2021-11-09 -- Scratchpad of CSE213 (Sec-1)
==========

Milestone-2: System Design:

- It includes many things, but we will restrict ourselves in producing following deliverables:
    - UML class diagram
    - File structure (representing database)
- To produce the above, we need a compresensive CRA-report in place (Milestone-1)

Steps to built class diagram:

- Understand the growth volume of the class-instances and normalize the high-volume growth classes
    - (We will try to understand this by our own terms "Master" and "Transaction" classes)
    - Avoid redundant fields to restrict volume

- Analyzing CRA report:
  - Identify Model classes representing application data:

    - Identify USER classes
    - Identify NON-USER classes

    - Finalize decision about the user defined type
        - Decide whether the type will be a:
            abstract-class / non-abstract class / interface

    - Finalize decision about the fields
        - Decide whether the field is static or non-static
        - Decide whether the field is final or not

    - Finalize decision about the methods
        - Decide whether the method is static or non-static
        - Decide whether the method is abstract or not
        - Decide whether the method is final or not
        - ## Does this method returns data back to the UI-scene or not?
            : this will guide to decide the return type of the method

  - Identify Controller classes representing UI-scene:

      - We will talk more on this when we start building FXML application (next class)

==========

Master vs Transaction class:
==========

Master class:

- Class which has a unique field (value of the field is unique for each instances)
  to identify an instance.
- Also the growth of instance volume (database size) is under control / insignificant

Transaction class:

- Class which has NO unique field (value of the field is unique for each instances)
  to identify an instance.
- Also the growth of instance volume (database size) is quite high

Example: Dutch Bangla Bank Limited:

#: No of daily new Account (class) os opened for the bank:
- No of branch: 214
- No of Fast Track booth: 1268
- No of ATM: 4930
- No of Agent banking: 63
- Assume, no of new accounts opened in a branch: 100
- Assume, no of new accounts opened in DBBL Fast Track booth: 50
- Assume, no of new accounts opened through DBBL Agent banking: 30

- No of new accounts: 214*100 + 1268*50 + 63*30 = 67,430
- In DBBL data center, 67,430 new records (Account class instances) are added to the database

#: No of daily bank-transactions occured in DBBL:
- No of cash deposit in a branch: 500
* 214 = 107000
- No of check deposit in a branch: 700
* 214 = 149800
- Deposit cash using CDM in fast track: 100
* 1268 = 126800
- Deposit check using CDM in fast track: 200
* 1268 = 263600
- Online fund transfer via internet banking: 50000
= 50000
- Online fund transfer via rocket (mobile banking): 10000
= 10000

- Withdrwal cash from each ATM: 300
* 4930 = 1479000
- Encashing check from each branch: 700
* 214 = 149800
- Wihdrwal as a result of online fund transfer:

= 50000

	Total bank-transactions of DBBL per day:
= 2341000

- Growth of volume: 67,430 vs 2341,000

public class Account{
	fields: accountNo, accountName, typeOfAccount, balance,
	address, email, contactNo, NID, ...
}
	- has unique field
	- volume growth under control
	- It is a Master class

public class BankTransaction{
	fields: locationId, amount, typeOfTransaction, date,
processedBy,
	accountNo, accountName, contactNo
}
	- No unique field
	- volume growth is quite high
	- It is a Transaction class

- Since volume growth is quite high, we need to normalize the class by eliminating
  redundant fields
- Also we can add an unique identifyer (new unique field) to the class as follows

public class BankTransaction{
	fields: transactionId,
	accountNo, locationId, amount, typeOfTransaction, date,
processedBy
}

==========

- After detection of classes, we need to establish relationships among those classes:
	- inheritance
	- aggregation
	- composition
	- association and multiplicity (1:1, 1:M, M:1, M:M, *:1, *:M, M:*)

- Then draw the UML class diagram for the above using one of the online tools. We will
  use lucidchart

To cover in next class (14-Nov-21):

- Writing workflow for a sample goal (IRAS)
- More in Interface
- Understanding MVC framework
- Introduction to FXML application development