

DAY-94 #100DAYSRTL

"UVM: UVM Object(Part-2)"

"Inbuild Implementation: Copy":-

Shallow vs Deep copy:-

- In the case of Deep Copy we have independent instances of the class
- In a Shallow copy we have single instances that are accessible from all the instances that copy the original class.

"Code Practising":-

```
include "uvm_macros.svh"
import uvm_bkg::%;
class temp extends wum_object;
bit [3:0] data_temp;
function new(string inst = "temp");
endfunction
'uvm_object_wiils_begin(temp)
'uvm_field_int(data_temp, UMM_DEFAULT)
'uvm_object_wiils_ind
'include "uvm_sequence_item;
rand bit [3:0] data;
temp t;
function new(string inst = "transaction");
include "uvm_sequence_item;
rand bit [3:0] data;
temp t;
function new(string inst = "transaction");
include "uvm_object_uviils_begin(transaction)
'uvm_object_uviils_begin(transaction)
'uvm_object_uviils_end
'uvm_object_uviils_end
modclass
module tb;
transaction tran, tr_b;
infill "uvm_object_uviils_end
endclass
module tb;
transaction transaction to both instances
tr_a new("tr_a");
it = b new("tr_b");
it = b new("tr_
```

"Result":-

Name			Value
	transaction		@335
data	integral		'h6
			@338
data_temp			
Name	Туре	Size	Value
b	transaction		@339
	integral		'h6
	-		@341
data_temp		4	'h3
Name		Size	
	transaction		@335
	integral	4	'h6
t	temp	-	@338
data_temp	integral	4	'h3
Name		Size	
	21		
tr_b	transaction	-	@339
data	integral	4	'h6
	temp		@341
data temp	integral	4	'h0

"Do Hook: Copy":-

"Code Practising":-

```
import uvm_pkg::*;
class temp extends uvm_object;
  uvm_object_utils(temp)
bit [3:0] data_temp;
function new(string inst = "temp");
super.new(inst);
endfunction
   virtual function void do_print(uvm_printer printer);
   super.do_print(printer);
printer.print_field("data_temp", data_temp, $bits(data_temp), UVM_DEC);
endfunction
endclass.
class transaction extends uvm_sequence_item;
 `uvm_object_utils(transaction)
rand bit [3:0] data;
 temp t;
function new(string inst = "transaction");
function new(string inst = transaction);
super.new(inst);
t = new("t");
endfunction
  virtual function void do_print(uvm_printer printer);
  super.do_print(printer);
printer.print_field("data", data, $bits(data), UVM_DEC);
printer.print_object("t", t);
   virtual function void do_copy(uvm_object rhs);
      transaction tr;
      super.do_copy(rhs);
$cast(tr, rhs);
data = tr.data;
   t.data_temp = tr.t.data_temp;
endfunction
endclass
module tb;
transaction tr_a, tr_b;
initial begin
//////adding constructor to both instances
  tr_a = new("tr_a");
tr_b = new("tr_b");
   //////////generate random data for one of instance
tr_a.randomize();
   tr_a.t.data_temp = 4'b0011;
   tr_a.print();
/////////////copy the content of instance a to b
   tr_b.print();
   tr_b.print();
/////////////update the content from any one instance
tr_b.t.data_temp = 4'b0000;
   tr_a.print();
tr_b.print();
endmodule
```

"Result":-

```
Type Size Value
       transaction -
            integral 4 'h6
  temp -
data_temp integral 4
                              @338
            Type
                   Size Value
Name
tr_b transaction -
data integral 4
t temp -
                            @339
                              @341
  data_temp integral 4 'h3
Name
            Type
                        Size Value
tr_a transaction - @335
data integral 4 'h6
t temp - @338
   data_temp integral
                        Size Value
tr_b transaction - @339 data integral 4 'h6
                             @339
            temp
                              @341
   data_temp integral 4
                              'h0
```

"Create Method":-

"Code Practising":-

```
`include "uvm_macros.svh"
 import uvm_pkg::*;
////all the classes -> create
////tlm ports -> new
class transaction extends uvm_sequence_item;
 rand bit [3:0] a;
 rand bit [3:0] b;
      bit [4:0] y;
 function new(input string inst = "transaction");
 super.new(inst);
 endfunction
uvm_object_utils_begin(transaction)
  `uvm_field_int(a,UVM_DEFAULT)
  `uvm_field_int(b,UVM_DEFAULT)
  `uvm_field_int(y,UVM_DEFAULT)
`uvm_object_utils_end
endclass
module tb;
 transaction tr;
 initial begin
   tr = transaction::type_id::create("tr");
   tr.randomize();
   tr.print();
 end
endmodule
```

"Result":-

```
Name Type Size value

tr transaction - @335
 a integral 4 'h6
 b integral 4 'h5
 y integral 5 'h0

Simulation has finished. There are no more test vectors to simulate.
```