VERIFICATION ACADEMY

Advanced OVM (& UVM)

Understanding the Factory

Tom Fitzpatrick
Verification Technologist

academy@mentor.com
www.verificationacademy.com

Mentor Graphics®

Create() vs. New()

- **Objects must be constructed**
- **`new()` hard-codes the type**
- **`create()` returns a constructed instance from the factory**
- **The factory lets you change the type of the created component**
  - Without modifying the instantiating code

```
class my_env extends ovm_env;
  virtual function void build();
    comp1 = new("comp1", this);
    comp2 = my_comp::type_id::create("comp2",
                                     this);
  endfunction
```

```
class my_comp extends ovm_component;
  `ovm_component_utils(my_comp)
  …
endclass
```
comp1

```
class my_Xcomp extends my_comp;
  `ovm_component_utils(my_Xcomp)
  …
endclass
```
comp2

Registering with the Factory

- **Objects are registered with the factory via macro**
  - `ovm_object_utils
  - `ovm_component_utils

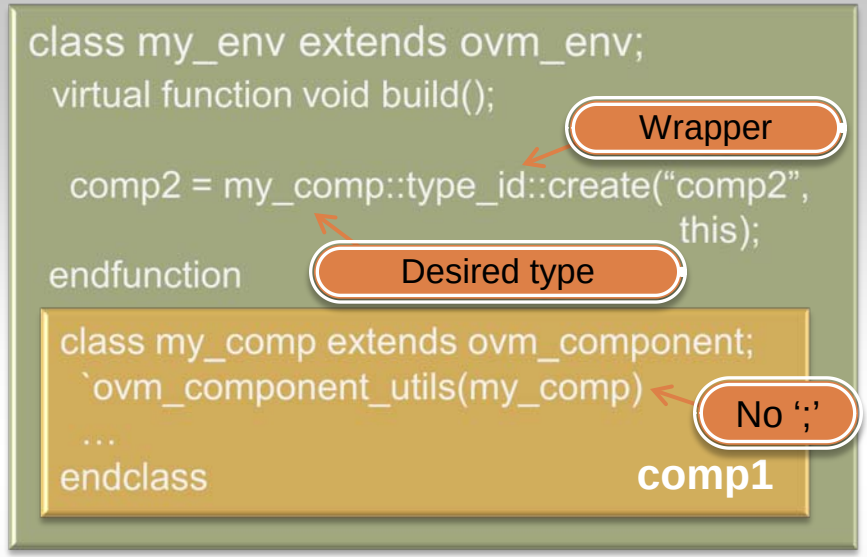
```
class my_comp extends ovm_component;
  `ovm_component_utils(my_comp)
  …
endclass
```

No ';'

comp1

Registering with the Factory

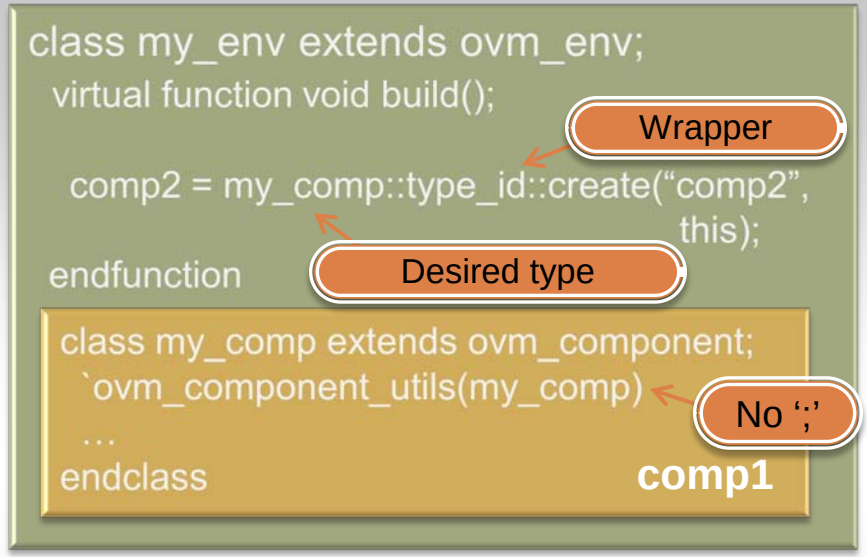
- **Objects are registered with the factory via macro**
  - `ovm_object_utils
  - `ovm_component_utils
- **Macro creates ‘type_id’ wrapper**
- **Static methods in wrapper**
  - create(): returns an instance of the type (no $cast needed)
  - get_type(): returns the type “handle”



