Object Oriented Programming concept
(Concept of Classes)

What is Oops :- Object Oriented Programming.and classes from the base of Oops programming.

Encapsulation: Oops bind data and function together or bundle of properties and methods.

Inheritance: extends the functionality of existing object.means we can shared or extends properties and methods.

Polymorphism: wait untill runtime to bind data with functions.

Class:-

A class is a user-defined data type means class is datatype we are going through multiple data types like logic ,int so class is just another one data type this includes two members data (class properties), sub-routine(means functions and tasks) that operate on data.

Class has only three elements,that is easy to learn,

i. **Properties**: properties include different datatypes.
ii. **Methods**: its include task and function present inside class.
iii. **Constraints**

Class Declaration: Class consists of all three elements properties,methods,constraints in single entity.

SYNTAX:

```
Class class_name;
                    //properties
                        Datatype;
                    //methods
                       Function function_name();
                       Endfunction
                    //constraint
                        (also used to initialize class properties)
        endclass
```

Class Object:

Class define as datatype.And object is instance of class.

Step1: declare a object,

```
Sample_class s;
```

Step2: create object,

```
s = new();
```

Note: if you step to ignored-object is unitialized with value=null.

```
If(s==null)
```

How to access the class properties and method?

To access the class property and methods use dot(.)

```
class sample_class;
bit[7:0]data;
bit[1:0]addr;
endclass

module top;
        sample_class s;
initial begin
        s=new();
        s.data=2'b11;
        s.addr=2'b01;
        $display("value of data=%0d addr=%0d",s.data,s.addr);
 end
endmodule
```

This keyword in systemVerilog

Without this keyword:

this keyword is used to refer to the current class properties. Normally used within a class to refer to its own properties or methods.

This keyword use only non static class method.and this is used to refer to the object handle eg:

```
Class packet; //wthout this method

                bit [7:0]data;

                bit[7:0]addr;

                Function void set(bit [7:0]data,addr);

                    data=data; //compiler gets confused on which is
            the class variable

                addr=addr;

        Endclass
```

With this keyword:

```
Class packet; //using this keyword correct method

                bit [7:0]data;

                bit[7:0]addr;

                Function void set(bit [7:0]data,addr);

                    this.data=data;

                      this.addr=addr;

        Endclass
```

```
class packet;
  bit [31:0] addr;
  bit [31:0] data;
  bit   wr_rd;

  function new(bit [31:0] addr,data,bit write);
    this.addr  = addr;
   this.data  = data;
   this.wr_rd = wr_rd;
  endfunction
  function void print();
    $display("\n addr  = %0h",addr);
    $display(" data  = %0h",data);
    $display(" wr_rd = %0h",wr_rd);
  endfunction
endclass

module top;
  packet pkt;
  initial begin
    pkt = new(32'h10,32'hFF,1'b1);
    pkt.print();
  end
endmodule
```

```
# Top level modules:
#       top
# End time: 14:55:34 on Sep 14,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# End time: 14:55:35 on Sep 14,2023, Elapsed time: 0:01:08
# Errors: 0, Warnings: 0
# vsim top
# Start time: 14:55:35 on Sep 14,2023
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.this2_sv_unit(fast)
# Loading work.top(fast)
#
#  addr  = 10
#  data  = ff
#  wr_rd = 0
```

Static class Properties and Method:

- Normal Properties are dynamic.
- A static Property is shared by all class instance.you can declare as properties being statics which means one copy of property which is shared by all of the instances of class.
- Static class properties and methods can be used without creating an object of that type.

Syntax

Static datatype_name property_name;

eg:-Static int count;

As int count is declared static into dynamic class-memory at static time.

Static method:

1. Same as static property -> method declared with **static** keywords.
2. Can be access with or without object,using class resolution operator ::
3. Static method can be access only static property of the class.

```
class packet;
bit[2:0] packet_count;
static int count;
function new();
        count++;
        packet_count=count;
endfunction
function void print();
        $display("packet are=%0d",packet_count);
endfunction
endclass

module top;
packet pkt[3];
initial begin
        foreach(pkt[i])begin
        pkt[i]=new();
        pkt[i].print();
        end
end
endmodule
```

Inheritance in SystemVerilog

- Inheritance is an oops concept that allows the user to create classes that are built upon existing classes
- New classes are created by inheriting properties and methods defined in an existing class.
- Existing class is called the base class(parent class),and the new class is referred to as the derived class(child class).
- Inheritance allows user to create classes which are derived from other classes.
- The derived class(child class) inherits all the properties and methods defined in base class(parent class).

Example: `class Packet` **// (Base Class)**

```
//properties
//methods
endclass
```

Class `packet1 extends packet` **//(extended class)**

```
//properties
//methods

endclass
```

```
class packet;
        bit [31:0]addr;
endclass

class packet1 extends packet;
        bit[31:0]data;
endclass

module top;
initial begin
        packet1 pkt=new();
        pkt.addr=50;
        pkt.data=30;
        $display("value of addr=%0d data=%0d",pkt.addr,pkt.data);
end
endmodule
```

```
#
# Top level modules:
#       top
# End time: 07:07:30 on Sep 16,2023, Elapsed time: 0:00:01
# Errors: 0, Warnings: 2
# vsim top
# Start time: 07:07:31 on Sep 16,2023
# ** Note: (vsim-3812) Design is being optimized...
# ** Warning: inheritance.sv(11): (vopt-2244) Variable 'pkt' is implicitly static. You must either explicitly declare it as static or automatic
# or remove the initialization in the declaration of variable.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading sv_std.std
# Loading work.inheritance_sv_unit(fast)
# Loading work.top(fast)
# value of addr=50 data=30

VSIM 4>
```

```
class sample_class;
int a;
bit[3:0]b;
endclass

class child_class extends sample_class;
string subject;

function new();
        a=10;
        b=4'b1001;
        subject="System_verilog";
endfunction
function void print();
        $display("\n\ta=%0d \n\tb=%0b \n\tsuject=%s",a,b,subject);
endfunction
endclass

module top;
initial begin
        child_class c=new();
        c.print();
end
endmodule
```

```
#
# Top level modules:
#       top
# End time: 07:30:00 on Sep 16,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 2
# End time: 07:30:02 on Sep 16,2023, Elapsed time: 0:01:55
# Errors: 0, Warnings: 1
# vsim top
# Start time: 07:30:02 on Sep 16,2023
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# ** Warning: inheritance2.sv(23): (vopt-2244) Variable 'c' is implicitly static. You must either explicitly declare it as static or automatic
# or remove the initialization in the declaration of variable.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading sv_std.std
# Loading work.inheritance2_sv_unit(fast)
# Loading work.top(fast)
#
#       a=10
#       b=1001
#       suject=System_verilog
```