The chameneons, a case study for Quasar

Evangelista Sami, Kaiser Claude Pradat-Peyre Jean-François, Rousseau Pierre

19 août 2003

This example has been presented in Chameneos, a Concurrency Game for Java, Ada and Others In Proceedings ACS/IEEE International Conference on Computer Systems and Applications 2003.

In this example, a chameneos enter in a mall. If the chameneos is the first one to enter, he waits for a second one. When they are two, they cooperate and change their color according to the color of the other chameneos. Then they go out of the mall.

This case is another illustration of the egg-shell model. When the first chameneos enter in the mall he changes `First_Call` to False, and wait until `First_Call` became True. With the protected object semantic, you will never follow the execution were all chameneos are in the mall but still waiting for a second chameneos (it happens when Cooperate entry is always executed first).

p_id_chameneos.ads :

```
package P_Id_Chameneos is
   subtype Id_Chameneos is Natural;
end P_Id_Chameneos;
```

p_colour.ads & p_colour.adb :

```
package P_Colour is
   type Colour is (Blue, Red, Yellow);
   function Complementary_Colour(C1, C2: Colour) return Colour;
end P_Colour;

-------------------------------------------------------------

package body P_Colour is
   function Complementary_Colour(C1, C2: Colour) return Colour is
   begin
      if (C1 = C2) then
         return C1;
      elsif C1 /= Red and C2 /= Red then
         return Red;
      elsif C1 /= Blue and C2 /= Blue then
         return Blue;
      else
         return Yellow;
      end if;
   end Complementary_Colour;
end P_Colour;
```

mall.ads & mall.adb :

```
with P_Id_Chameneos; use P_Id_Chameneos;
with P_Colour; use P_Colour;
package Mall is
   function Cooperation(Xa: Id_Chameneos; Ca: Colour) return Colour;
```

```
end Mall;

-------------------------------------------------------------

package body Mall is

   protected Cooperation_Synchro is
      entry Cooperate(X: in Id_Chameneos; C: in Colour; C_Other: out Colour);
   private
      entry Waiting(X: in Id_Chameneos; C: in Colour; C_Other: out Colour);
      First_Call : Boolean := True;
      A_Colour  : Colour;
      B_Colour  : Colour;
   end Cooperation_Synchro;

   protected body Cooperation_Synchro is
      entry Cooperate(X: in Id_Chameneos; C: in Colour; C_Other: out Colour)
         when True is
      begin
         if (First_Call) then
            A_Colour := C;
            First_Call := False;
            requeue Waiting;
         else
            B_Colour := C;
            C_Other := A_Colour;
            First_Call := True;
         end if;
      end Cooperate;

      entry Waiting(X: in Id_Chameneos; C: in Colour; C_Other: out Colour) when First_Call is
      begin
         C_Other := B_Colour;
      end;
   end Cooperation_Synchro;

   function Cooperation(Xa: Id_Chameneos; Ca: Colour) return Colour is
      Other_Colourb : Colour;
   begin
      Cooperation_Synchro.Cooperate(Xa, Ca, Other_Colourb);
      return Other_Colourb;
   end Cooperation;
end Mall;
```

p_id_chameneos.ads & p_id_chameneos.adb :

```
with P_Id_Chameneos; use P_Id_Chameneos;
with P_Colour; use P_Colour;
with Text_IO; use Text_IO;
package P_Chameneos is
   task type Chameneos is
      entry Start(Id2: in Id_Chameneos; C2: in Colour);
   end Chameneos;
end P_Chameneos;

-------------------------------------------------------------

with Mall; use Mall;
package body P_Chameneos is

   task body Chameneos is
```

```
      My_Id       : Id_Chameneos;
      My_Colour   : Colour;
      Other_Colourz : Colour;

      procedure Message(Mess : in String) is
      begin
         Put("(" & Id_Chameneos'Image(My_Id) & ")␣I␣am␣");
         Put_Line(Colour'Image(My_Colour) & "␣and␣" & Mess);
      end Message;

      procedure Eating_Honey_Suckle_And_Training is
      begin
         Message("I␣am␣eating␣honey␣suckle␣and␣training");
      end Eating_Honey_Suckle_And_Training;

      procedure Going_To_The_Mall is
      begin
         Message("I␣am␣going␣to␣the␣mall");
      end Going_To_The_Mall;

      procedure Mutating
         (My_Id : Id_Chameneos ;
          My_Colour : in out Colour ;
          Other_ColourT : in out Colour) is
      begin
         Message("I␣am␣ready␣to␣mute");
         Other_ColourT := Cooperation(My_Id, My_Colour);
         My_Colour := Complementary_Colour( My_Colour, Other_ColourT);
         Message("I␣have␣performed␣a␣mutation");
      end Mutating;
   begin
      accept Start(Id2: in Id_Chameneos; C2: in Colour) do
         My_Id := Id2;
         My_Colour := C2;
      end Start;
      loop
         Eating_Honey_Suckle_And_Training;
         Going_To_The_Mall;
         Mutating(My_Id, My_Colour, Other_Colourz);
      end loop;
   end Chameneos;
end P_Chameneos;
```

simulation.adb :

```
with P_Chameneos; use P_Chameneos;
with Mall; use Mall;
with P_Colour; use P_Colour;
with P_Id_Chameneos; use P_Id_Chameneos;

procedure Simulation is

   The_Colours : array(1..5) of Colour := (Yellow, Blue, Red, Blue, Yellow);
   The_Chaemenos : array(1..5) of Chameneos;

   C  : Colour;
   Id : Id_Chameneos;
begin
   for I in The_Chaemenos'Range loop
      C  := The_Colours(I);
      Id := I;
```

```
        The_Chaemenos(I).Start(Id, C);
    end loop;
end Simulation;
```