**accounting:**

1-in entity \*->card relationship entityType Must be person/organization

2-bank defines fee for each card

3-account balance needs to be updated by database trigger each time new record inserted

4-invoice payee/payer entity need to be of entityType{person/org}

5-in entity/Card relation entity must be of entityType {person/org}

**Invoice and Services >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

1-all the services attached to invboice need to be in the same currency as invoice

2-all services added to the invoice need to the follow invoice\_payeeID =service\_giverID

***create Services***

entities including in service need to be of entityType{}

***create invoice***

invoiceAction

invoiceActionTransaction

***create service[s] and assign to invoice***

***payInvoice***

invoicePayment

invoiceAction

invoiceActionTransaction

**CARDS and BANKING >>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

***create bank***

***assign card to the bank***

***create fee and assign to card***

**Test Scenario >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

**(1)**

**1-reset database and seeds , 2-setup basic entities**

2-create initial lookups

3-create currency

3-create 2 persons

4-attach accounts to persons

5-Create 2 services

6-Creeat Invoice

7-assign Services to invoice

8-Finalize Invoice

9-Create Cards and assign to users

10-assign cards to persons

11-to partial payments for invoices

**(2)**

cancel payments wéo applying fees

**lookup types:**

glType

catType

cardtype

ccCardType

paymentType

extPaymentType

invoiceStat

currencyType

entityType

contectType

officeType

userType

sysUserType

**OOP [programming structure]**

1-class implementation for each entity needs its

(1)inheritance Distinctor,

(2)properties [like DB table definition] and reference [ID or object reference] to other related objects

Note1: references can be fetched using Methods instead of saving for a long time

Note2:these references can be in form of list<EF objects> or data table using SP

(3)constructor(optional) for each inheritance level.

2-lower db connections can be EF or SP and each time need to reload the object

3-non-related sets of data can be fetched from static Methods

4-obviously there is a need for mapping between server-side class definition and data fetched from database using SP or EF

5-any Method call may change stat of object including new values for properties

concept : server side encapsulation is a shell

6-Method-return-type can be any kind of data structure EF objects list or dataset

7-define class props like database in hierarchical method and remove redundant props

8-use virtual/override technique to populate inherited tables

**Design**

1-analysis and business rules

2-class and object diagram

3-database design and cascading flows

4-loop

development module by module and documentation in code

write test cases and update document

write SQL select result queries

build required views

end-loop