Registering with the Factory

- **Objects are registered with the factory via macro**
  - `ovm_object_utils
  - `ovm_component_utils
- **Macro creates 'type_id' wrapper**
- **Static methods in wrapper**
  - create(): returns an instance of the type (no $cast needed)
  - get_type(): returns the type "handle"
  - set_type_override()
  - set_inst_override()

  } Factory type overrides



Overriding a Type

```
class test  extends ovm_test;
  function void build();
    my_env e = new("e");



  endfunction
endclass
```

```
class my_env extends ovm_env;
  `ovm_component_utils(my_env)
  shape u1,u2;// default square

  function void build();
    u1 = shape::type_id::create("u1",this);
    u2 = shape::type_id::create("u2",this);

  …
endfunction
```

U1 U2

Overriding a Type

```
class test  extends ovm_test;
  function void build();
    my_env e = new("e");
    shape::type_id::set_type_override( circle::get_type() );


  endfunction
endclass
```

New Desired type

```
class my_env extends ovm_env;
  `ovm_component_utils(my_env)
  shape u1,u2;// default square

  function void build();
    u1 = shape::type_id::create("u1",this);
    u2 = shape::type_id::create("u2",this);

  …
endfunction
```

U1 U2

All Instances Overridden

Overriding an Instance

```
class test  extends ovm_test;
  function void build();
    my_env e = new("e", this);
    shape::type_id::set_type_override( circle::get_type() );
    shape::type_id::set_inst_override( triangle::get_type(), "e.u2" );

  endfunction
endclass
```

New Desired type

Instance Name

```
class my_env extends ovm_env;
  `ovm_component_utils(my_env)
  shape u1,u2;// default square

  function void build();
    u1 = shape::type_id::create("u1",this);
    u2 = shape::type_id::create("u2",this);

  …
endfunction
```

Instance Changed

Using Parameterized Types

```
class test  extends ovm_test;
  function void build();
    my_env e = new("e", this);
    shape::type_id::set_type_override( circle::get_type() );
    shape::type_id::set_inst_override( triangle::get_type(), "e.u2" );

  endfunction
endclass
```

```
class red #(int SIDES=3)
            extends ovm_component;
  `ovm_component_param_utils(red#(SIDES))
```

```
class my_env extends ovm_env;
  `ovm_component_utils(my_env)
  shape u1,u2;// default squ
  red #(4) u3;
  function void build();
    u1 = shape::type_id::create("u1",this);
    u2 = shape::type_id::create("u2",this);
    u3 = red#(4)::type_id::create("u3",this);
  …
endfunction
```

Parameterized type

Using Parameterized Types

```
class test  extends ovm_test;
  function void build();
    my_env e = new("e", this);
    shape::type_id::set_type_override( circle::get_type() );
    shape::type_id::set_inst_override( triangle::get_type(), "e.u2" );
    red#(4)::type_id::set_type_override( blue#(4)::get_type() );
  endfunction
endclass
```

```
class blue #(int SIDES=3)
              extends red #(SIDES);
  `ovm_component_param_utils(blue#(SIDES))
```

```
class my_env extends ovm_env;
  `ovm_component_utils(my_env)
  shape u1,u2;// default square
  red #(4) u3;
  function void build();
    u1 = shape::type_id::create("u1",this);
    u2 = shape::type_id::create("u2",this);
    u3 = red#(4)::type_id::create("u3",this);
  …
 endfunction
```

Environments are Components

```
class test  extends ovm_test;
  function void build();
    my_env e = my_env::type_id::create("e", this);
    shape::type_id::set_type_override( circle::get_type() );
    shape::type_id::set_inst_override( triangle::get_type(), "e.u2" );
    red#(4)::type_id::set_type_override( blue#(4)::get_type() );
  endfunction
endclass
```

```
class my_env extends ovm_env;
  `ovm_component_utils(my_env)
  shape u1,u2;// default square
  red #(4) u3;
  function void build();
    u1 = shape::type_id::create("u1",this);
    u2 = shape::type_id::create("u2",this);
    u3 = red#(4)::type_id::create("u3",this);
  …
 endfunction
```

Tests are Components, too!

- **run_test() creates the test from the factory**

```
module top;
  ...

  initial
  begin: blk
    ...
    run_test();
  end

endmodule: top
```

```
class test extends ovm_test;
  `ovm_component_utils(test)
...
endclass
```

Register the test with the factory

Command line: `vsim +OVM_TESTNAME=test`

Tests are Components, too!

