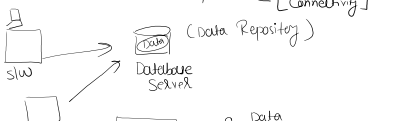
SDLC

① Requirement gathering
- Fund transfer
- Bal Enq
- Account Statement
- Bill

② Requirement Analysis

[Requirements]
- Functional Req
  - B. Logic Comp
    - Fund transfer
    - Bal Enq
- Non Functional Req
  - [Service code]
  - [validation]
  - [Security services]
  - [Connectivity]

[Sum = a+b ;] ?



B Logic ← Data, client
Eg. AccNo

Persistance of Data
- Store Data of an applicaⁿ
  - RAM × volatile
  - HW Disk
- External Data Stores / Repositories
  - RDB
  - flat file system

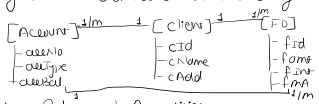
Persistance

③ [SLW Design] Design
- Structure
- User Interface (GUI)
- Database design

① Identify all the prime entities of the SLW system
- Data, Functionalities

[Account] [Client] [FD] [RD]

② Identify all the attributes of the entity

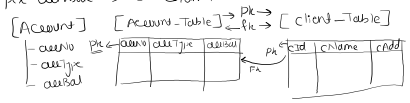


③ Establish Relation bet^n entities
- ERD

[Database-Design]

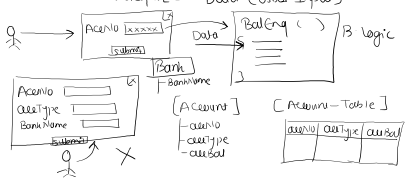
Per Entity → 1 Table
per attribute → 1 column

[Client]
- cId
- cName
- cAdd



③ GUI (Graphical User Interface) / CUI

[GUI]
- End Users — Simplicity of usage, self Interactive
- Developer — Data (Users Input)



④ [Coding]

[OOPS]

Design — Entity
Code — Object

[Object] — predefined struct + behv
- Tangible object — Eg. [Car]
- Intangible object — Eg. [Insurance car]

Design — Entity × attributes
Code — Object — variables

[Characteristics of an object]

① Identity — unique property
- set of properties

Eg. [Account] (accNO CPK) [Student] (rollNO CPK)

Eg. no plate [MH-12 BY 4220] — Composit key — ③

② State :- The current value of attribute of an obj.

Eg. Account (accNO, accBal = 100,000/-) ✓
Car (carNO, Speed = 50)

State — Inactive / active
State :- motionless / :- motion

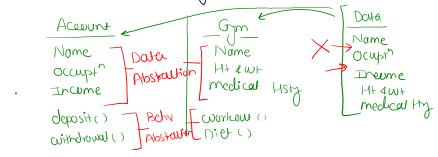
③ Behaviour :- Action - reaction

④ Responsibility :- The role object is expected to serve within system

Eg. Student - object
Studying - role - responsibility

[Pillars of OOPS]

① [Abstraction] — focusing on required & relevant details only.



Eg. [Account]
- accNO
- accType
- accBal
- BankName

Data abstraction

fundtransfn
deposit
balEnq
— Behv Abstract

② Encapsulation :- Objects Capsule

Capsules — Simple Security — Hiding the data & impl

I Account
- accNO
- accBal
- accType

II Access Specifiers
- private
- public
- protected
- default

III Class
Data + Behv

— Abstraction — what to hide
— Encapsulation — How to hide *

Object Collaboration (Communication)
- Inheritance
- Containment

×

③ Inheritance — why → Extensibility
what → parents features + childs own features

Eg. [Nokia 2600] – Parent
- InFlow
- Sms

[Nokia 6300] – child
- InFlow
- Sms
- Camera / Fm

Extensibility

P m.D
C m.D + Cardiologist

[Account] – Parent
- accNo
- accType
- accBal

[SavingAccount] – child
- accNo
- accType
- accBal
- intRate

[DematAccount]
reuse
trading

[Account] → Parent
- accNo
- accType
- accBal

General properties → Generalisation

[— is a —]

[SavingAccount] → child
- intRate

Special property → Specialisation

Simple
P
↑
C

Single chain
P
C C

Multilevel
G
↑
F
↑
S

[Java]
Multiple
C
P P
C
- cyclic dimond
- Hybrid

[Containment Relationship]

Container obj
[Content obj]
created()

Eg. Laptop – is Contained
klb – is content
Battery – is Content obj

Eg. Car – Contained
Engine – Content

1 Container obj
Can Hold 1 or more
Content obj to
reuse them

Eg. Brain, Heart – Content obj.
Human body
Container

Container
B.logic comp.
Security
DB Connecting
Reuse
Services
Convert
obj

④ Polymorphism:- Poly – many & morph – form

Static Polymorphism
1 Object – many roles
[within a same class]
– method overloading –

Dynamic Polymorphism
Parent – child
(Inheritance)
– method Overriding

Simplicity & maintainability

P
Com()
m.D
C
Com()
m.D
+
Cardiology

↳ Extensibility

Post codding in SDLC
⑤ Testing – [Junit framework / unit Testing]
⑥ Deployment – Local Server → Hosting Server
⑦ Maintenance