```
model Mercado

-- enumeration

enum TipoComprador {Mayorista, Minorista, Restauradores}

enum TipoProducto {Pescado, Carne, Fruta, Hortalizas}

enum NivelCalidad {Excelente, Buena, Normal, Mala, MuyMala}

-- classes

class Proveedor
attributes
  idProveedor  : Integer
  Nombre : String
  Telefono : Integer
operations
end

class Nave
attributes
  idNave:Integer
  Direccion: String
  Capacidad: Integer
  Telefono: Integer
  CalidadTotal: NivelCalidad
  Productos: TipoProducto
operations
end

class Comprador
attributes
  idComprador  : Integer
  Nombre : String
  Telefono : Integer
  TipoCompra : TipoComprador
operations
end

class Compra
attributes
  idVenta: Integer
  Cantidad: Integer
```

```
  TipoProductoCompra: TipoProducto
  PrecioTotal : Real
operations
 determinarPrecio(Cantidad : Integer, Calidad : NivelCalidad,PrecioBase : Real,OfertaDemanda
: Integer) : Real
end

class LineadeCompra
attributes
  idLinea: Integer
  Tipo: TipoProducto
  Cantidad: Integer
operations
  determinarDemanda()
end

class Producto
attributes
  idProducto: Integer
  Nombre: String
  Tipo: TipoProducto
  Stock: Integer
  PrecioBase: Real
  Calidad: NivelCalidad
operations
  getCalidad() : NivelCalidad
  getPrecioBase() : Real
  getStock() : Integer
end

abstract class Vehiculo
attributes
  idVehiculo: Integer
  Capacidad: Integer
  Cantidad: Integer
  Matricula: String
  TipoProductoTransporta: TipoProducto
operations
end

class Camion < Vehiculo
attributes
  Medidas: Integer
  NumRuedas: Integer
```

```
operations
end

class Furgoneta < Vehiculo
attributes
  NumPuertas : Integer
operations
end

class Turismo < Vehiculo
attributes
  NumPuertas : Integer
  TaraMaxima : Integer
operations
end

-- associations

association Transporta between
  Vehiculo[1..*] role vehiculo
  Proveedor[1] role proveedor
end

association Utiliza between
  Nave[1] role tienda
  Proveedor[0..*] role proveedor
end

association Existe between
  Nave[1] role tienda
  Producto[0..*] role producto
end

association Contiene between
  Producto[1..*] role producto
  LineadeCompra[0..*] role cantidad
end

association Realiza between
  Comprador[1] role cliente
  Compra[1..*] role compra
end

composition CompraLinea between
```

```
  Compra[1] role conjunto
  LineadeCompra[1..*] role cantidad
end

--constraints

constraints

context Vehiculo
        inv VehiculoCantidad: Capacidad >= Cantidad
        inv idDistintoV: Vehiculo.allInstances->forAll(v1, v2 | v1.idVehiculo <> v2.idVehiculo
implies v1.Matricula <> v2.Matricula)

context Furgoneta
        inv NumeroPuertasFurgo: NumPuertas >= 1

context Turismo
        inv MismaCapacidad: TaraMaxima < Capacidad
        inv NumeroPuertasTurismo: NumPuertas >= 1

context Camion
        inv NumeroRuedasCamion: NumRuedas >= 4

context Nave
        inv ProductosN:self.producto->forAll(P | self.Productos = P.Tipo)
        inv idDistintoN: Nave.allInstances->forAll(n1, n2 | n1.idNave <> n2.idNave implies
n1.Direccion <> n2.Direccion)

context Proveedor
        inv idDistintoP: Proveedor.allInstances->forAll(p1, p2 | p1.idProveedor <>
p2.idProveedor implies p1.Telefono <> p2.Telefono)

context LineadeCompra
        inv CompraReal: self.producto->forAll(P | self.Cantidad < P.Stock)
        inv ProductosL: self.producto->forAll(P | self.Tipo = P.Tipo)




context Comprador
        inv idDistintoC: Comprador.allInstances->forAll(c1, c2 | c1.idComprador <>
c2.idComprador implies c1.Telefono <> c2.Telefono)
        inv TipoMayo: self.compra->forAll(C | C.Cantidad >= 50 implies self.TipoCompra =
```

TipoComprador::Mayorista)

– a partir de aqui