COSC 3319.01 MWF

Bryan Benham

11/02/2022

Lab 1: Option A

The Quality of Life and Economy Update

I have added counters to the total number of rejected items, number of meat and total items processed into the queue, number of meat and total items sold during the day, along with prices/profit for each item available to be sold and the ability to change the queue size through user input.

CODE

```
-- in file Food_SalesService.ads

with Food_DataStructures;  --use Food_DataStructures;
with Stats_FoodDistribution;  use Stats_FoodDistribution;

package Food_SalesService is
   task type RetailSales;
   --task RetailSales;  -- May used if only a single point of sales is
required.
end Food_SalesService;
--------------------------------------------------------------------------------
-- in file Food_SalesService.adb

with Food_DataStructures;  use Food_DataStructures;
with Stats_FoodDistribution;  use Stats_FoodDistribution;
with GateKeeperService; use GateKeeperService;

with Ada.Text_IO;  use Ada.Text_IO;

package body Food_SalesService is

   package Integer_IO is new Ada.Text_IO.Integer_IO(Integer);
   use Integer_IO;

   meatSold: Integer := 0;
```

```
totalSold: Integer := 0;
money: Integer := 0;

task body RetailSales is
   food: Food_Pack;
   availableForSale: Boolean := true;

begin
  delay 1.0;  -- Allow for initialization activities.
   loop

      GateKeeper.retrieveMessage( food, availableForSale );
      delay( duration( Next_Exponential * 2.0 ) );

      case getFood_PackFoodType(food) is
         when Steak => money := money + 3500;
         when Fowel => money := money + 1500;
         when Pork => money := money + 2000;
         when Fish => money := money + 2500;
         when Wheat => money := money + 200;
         when Corn => money := money + 350;
         when Rice => money := money + 100;
         when Potatoes => money := money + 500;
         when Squash => money := money + 400;
         when Tomato => money := money + 250;
      end case;

      if getFood_PackFoodType(food) not in GrainVegetable then
         meatSold := meatSold + 1;
```

```
            totalSold := totalSold + 1;
         else
            totalSold := totalSold + 1;
         end if;

         put("Retail Sales successfuly sold "); PrintFood_Pack( food );
new_line(2);
         put("Total profit generated thus far: " ); put(money); new_line;
         put("Total number of meat sold thus far: "); put(meatSold); new_line;
         put("Total number of products (incl. meat) sold thus far: ");
put(totalSold); new_line(2);

      end loop;
   end RetailSales;
end Food_SalesService;
-------------------------------------------------------------------------------
-- in file GateKeeperService.ads

with Food_DataStructures;  use Food_DataStructures;
With Stats_FoodDistribution;  use Stats_FoodDistribution;
with CircularQue;

package GateKeeperService is
   task GateKeeper is
      entry acceptMessage( newFood: in Food_Pack );

      entry retrieveMessage( newFood: out Food_Pack; availableForShipment: out
Boolean );
   end GateKeeper;
```

```
end GateKeeperService;
-------------------------------------------------------------------------------
-- in file GateKeeperService.adb

with Ada.Text_IO;  use Ada.Text_IO;
with Ada.Calendar;  use Ada.Calendar;
with Food_SalesService; use Food_SalesService;



package body GateKeeperService is

   package IntegerIO is new Ada.Text_IO.Integer_IO(Integer);  use IntegerIO;

   task body GateKeeper is



      qSize: Natural;


      rejected: Integer := 0;
      meatCount: Integer := 0;
      totalCount: Integer := 0;



      -- Declare food packet counters here.


      Start_Time: Ada.Calendar.Time;
      End_Time:   Ada.Calendar.Time;


   begin
```

```
      put("Input a Queue Size: "); get(qSize); new_line;

      declare
         package CircularQueue is new CircularQue (Food_Pack, qSize); --
default size 10.
         use CircularQueue;
      begin


      delay 0.5;  -- allow 1/2 hour to initialize facility.
      Start_Time := Ada.Calendar.Clock;
      End_Time := Start_Time + 1.0 * 8.0 * 2.0; -- 1.0 sec./hour * 8
hours/days * 5 days


      -- Terminate after losing 5 customers or time to close has arrived.
      while rejected < 5 and Ada.Calendar.Clock < End_Time loop
         select
            -- new arrivals of food
            accept acceptMessage( newFood: in Food_Pack) do
               if not( CircularQueFull ) then
                  if getFood_PackFoodType(newFood) not in GrainVegetable then
                     CircularQueue.insertMeat( newFood );
                     meatCount := meatCount + 1;

                  else
                     CircularQueue.acceptMessage( newFood );
```

```
                end if;

                put("GateKeeper insert accepted ");
                PrintFood_Pack( newFood ); new_line;

            else
                rejected := rejected + 1;
                put("Rejected by GateKeeper: "); new_line;
                PrintFood_Pack( newFood ); new_line;
                put("Rejected = "); put(rejected);
                put(". Sent to another distribution facility!");
new_line(3);
            end if;

            totalCount := totalCount + 1;

        end acceptMessage;
      or
        -- Accept request for distribution from sales
        accept retrieveMessage( newFood: out Food_Pack;
availableForShipment: out Boolean) do
          availableForShipment := False;
          if not(CircularQueue.circularQueEmpty) then
              availableForShipment := True;
              CircularQueue.retrieveMessage( newFood );

              PrintFood_Pack( newFood ); put(" Removed by GateKeeper for
shipment."); new_line;
           end if;
```

```
            end retrieveMessage;
         end select;

         delay 1.1; -- Complete overhead due to accepting or rejecting a
request prior to new iteration.



      end loop;

      -- print time in service, statistics such as number of meat food packets
processed , non-meat products processed,
      -- and number of arriving food vessels rejected.

      new_line(2); put("Total number of rejected items: "); put(rejected);
      new_line(2); put("Total number of meat processed: "); put(meatCount);
      new_line(2); put("Total number of items processed: "); put(totalCount);

      new_line(2); put("Queue Size: "); put(qSize);
      new_line(2); put("Hours of operation prior to closing: ");
      Ada.Text_IO.Put_Line(Duration'Image(Ada.Calendar.Clock - Start_Time));
new_line(2);
      end;
   end GateKeeper;
end GateKeeperService;
-------------------------------------------------------------------------------
```

OUTPUT

Input a Queue Size:

How many Product Generators?

How many points of sale?

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 5.66092E-02 Time units!

.

.

.

Retail Sales successfuly sold TOMATO B

Total profit generated thus far: 450

Total number of meat sold thus far: 0

Total number of products (incl. meat) sold thus far: 3

B delivered.

Retail Sales successfuly sold SQUASH B

Total profit generated thus far: 850

Total number of meat sold thus far: 0

Total number of products (incl. meat) sold thus far: 4

GateKeeper insert accepted TOMATO B

Next grain shipment arrives 1.03604E+00 Time units!

FOWEL M Removed by GateKeeper for shipment.

M delivered.

GateKeeper insert accepted FISH M

Next grain shipment arrives 6.55132E+00 Time units!

FISH M Removed by GateKeeper for shipment.

Total number of rejected items: 1

Total number of meat processed: 2

Total number of items processed: 9

Queue Size: 4

Hours of operation prior to closing: 16.770419500

Retail Sales successfuly sold FOWEL M

Total profit generated thus far: 2350

Total number of meat sold thus far: 1

Total number of products (incl. meat) sold thus far: 5

B delivered.

Retail Sales successfuly sold FISH M

Total profit generated thus far: 4850

Total number of meat sold thus far: 2

Total number of products (incl. meat) sold thus far: 6