**-----ENUM-----**

MANEIRA DE REPRESENTAR UM CONJUNTO DE CONSTANTES/PALAVRAS, SENDO CADA CONSTANTE/PALAVRA UM VALOR NUMÉRICO.

EX: UMA CLASSE PEDIDO, CUJO ATRIBUTO Status É UM ENUM DA SEGUINTE FORMA

class Order

{

public int Id { get; set; }

public DateTime Moment { get; set; }

public OrderStatus Status { get; set; }

}

ATRIBUTO Status É DO TIPO enum OrderStatus, SENDO ESSE enum

enum OrderStatus : int

{

PendingPayment = 0,

Processing = 1,

Shipped = 2,

Delivered = 3

}

ASSIM, OS ESTADOS DO PEDIDO PODEM SER RELACIONADOS COM ESSES VALORES NO MOMENTO DE INSTANCIAR UM OBJETO Order

NO PROGRAMA PRINCIPAL

Order order1 = new Order

{

Id = 01,

Moment = DateTime.Now,

Status = OrderStatus.PendingPayment //A PARTIR DO . O COMPILADOR SUGERE

};

PARA ORGANIZAR ESSA ESTRUTURA NUM PROJETO, O IDEAL É CRIAR UMA PASTA ENTIDADES, ONDE FICARÃO TODAS AS CLASSES QUE REPRESENTAM ENTIDADES DA APLICAÇÃO, E DENTRO DESSA PASTA ENTIDADES, CRIAR UMA SUBPASTA ENUMS, ONDE FICARÃO OS ENUMS RELACIONADOS.

ADICIONAR->CLASSE->NOMEAR->TROCAR class POR enum

LEMBRAR DE: PARA QUE UMA ENUM/CLASSE DE UM NAMESPACE DIFERENTE SEJA VISÍVEL, É NECESSÁRIO IMPORTÁ-LA NO CÓDIGO

using ProjetoOO\_3.Entities;

using ProjetoOO\_3.Entities.Enums;

**---CONVERTENDO ENUM PARA STRING---**

string txt = OrderStatus.PendingPayment.ToString();

**---CONVERTENDO STRING PARA ENUM---**

OrderStatus order\_status = Enum.Parse<OrderStatus>("Delivered");

//CONSIDERANDO QUE NA ESTRUTURA DO ENUM OrderStatus JÁ TENHA O VALOR Delivered

//ASSIM A STRING PASSADA NO PARÂMETRO SE TORNA O VALOR DO OBJETO order\_status

//NÃO FUNCIONARIA SE A STRING NÃO EXISTISSE DENTRO DA ESTRUTURA DO ENUM (Delivere POR EXEMPLO)

**REPRESENTAÇÃO UML ENUM**



**---COMPOSIÇÃO---**

UM OBJETO CONTÉM OUTRO - TEM UM / TEM VÁRIOS

DIAMANTE PRETO - TODO PARTE

**ENTRE ENTIDADES**



UM PEDIDO CONTÉM VÁRIOS ITENS

UM PEDIDO ESTÁ RELACIONADO A UM CLIENTE (COMPOSIÇÃO TAMBÉM)

