```
// file1 : user_defined.sv

// User-defined phases, added to the predefined run-time phase schedule

`include "uvm_macros.svh"

package pkg;
  import uvm_pkg::*;

// Step : 1) Define a base class that declares the user-defined phase method

  class extended_component extends uvm_component;

    function new (string name, uvm_component parent);
      super.new(name, parent);
    endfunction

    virtual task karthi_phase(uvm_phase phase);
    endtask

  endclass

// Step : 2) Define the user-defined phase class

  class karthi_user_phase extends uvm_task_phase; // User-defined phase class

    static local karthi_user_phase m_singleton_inst;

    protected function new (string name = "");
      super.new(name);
    endfunction

    // Define a static function to return a singleton instance of this class
    // Will be called below to identify the phase when inserting it into a schedule
    static function karthi_user_phase get;
      if (m_singleton_inst == null)
        m_singleton_inst = new("karthi_user_phase");
      return m_singleton_inst;
    endfunction

    // A phase is a functor with behavior defined by its exec_task method
    // exec_task is called implicitly when the phase is entered (depending on the sc
hedule)
    task exec_task(uvm_component comp, uvm_phase phase); // (args are wrong in Class
 Reference)
      extended_component c;
      if ($cast(c, comp))
        c.karthi_phase(phase);  // Call the overridden user-defined phase task
    endtask

  endclass

// Step : 3) Extend any component that is to have the user-defined phase from the ne
w base class

  class env extends extended_component;
    `uvm_component_utils(env)

    int m_delay;

    function new (string name, uvm_component parent);
      super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
      if (!uvm_config_db#(int)::get(this, "", "delay", m_delay))
        `uvm_fatal("", "Delay missing from config db")
    endfunction

    task reset_phase(uvm_phase phase);
      `uvm_info("reset", "reset_phase called", UVM_MEDIUM)
      phase.raise_objection(this);
      #(m_delay);
```

```
      phase.drop_objection(this);
      `uvm_info("rest", "reset_phase returning", UVM_HIGH)
    endtask

    task configure_phase(uvm_phase phase);
      `uvm_info("configure", "configure_phase called", UVM_MEDIUM)
      phase.raise_objection(this);
      #(m_delay);
      phase.drop_objection(this);
      `uvm_info("configure", "configure_phase returning", UVM_HIGH)
    endtask

// Step : 4) Override the user-defined phase method

    task karthi_phase(uvm_phase phase);
      `uvm_info("karthi", "karthi_phase called", UVM_MEDIUM)
      phase.raise_objection(this);
      #(m_delay);
      phase.drop_objection(this);
      `uvm_info("karthi", "karthi_phase returning", UVM_HIGH)
    endtask

    task main_phase(uvm_phase phase);
      `uvm_info("main", "main_phase called", UVM_MEDIUM)
      phase.raise_objection(this);
      #(m_delay);
      phase.drop_objection(this);
      `uvm_info("main", "main_phase returning", UVM_HIGH)
    endtask

    task shutdown_phase(uvm_phase phase);
      `uvm_info("shutdown", "shutdown_phase called", UVM_MEDIUM)
      phase.raise_objection(this);
      #(m_delay);
      phase.drop_objection(this);
      `uvm_info("shutdown", "shutdown_phase returning", UVM_HIGH)
    endtask

  endclass

  class test extends uvm_test;
    `uvm_component_utils(test)

    function new (string name, uvm_component parent);
      super.new(name, parent);
    endfunction

    env m_env;

    function void build_phase(uvm_phase phase);
      uvm_phase schedule;

      uvm_config_db#(int)::set(this, "m_env", "delay", 1);
      m_env = env::type_id::create("m_env", this);

// Step : 5) Insert the user-defined phase into the UVM Run-Time Schedule

      // Any new phases need to be added before entering the run_phase
      schedule = uvm_domain::get_uvm_schedule(); // The predefined UVM Run-Time sche
dule
      schedule.add(karthi_user_phase::get(), .after_phase(uvm_configure_phase::get()
),
                                             .before_phase(uvm_main_phase::get()));
      // The user-defined phase is here being added to the predefined Run-Time Domai
n, not to any user-defined domain.
      // This means we do not have the flexibility to synchronize phases across doma
ins

    endfunction

    function void start_of_simulation_phase(uvm_phase phase);
      this.set_report_verbosity_level_hier(UVM_HIGH);
    endfunction
```

```
  endclass
endpackage

// file2 : top.sv

module top;

  import uvm_pkg::*;
  //import pkg::*;

  initial
    run_test("test");

endmodule

// file3 : makefile

SVTB1= ./top.sv
INC = +incdir+./user_defined.sv
SVTB2 = ./user_defined.sv

#sv_cmp          =>  Create library and compile the code.
#run_test        =>  clean, compile & run the simulation for sub-run & user-defined p
hase.

sv_cmp:
        vcs -l vcs.log -timescale=1ns/1ns -sverilog -ntb_opts uvm -debug_access+all
-full64 $(INC) $(SVTB1) $(SVTB2) +define+UVM_REPORT_DISABLE_FILE_LINE

run_test:       clean  sv_cmp
        ./simv -a test.log +ntb_random_seed_automatic #+UVM_TESTNAME=test

//////////////// RESULT

UVM_INFO @ 0: reporter [RNTST] Running test test...
UVM_INFO @ 0: uvm_test_top.m_env [reset] reset_phase called
UVM_INFO @ 1: uvm_test_top.m_env [rest] reset_phase returning
UVM_INFO @ 1: uvm_test_top.m_env [configure] configure_phase called
UVM_INFO @ 2: uvm_test_top.m_env [configure] configure_phase returning
UVM_INFO @ 2: uvm_test_top.m_env [karthi] karthi_phase called
UVM_INFO @ 3: uvm_test_top.m_env [karthi] karthi_phase returning
UVM_INFO @ 3: uvm_test_top.m_env [main] main_phase called
UVM_INFO @ 4: uvm_test_top.m_env [main] main_phase returning
UVM_INFO @ 4: uvm_test_top.m_env [shutdown] shutdown_phase called
UVM_INFO @ 5: uvm_test_top.m_env [shutdown] shutdown_phase returning


--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   11
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[configure]     2
[karthi]     2
[main]     2
[reset]     1
[rest]     1
[shutdown]     2
$finish called from file "/home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/
uvm_root.svh", line 437.
$finish at simulation time                   5
```