```
--------------------------------------
-- Heating_Controller5A.adb
--------------------------------------
with Ada.Real_Time, Shared_Object1ImplPkg, Ada.Integer_Text_IO, Ada.Text_IO;
use Ada.Real_Time, Shared_Object1ImplPkg, Ada.Integer_Text_IO, Ada.Text_IO;

procedure Heating_Controller5A is
 shared_Object1ImplInst: Shared_Object1Impl;

 task Display_Update_Task1Impl is
  pragma Priority(5);
 end Display_Update_Task1Impl;



-- TASK Display_Update_Task1Impl

 task body Display_Update_Task1Impl is
  Max : constant := 95;
  Min : constant := 5;
  cttm1 : Integer := 0;
  ttm : Integer := 0;
  sinc_flag : Boolean := FALSE;
  sdec_flag : Boolean := FALSE;
  period: constant Time_Span := To_Time_Span(0.5);
  nextTime: Time := clock + period;

 begin
  loop
   delay until nextTime;
   shared_object1implInst.SOGet_Target_Temperature1(cttm1);
   Envir1Impl.ENSense_PressIncrease_Target_Temperature(sinc_flag);
   if(cttm1 < Max and sinc_flag = TRUE) then
      ttm := (cttm1 + 1);
      sinc_flag := FALSE;
   else
       sinc_flag := FALSE;
   end if;
   Envir1Impl.ENSense_PressDecrease_Target_Temperature(sdec_flag);
   if(sdec_flag = TRUE and cttm1 > Min) then
      ttm := (cttm1 - 1);
      sdec_flag := FALSE;
   else
       sdec_flag := FALSE;
   end if;
   shared_object1implInst.SOSet_Target_Temperature(ttm);
   Envir1Impl.ENDisplay_Target_Temperature(ttm);
   nextTime := nextTime + period;
  end loop;
 end Display_Update_Task1Impl;

 task Envir1Impl is
  pragma Priority(5);
   entry ENSense_PressIncrease_Target_Temperature(state_inc: out Boolean);
   entry ENSense_PressDecrease_Target_Temperature(state_dec: out Boolean);
   entry ENDisplay_Target_Temperature(tm_tt: in Integer);
   entry ENSense_Temperatures(t1: out Integer; t2: out Integer);
   entry ENDisplay_Current_Temperature(tm_avt: in Integer);
```

```
    entry ENActuate_Heat_Source(state_hsc: in Boolean);
    entry ENActuate_OverHeat_Alram(state_ota: in Boolean);
    entry ENSense_Heater_Status(state_hss: out Boolean);
    entry ENActuate_NoHeat_Alarm(state_nha: in Boolean);
  end Envir1Impl;


-- TASK Envir1Impl

  task body Envir1Impl is
    Max : constant := 95;
    Min : constant := 5;
    inc_flag : Boolean := FALSE;
    dec_flag : Boolean := FALSE;
    ts1 : Integer := 0;
    ts2 : Integer := 0;
    hss : Boolean := FALSE;
    anha : Boolean := FALSE;
    ttd : Integer := 25;
    hsa : Boolean := FALSE;
    ctd : Integer := 20;
    aota : Boolean := FALSE;
    period: constant Time_Span := To_Time_Span(0.1);
    nextTime: Time := clock + period;

    procedure ENAlter_Temperature_Sensor1 is
    begin
      ts1 := (ts1 + 1);
    end;

    procedure ENAlter_Temperature_Sensor2 is
    begin
      ts2 := (ts2 + 1);
    end;

    procedure ENAlter_Heater_Status is
    begin
      hss := FALSE;
    end;

  begin
    loop
      delay until nextTime;
      if(inc_flag = FALSE) then
         inc_flag := TRUE;
      else  if(inc_flag = TRUE) then
              inc_flag := FALSE;
            end if;
      end if;
      if(dec_flag = FALSE) then
         dec_flag := TRUE;
      else  if(dec_flag = TRUE) then
              dec_flag := FALSE;
            end if;
      end if;
      ENAlter_Heater_Status;
      ENAlter_Temperature_Sensor1;
      put("current temperature: "); put(ctd); New_Line;
```

```
    select
     accept ENSense_PressIncrease_Target_Temperature(state_inc: out Boolean) do
           state_inc := inc_flag;
     end ENSense_PressIncrease_Target_Temperature;
    or
     accept ENSense_PressDecrease_Target_Temperature(state_dec: out Boolean) do
           state_dec := dec_flag;
     end ENSense_PressDecrease_Target_Temperature;
    or
     accept ENDisplay_Target_Temperature(tm_tt: in Integer) do
           ttd := tm_tt;
     end ENDisplay_Target_Temperature;
    or
     accept ENSense_Temperatures(t1: out Integer; t2: out Integer) do
           t2 := ts2;
     t1 := ts1;
     end ENSense_Temperatures;
    or
     accept ENDisplay_Current_Temperature(tm_avt: in Integer) do