UM ITEM TEM 1 PRODUTO

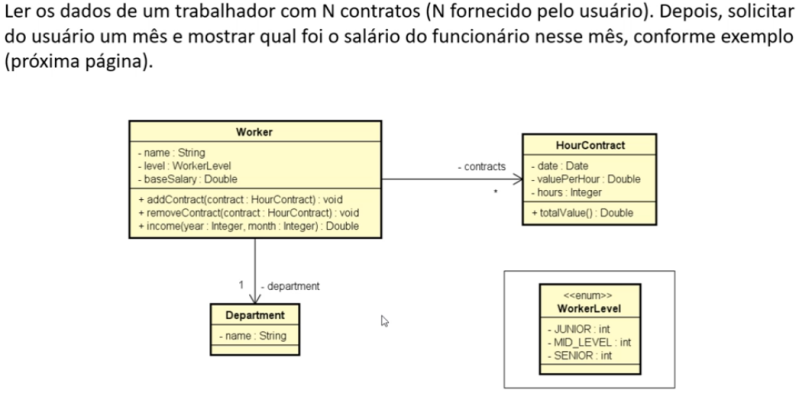
**ENTRE SERVIÇOS**



TANTO UM SERVIÇO DE PEDIDO QUANTO UM SERVIÇO DE AUTENTICAÇÃO ESTÃO RELACIONADOS COM SERVIÇO DE E-MAIL

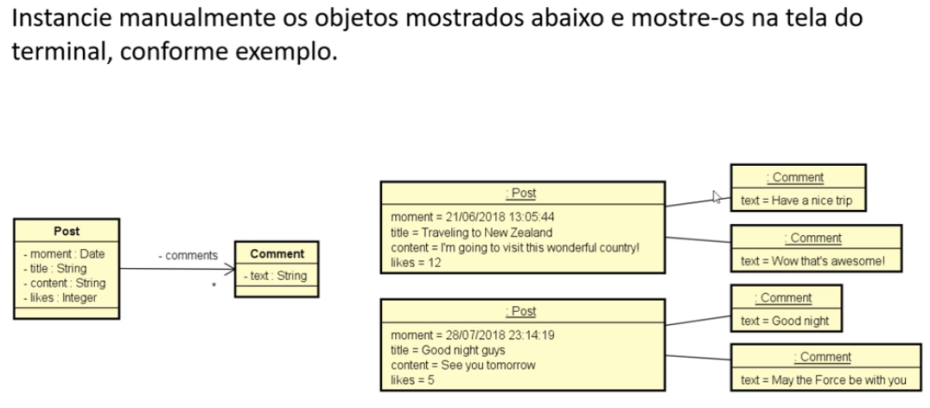
UM SERVIÇO DE PEDIDO ESTÁ RELACIONADO COM UM REPOSITÓRIO DO PEDIDO (QUE ACESSA O BD)

