...\ProgramaciónIes2023\LaPaulita - copia\LaPaulita.Sales\ValueObject\OrderDetail.cs 1

```csharp
namespace LaPaulita.Sales.BusinessRules.ValueObject
{
    public class OrderDetail : IEquatable<OrderDetail>
    {
        #region "Campos"
        readonly int _productId;
        readonly decimal _productPrice;
        readonly short _productQuantity;
        #endregion

        #region "Propiedades"
        //public int ProductId { get { return _productId; } }
        public int ProductId => _productId; //expresión Lambda
        public decimal ProductPrice => _productPrice;
        public short ProductQuantity => _productQuantity;
        #endregion

        #region "Cosntructor"
        public OrderDetail(int productId, decimal productPrice, short productQuantity)
        {
            _productId = productId;
            _productPrice = productPrice;
            _productQuantity = productQuantity;
        }
        #endregion

        #region"Equals and GetHashCode"
        public override bool Equals(object? obj)
        {
            return Equals(obj as OrderDetail);
        }

        public bool Equals(OrderDetail? other)
        {
            //return other is not null &&
            //       ProductId == other.ProductId &&
            //       ProductPrice == other.ProductPrice &&
            //       ProductQuantity == other.ProductQuantity;

            //*****************************************************
            // Otra forma usando GetHashCode.
            //*****************************************************
            if (other != null) { return GetHashCode() == other.GetHashCode(); }
            else { return false; }
        }

        public override int GetHashCode()
        {
            return HashCode.Combine(ProductId, ProductPrice, ProductQuantity);
        }
        #endregion
    }
}

```

...rogramaciónIes2023\LaPaulita - copia\LaPaulita.Sales\ValueObject\OrdersDetails.cs 1

```csharp
namespace LaPaulita.Sales.BusinessRules.ValueObject
{
    public record class OrdersDetails(int ProductId, decimal ProductPrice, short ProductQuantity);
}

```

```
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.ValueObject;

namespace LaPaulita.Sales.BusinessRules.Agregates
{
    public class CreateOrder : OrderHeader
    {
        //**********************************************************************
        // Esto si lo hago con una clase que es preparada por nosotros para ser
        // inmutable.
        //**********************************************************************


        // Campo
        readonly List<OrderDetail> records = new List<OrderDetail>();
        // Propiedad
        public IReadOnlyList<OrderDetail> Details => records;

        public void AddDetail(OrderDetail record)
        {
            var ExistingOrderDetail = records.FirstOrDefault(r => r.ProductId == record.ProductId);
            if (ExistingOrderDetail != default)
            {
                records.Add(new OrderDetail(record.ProductId, record.ProductPrice,
                    (short)(record.ProductQuantity + ExistingOrderDetail.ProductQuantity)));
                records.Remove(ExistingOrderDetail);
            }
            else
            {
                records.Add(record);
            }
        }

        public void AddRecord(int productId, decimal productPrice, short productQuantity)
        {
            AddDetail(new OrderDetail(productId, productPrice, productQuantity));
        }

        public static CreateOrder From(OrderHeaderDto orderHeaderDto)
        {
            // Aqui realizamos el mapeo de las propiedades del DTO con la Entidad.
            CreateOrder createOrder = new CreateOrder
            {
                ClientId = orderHeaderDto.ClientId,
                ShippingAddress = orderHeaderDto.ShippingAddress,
                ShippingCity = orderHeaderDto.ShippingCity,
                ShippingCountry = orderHeaderDto.ShippingCountry,
                ShippingZip = orderHeaderDto.ShippingZip
            };

            foreach (var item in orderHeaderDto.OrderDetails)
            {
                createOrder.AddRecord(item.ProductId, item.ProductPrice, item.ProductQuantty);
            }
            return createOrder;
        }
    }
}

```

```csharp
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.ValueObject;

namespace LaPaulita.Sales.BusinessRules.Agregates
{
    public class CreatesOrders : OrderHeader
    {
        //**********************************************************************
        // Esto si lo hago con una clase tipo record inmutable.
        //**********************************************************************

        readonly List<OrdersDetails> records = new List<OrdersDetails>();
        public IReadOnlyList<OrdersDetails> Details => records;

        /// <summary>
        /// Método que agrega un nuevo record a <c>OrdersDetails</c> de una orden de compra
        /// <c>OrderHeader</c>.
        /// Si el nuevo record a agregar tiene un <c>ProductId</c> que ya existe en
        /// <c>OrdersDetails</c>, solo se modifica la cantidad del producto adicinando al
        /// valor existente la nueva cantidad.
        /// </summary>
        /// <param name="record">Objeto que contiene los datos del nuevo registro a agregar.</param>
        public void AddRecord(OrdersDetails record)
        {
            var ExistingOrderDetail = records.FirstOrDefault(r => r.ProductId == record.ProductId);
            if (ExistingOrderDetail != default)
            {
                // Con with creamos una copia del objetoque esta a su izquierda pero con
                // las propiedades con valores modificado (entre la llaves)
                // Es decir agrega a la lista records una copia del objeto ExistingOrderDetail
                // pero modificando el valor de la propiedad ProductQuantity, asignandole como valor
                // lo que tiene ExistingOrderDetail.ProductQuantity sumado a lo que tiene
                // record.ProductQuantity

                records.Add(ExistingOrderDetail with
                {
                    ProductQuantity = (short)(ExistingOrderDetail.ProductQuantity + record.ProductQuantity)
                });
                records.Remove(ExistingOrderDetail);
            }
            else
            {
                records.Add(record);
            }
        }

        public void AddRecord(int _productId, decimal productPrice, short productQuantity)
        {
            AddRecord(new OrdersDetails(_productId, productPrice, productQuantity));
        }

        public static CreateOrder From(OrderHeaderDto orderHeaderDto)
        {
            // Aqui realizamos el mapeo de las propiedades del DTO con la Entidad.
            CreateOrder createOrder = new CreateOrder
            {
                ClientId = orderHeaderDto.ClientId,
                ShippingAddress = orderHeaderDto.ShippingAddress,
                ShippingCity = orderHeaderDto.ShippingCity,
                ShippingCountry = orderHeaderDto.ShippingCountry,
                ShippingZip = orderHeaderDto.ShippingZip
            };

            foreach (var item in orderHeaderDto.OrderDetails)
            {
                createOrder.AddRecord(item.ProductId, item.ProductPrice, item.ProductQuantty);
            }
            return createOrder;
        }
    }
}

