```
with AccelerometerTask_pk;
with distance_sensor;
with Wheels;
with Acc_Storage_pk;
with distance_sensor_storage_pk;
with Ada.Real_Time; use Ada.Real_Time;

package body Control_Program is

   -- This is the states the car can have:
   --     forward
   --     turn_right
   --     turned
   type move_state is (forward, turn_right, turned);

   -- The task Control_Car is a task that get in the infromation from the sensor task and control the movements to the car.
   -- By processing this data, this task set the state of what the car shall do.
   -- The task use a case statement to switch between the state to the car.
   task body Control_Car is
      Car : Wheels.Set_of_wheels;                                               -- The car variable define the car in wheels.
      current_state : move_state := forward;                                    -- Before the loop in the task start the current_state to the car
      Time_Now : Time;                                                          -- The variable Time_Now and time_next is a time variable that is
      Time_next : Time;                                                         -- how long time the car shall turn right after the dictance senso
      D : Time_Span := Milliseconds (1700);                                     -- The variable D is used to control how long time the car shall t
   begin
      loop
         -- This case statement is used to set the states that control the movements to the car.
         case current_state is
            when forward =>                                                     -- The forward case set the car to drive forward.
               Wheels.Drive_forward(Car);
               if not(Acc_Storage_pk.storage.get_upright) then                  -- If the accelerometer detect that the car has overturned the cur
                  current_state := turned;
               elsif distance_sensor_storage_pk.Sensor_flag.Get  then           -- If the distance sensor detect something in front the Time_Next
                  Time_Next := Clock + D;
                  current_state := turn_right;                                  -- Then the current_state is switched to turn_right.
               else
                  Wheels.Drive_forward(Car);                                    -- Now it has been determined that everything is OK. We repeatedly
               end if;
            when turn_right =>                                                  -- The turn_right case set the car to rotate clockwise.
               Wheels.Rotate_clockwise(car);
               if not(Acc_Storage_pk.storage.get_upright) then                  -- If the accelerometer detect that the car has overturned the cur
                  current_state := turned;
               end if;
               Time_Now := Clock;                                               -- If the car dosent overturn the car will rotate until the time_N
                  if (Time_Now > Time_Next) then                                -- When Time_Now is more than Time_Next the current_state will swi
                  current_state := forward;
               end if;
            when turned =>                                                      -- The turned case set the car on brake wich mean that the wheels
               Wheels.Brake(Car);
               if (Acc_Storage_pk.storage.get_upright) then                     -- If the accelerometer detect that the car is upright then the cu
                  current_state := forward;
               end if;
         end case;
         delay until Clock + Microseconds(500);
      end loop;

   end Control_Car;


end Control_Program;

```