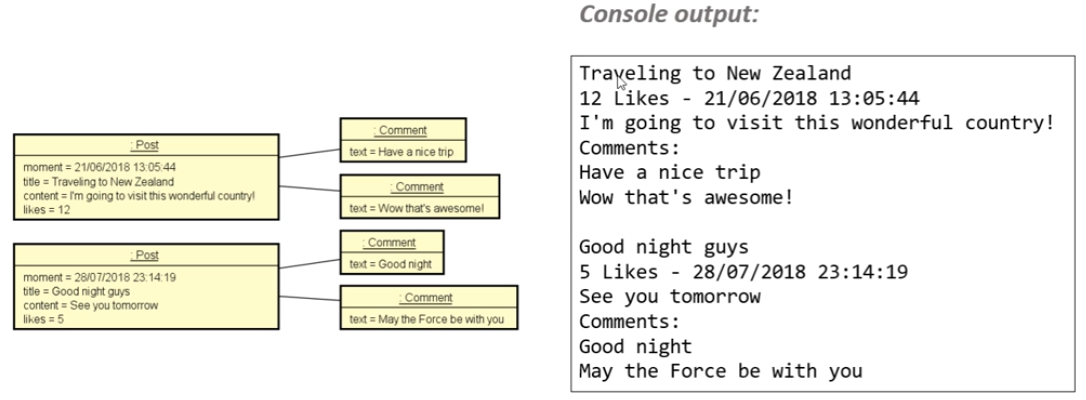
**EXERCÍCIO RESOLVIDO 1**

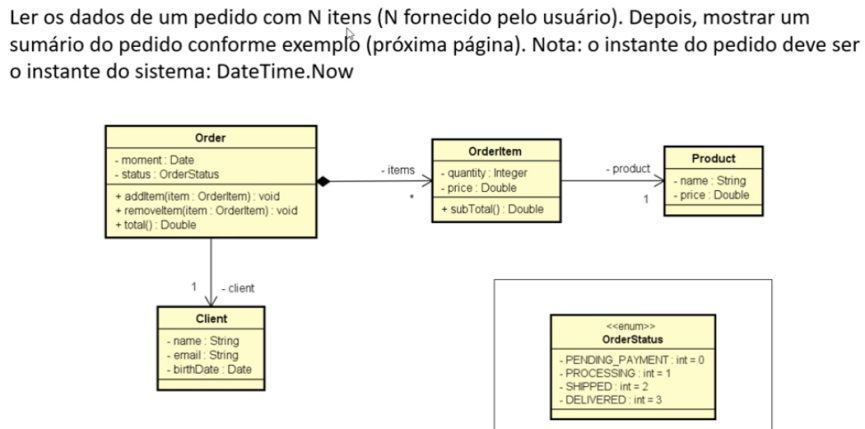
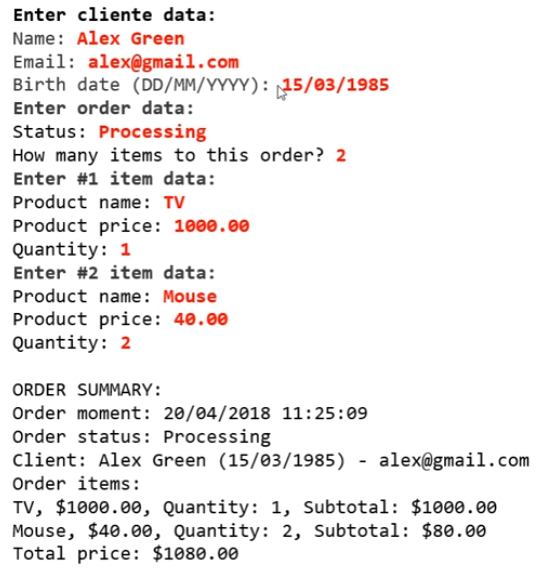
\*NO CONSTRUTOR DE WORKER, NÃO USAR COMO ARGUMENTO A LISTA. SEMPRE NUMA RELAÇÃO DE COMPOSIÇÃO DE UM PARA MUITOS, O CONSTRUTOR DE “UM” NÃO INICIA JÁ COM A LISTA DE “MUITOS”, O MUITOS É ADICIONADO AOS POUCOS.

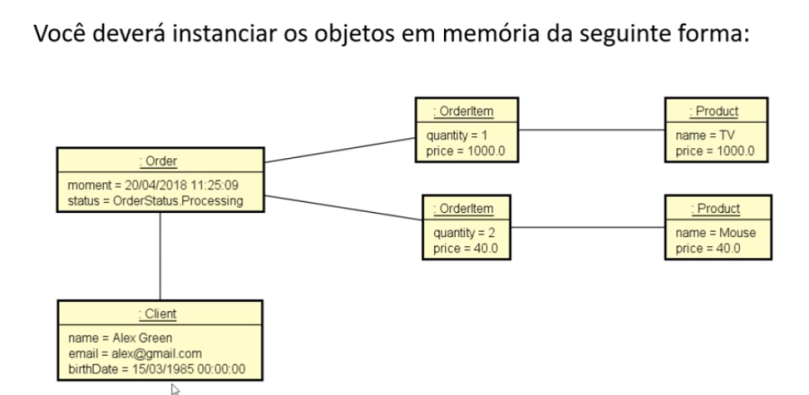
**EXERCÍCIO RESOLVIDO 2**





**EXERCÍCIO RESOLVIDO 3**



**StringBuilder**

UMA CLASSE QUE PODE SER UTILIZADA PARA OTIMIZAR O MÉTODO ToString DE UMA CLASSE. QUANDO O MÉTODO ToString TEM UM CONTEÚDO MUITO GRANDE, É RECOMENDÁVEL UTILIZAR O StringBuilder PARA CRIAR A SAÍDA.