- **Always call run_test() with null argument**

```
module top;
  ...

  initial
  begin: blk
    ...
    run_test();
  end

endmodule: top
```

```
class test extends ovm_test;
  `ovm_component_utils(test)
...
endclass
```

```
class test2 extends ovm_test;
  `ovm_component_utils(test2)
...
endclass
```

Command line: `vsim +OVM_TESTNAME=test2`

Use Factory on Sequence Items

- **Sequence Items may also be overridden**

```
class my_seq extends
    ovm_sequence #(my_tr);
  …
  task body();
    my_tr tx;
    tx = my_tr::type_id::create("tx"
    …
  endtask
endclass
```

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

Register with Factory

Use Factory on Sequence Items

- **Sequence Items may also be overridden**

```
class my_seq extends
    ovm_sequence #(my_tr);
  …
  task body();
    my_tr tx;
    tx = my_tr::type_id::create("tx", ,get_fullname());
  …
  endtask
endclass
```

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

Register with Factory

Empty parent for objects

Allow factory to find this sequence

Use Factory on Sequence Items

- **Sequence Items may also be overridden**

```
class my_seq extends
    ovm_sequence #(my_tr);
  …
  task body();
    my_tr tx;
    tx = my_tr::type_id::create("tx", ,get_fullname());
    start_item(tx);
    assert( tx.randomize() with { cmd == 0; } );
    finish_item(tx);
    …
  endtask
endclass
```

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

Register with Factory

Use Factory on Sequence Items

- **Override items by type**

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

```
class err_tr extends my_tr;
  `ovm_object_utils(err_tr)
…
endclass
```

```
class err_test extends my_test;
  …
  task run();
    my_tr::type_id::set_type_override(err_tr::get_type());

    …
  endtask
endclass
```

Use Factory on Sequence Items

- **Override items by type or instance**

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

```
class err_tr extends my_tr;
  `ovm_object_utils(err_tr)
…
endclass
```

```
class err_test extends my_test;
  …
  task run();
    my_tr::type_id::set_inst_override(err_tr::get_type(),
                                      “e.agent.sqr.s1.tx”);
    …
  endtask
endclass
```

Use Factory on Sequence Items

- **Override items by type or instance**

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

```
class err_tr extends my_tr;
  `ovm_object_utils(err_tr)
…
endclass
```

```
class err_test extends my_test;
  …
  task run();
    my_tr::type_id::set_inst_override(err_tr::get_type(),
                                      "e.agent.sqr.s1.tx");
    …
  endtask
endclass
```

Path to sequencer

Use Factory on Sequence Items

- **Override items by type or instance**

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

```
class err_tr extends my_tr;
  `ovm_object_utils(err_tr)
…
endclass
```

```
class err_test extends my_test;
  …
  task run();
    my_tr::type_id::set_inst_override(err_tr::get_type(),
                                      "e.agent.sqr.s1.tx");
    …
  endtask
endclass
```

Path to sequencer

Sequence name

Use Factory on Sequence Items

- **Override items by type or instance**

```
class my_tr extends ovm_sequence_item;
  `ovm_object_utils(test)
…
endclass
```

```
class err_tr extends my_tr;
  `ovm_object_utils(err_tr)
…
endclass
```

```
class err_test extends my_test;
  …
  task run();
    my_tr::type_id::set_inst_override(err_tr::get_type(),
                                      "e.agent.sqr.s1.tx");
    …
  endtask
endclass
```

Path to sequencer

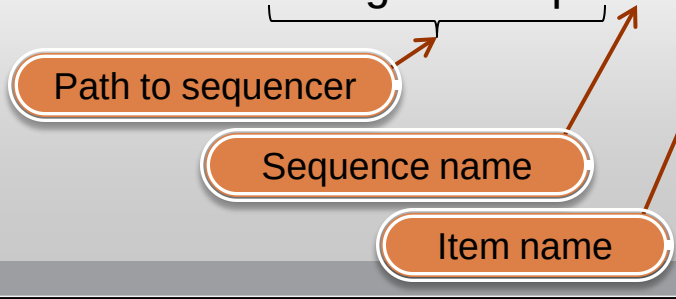
Sequence name

Item name

Use Factory on Sequence Items

- **Instance overrides let you change the type of item generated by**
  - a particular sequence on a sequencer
  - a specific item name created by a sequence

```
class err_test extends my_test;
  …
  task run();
    my_tr::type_id::set_inst_override(err_tr::get_type(),
                                      "e.agent.sqr.s1.tx");
    …
  endtask
endclass
```



Summary

- **Use `ovm_object/component_utils macro to register with the factory**
- **Always call <type>::type_id::create()**
- **Register tests with the factory**
- **Call run_test() with null argument**
  - Specify which test via the command line

VERIFICATION ACADEMY

Advanced OVM (& UVM)

Understanding the Factory

Tom Fitzpatrick
Verification Technologist

academy@mentor.com
www.verificationacademy.com

Mentor Graphics®