```

...urce\ProgramaciónIes2023\LaPaulita - copia\LaPaulita.Sales\DTOs\OrderHeaderDto.cs 1

```
namespace LaPaulita.Sales.BusinessRules.DTOs
{
    public class OrderHeaderDto
    {
        public int ClientId { get; set; }
        public string ShippingAddress { get; set; }
        public int ShippingCity { get; set; }
        public int ShippingCountry { get; set; }
        public string ShippingZip { get; set; }

        public List<OrderDetailDto> OrderDetails { get; set; }
    }
}
```

...urce\ProgramaciónIes2023\LaPaulita - copia\LaPaulita.Sales\DTOs\OrderDetailDto.cs 1

```
namespace LaPaulita.Sales.BusinessRules.DTOs
{
    public class OrderDetailDto
    {
        public int ProductId { get; set; }
        public decimal ProductPrice { get; set; }
        public short ProductQuantty { get; set; }
    }
}
```

...Ies2023\LaPaulita - copia\LaPaulita.Sales\DTOs\ValidatorDTO\ValidationErrorDto.cs 1

```
namespace LaPaulita.Sales.BusinessRules.DTOs.ValidatorDTO
{
    public class ValidationErrorDto
    {
        public string PropertyName { get; set; }
        public string ErrorMessage { get; set; }
    }
}

```

...gramaciónIes2023\LaPaulita - copia\LaPaulita.Sales\Wrappers\WrappersSalesOrder.cs 1

```
namespace LaPaulita.Sales.BusinessRules.Wrappers
{
    public class WrappersSalesOrder
    {
        public int OrderId { get; set; }
        public List<ValidationErrorDto> Errors { get; set; }
    }
}

```

...LaPaulita - copia\LaPaulita.Sales\Interface\Controllers\ICreateOrderController.cs 1

```
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.BusinessRules.Interface.Controllers
{
    public interface ICreateOrderController
    {
        Task<WrappersSalesOrder> CreateOrder(OrderHeaderDto order);
    }
}

```

...2023\LaPaulita - copia\LaPaulita.Sales\Interface\Getways\ICreateOrderInputPort.cs 1

```
using LaPaulita.Sales.BusinessRules.DTOs;

namespace LaPaulita.Sales.BusinessRules.Interface.Getways
{
    public interface ICreateOrderInputPort
    {
        Task Handle(OrderHeaderDto createOrderDto);
    }
}

```

...023\LaPaulita - copia\LaPaulita.Sales\Interface\Getways\ICreateOrderOutputPort.cs 1

```
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.BusinessRules.Interface.Getways
{
    public interface ICreateOrderOutputPort
    {
        //ValueTask Handle(int orderId);
        ValueTask Handle(WrappersSalesOrder order);
        ValueTask ValidationFaild(List<ValidationErrorDto> validationError);
    }
}

```

...3\LaPaulita - copia\LaPaulita.Sales\Interface\Presenters\ICreateOrderPresenter.cs 1

```
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.BusinessRules.Interface.Presenters
{
    public interface ICreateOrderPresenter : ICreateOrderOutputPort
    {
        //int OrderId { get; }
        //List<ValidationErrorDto> ErrorsList { get; }
        WrappersSalesOrder Order { get; }
    }
}

```

...Paulita - copia\LaPaulita.Sales\Interface\Repositories\ISalesCommandRepository.cs 1

```
using LaPaulita.Entity.Interfaces;
using LaPaulita.Sales.BusinessRules.Agregates;

namespace LaPaulita.Sales.BusinessRules.Interface.Repositories
{
    public interface ISalesCommandRepository : IUnitOfWork
    {
        Task CreateOrder(CreateOrder order);
    }
}

```

...ProgramaciónIes2023\LaPaulita - copia\LaPaulita.Sales\Interface\ISpecification.cs 1

```
using LaPaulita.Sales.BusinessRules.DTOs.ValidatorDTO;

namespace LaPaulita.Sales.BusinessRules.Interface
{
    public interface ISpecification<T> where T : class
    {
        List<ValidationErrorDto> IsValid();
    }
}

```