public override string ToString()

{

StringBuilder sb = new StringBuilder();//MONTAR STRING COM MAIS EFICIÊNCIA

sb.AppendLine(Title);

sb.AppendLine($"{Likes} Likes - {Moment.ToString("dd/MM/yyyy HH:mm:ss")}");

sb.AppendLine(Content);

sb.AppendLine("Comments");

foreach(Comment c in Comments)

{

sb.AppendLine(c.Text);

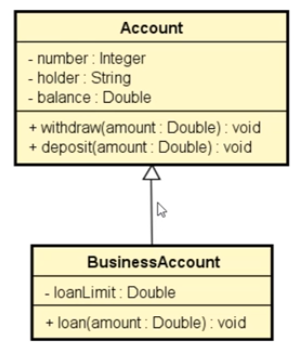
}

return sb.ToString();

}

HERANÇA

ASSOCIAÇÃO ENTRE CLASSES QUE PERMITE REUTILIZAR ATRIBUTOS E MÉTODOS DA CLASSE BASE PARA A CLASSE DERIVADA, ADICIONANDO SOMENTE O QUE FOR NECESSÁRIO, AO INVÉS DE REPETIR TUDO.



PARA GARANTIR QUE UM ATRIBUTO/MÉTODO SÓ É ACESSÍVEL NA CLASSE OU EM UMA DE SUAS SUBCLASSES, USAR protected

---USANDO CONTRUTOR DA BASE NA DERIVADA---

CONSTRUTOR DA BASE

public Account(int number, string holder, double balance)

{

Number = number;

Holder = holder;

Balance = balance;

}

CONSTRUTOR DA DERIVADA - USA AS INFORMAÇÕES E ADICIONA UM PARÂMETRO

public BusinnesAccount(int number, string holder, double balance, double loanLimit)

: base(number, holder, balance)

{

LoanLimit = loanLimit;

}

---UPCASTING E DOWNCASTING---

SÃO AÇÕES DE:

UPCASTING - INSTANCIAR UM OBJETO DA CLASSE BASE RECEBENDO UM CONSTRUTOR DA CLASSE DERIVADA

DOWNCASTING-INSTANCIAR UM OBJETO DA CLASSE DERIVADA RECEBENDO UM CONSTRUTOR DA CLASSE BASE.

AMBOS SÃO ACEITOS

CONSIDERANDO AS CLASSES Account(BASE) , BusinessAccount(DERIVADA) E SavingAccount(DERIVADA)

Account a = new Account(1, "Luciano", 10500);

BusinnesAccount b = new BusinnesAccount(2, "Comercial Luciano", 35000, 12000);

Account c = new SavingsAccount(3, "Lara", 12000, 0.05);//UPCASTING

//CASTING USANDO as

//Account d = (BusinnesAccount)b;

//Account d = b as BusinnesAccount;

SavingsAccount e = (SavingsAccount)c; //DOWN CASTING

//c É SavingAccount - COMPATÍVEL COM e

//BusinnesAccount e = (BusinnesAccount)a;//DOWNCASTING DEFEITUOSO - NÃO COMPATÍVEL

//BusinnesAccount f = (BusinnesAccount)c;//DOWNCASTING DEFEITUOSO - NÃO COMPATÍVEL

//PARA TESTAR ANTES DE FAZER O CASTING

if (c is BusinnesAccount)

{

BusinnesAccount f = (BusinnesAccount)c;//DOWNCASTING DEFEITUOSO - NÃO COMPATÍVEL

}

else

{

Console.WriteLine("NÃO COMPATÍVEL");

}

DOWNCASTING - OPERAÇÃO INSERGURA - DEVE SER USADA SOMENTE QUANDO NECESSÁRIO