist before creation & after destroyed

Lifetime – Automatic in dynamic class ⟷ **Scope** – accessible with instance of the class

```
class a
    static int i;
endclass  : a
```

← As int i is declared static into dynamic class – memory at static time

Accessing Static Property "using objects"	**Accessing Static Property "without using objects"**
After object creation (no need to initialize), objects share same memory of static property	Assigned memory at static time, no object needed Access static property - class resolution operator '::'
a a1, a2, a3; a1.i = 2 ; b = a2.i ; a3.i = 4;	a::i == 2;

Class : Static Method

- Same as static property – methods declared with '**static**' keyword
- can be accessed with and without object, using class resolution operator '::'
- Can access only static property of the class, as method exists before instance creation

8.6 - The lifetime of methods declared as part of a class type shall be automatic. It shall be illegal to declare a class method with a static lifetime.

static/automatic function static/automatic(1st-lifetime fo function, 2nd-lifetime of variables inside function)

Static Method (static function)	**method static (function static)**
class a; static function /*automatic*/ f1(); **static class method with automatic variable lifetime** **scope: local to class** **lifetime: function-static, variables inside function - automatic**	class b; /*automatic*/ function static f2(); //**NOT ALLOWED** HAVING FUNCTION STATIC INSIDE CLASS - 8.6 LRM //Gives warning as 8.6, but works **non-static method with static variable lifetime** **scope: local to class object** **lifetime: function-automatic, var inside-static**
class a; static function static f1(); //Gives warning as 8.6, but works **static class method with static variable lifetime** **scope: local to class** **lifetime: function-static, variables inside - static**	class b; /*automatic*/ function /*automatic*/ f2(); **scope: local to class object** **lifetime: automatic**

Thank You

- For more videos, please visit us at -

https://aspiringdirections.wordpress.com