           ctd := tm_avt;
     end ENDisplay_Current_Temperature;
    or
     accept ENActuate_Heat_Source(state_hsc: in Boolean) do
           hsa := state_hsc;
     end ENActuate_Heat_Source;
    or
     accept ENActuate_OverHeat_Alram(state_ota: in Boolean) do
           aota := state_ota;
     end ENActuate_OverHeat_Alram;
    or
     accept ENSense_Heater_Status(state_hss: out Boolean) do
           state_hss := hss;
     end ENSense_Heater_Status;
    or
     accept ENActuate_NoHeat_Alarm(state_nha: in Boolean) do
           anha := state_nha;
     end ENActuate_NoHeat_Alarm;
    else
     null;
    end select;
    nextTime := nextTime + period;
  end loop;
end Envir1Impl;

task Heater_Monitor_Task1Impl is
  pragma Priority(5);
end Heater_Monitor_Task1Impl;


-- TASK Heater_Monitor_Task1Impl

task body Heater_Monitor_Task1Impl is
  Max : constant := 95;
  Min : constant := 5;
  hs1 : Boolean := FALSE;
  nha : Boolean := FALSE;
  shs : Boolean := FALSE;
  period: constant Time_Span := To_Time_Span(0.25);
```

```
  nextTime: Time := clock + period;

 begin
  loop
   delay until nextTime;
   Envir1Impl.ENSense_Heater_Status(shs);
   shared_object1implInst.SOGet_Stored_HeatSource(hs1);
   if(hs1 /= shs) then
      nha := TRUE;
   else
       nha := FALSE;
   end if;
   Envir1Impl.ENActuate_NoHeat_Alarm(nha);
   nextTime := nextTime + period;
  end loop;
 end Heater_Monitor_Task1Impl;

 task Temp_Ctrl_Task1Impl is
  pragma Priority(5);
 end Temp_Ctrl_Task1Impl;


-- TASK Temp_Ctrl_Task1Impl

 task body Temp_Ctrl_Task1Impl is
  Max : constant := 95;
  Min : constant := 5;
  stm1 : Integer := 0;
  stm2 : Integer := 0;
  avt : Integer := 0;
  cttm2 : Integer := 0;
  hsc : Boolean := FALSE;
  ota : Boolean := FALSE;
  period: constant Time_Span := To_Time_Span(0.25);
  nextTime: Time := clock + period;

  procedure TCCalculate_Average_Temperature is
  begin
   avt := ((stm1 + stm2) / 2);
  end;

 begin
  loop
   delay until nextTime;
   Envir1Impl.ENSense_Temperatures(stm1, stm2);
   TCCalculate_Average_Temperature;
   Envir1Impl.ENDisplay_Current_Temperature(avt);
   shared_object1implInst.SOGet_Target_Temperature2(cttm2);
   if(avt < cttm2) then
      hsc := TRUE;
   else
       hsc := FALSE;
   end if;
   shared_object1implInst.SOSet_Heat_Source_State(hsc);
   Envir1Impl.ENActuate_Heat_Source(hsc);
   if(avt > Max) then
      ota := TRUE;
   else
```

```
          ota := FALSE;
      end if;
      Envir1Impl.ENActuate_OverHeat_Alram(ota);
      nextTime := nextTime + period;
    end loop;
  end Temp_Ctrl_Task1Impl;

begin
  null;
end Heating_Controller5A;


----------------------------------------
-- Shared_Object1Impl.ads
----------------------------------------
package Shared_Object1ImplPkg is

  protected type Shared_Object1Impl is
    procedure SOGet_Target_Temperature1(tm: out Integer);
    procedure SOSet_Target_Temperature(tm: in Integer);
    procedure SOGet_Target_Temperature2(tm: out Integer);
    procedure SOSet_Heat_Source_State(state: in Boolean);
    procedure SOGet_Stored_HeatSource(state: out Boolean);
  private
    ctm : Integer := 0;
    shss : Boolean := FALSE;
    cttm : Integer := 0;
  end Shared_Object1Impl;
end Shared_Object1ImplPkg;

----------------------------------------
-- Shared_Object1Impl.adb
----------------------------------------
package body Shared_Object1ImplPkg is
  protected body Shared_Object1Impl is
    procedure SOGet_Target_Temperature1(tm: out Integer) is
    begin
      tm := cttm;
    end;
    procedure SOSet_Target_Temperature(tm: in Integer) is
    begin
      cttm := tm;
    end;
    procedure SOGet_Target_Temperature2(tm: out Integer) is
    begin
      tm := cttm;
    end;
    procedure SOSet_Heat_Source_State(state: in Boolean) is
    begin
      shss := state;
    end;
    procedure SOGet_Stored_HeatSource(state: out Boolean) is
    begin
      state := shss;
    end;
  end Shared_Object1Impl;
end Shared_Object1ImplPkg;
```