* Accounting Year concept
→

SE

23/02/24
Friday

* Software development Process and Process Models
↳

* Fourth Generation Techniques [4GT]
↳ ↱ Developer oriented
↳ Eg: Ruby on Rails, Python
[4 generation of languages

Requirement Gathering → Design strategy → Implementation using 4GL

Merits:
① Reduction in software development time.
② Increase productivity of developer.

Demerits:
① Not very easy.

* Component Based Development
↳ Based on the spiral model.
↳ It says if you have two projects which are very similar and one is done and second you are starting with, then you don't need to

Types of process models.

↳ Prescriptive
① Waterfall
② Incremental
③ RAD

Evolutionary
① Spiral
② 4GT
③ Component
④ Concurrent dev

Agile

build the second from scratch, you can reuse components from first

Merits → Faster development.

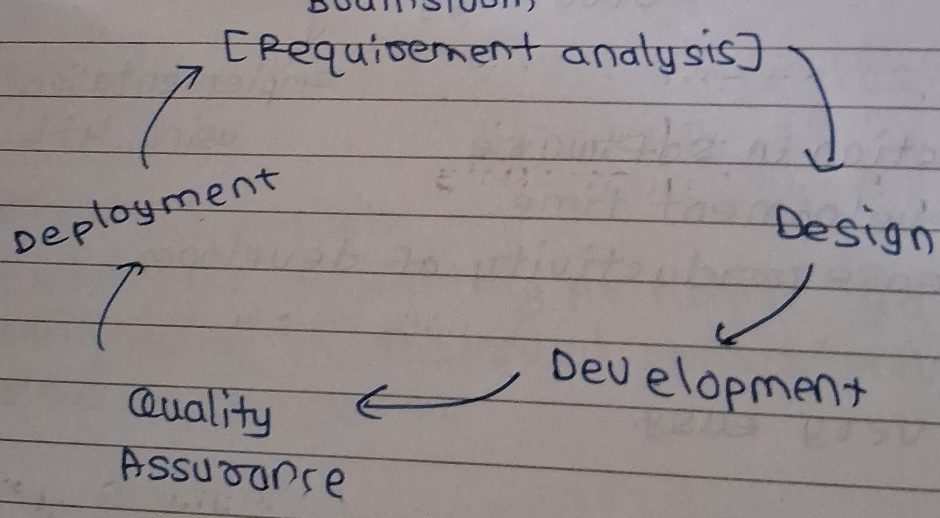
Demerits → integration of components from different sources is hard.

*Agile Process Model → customer is part of your team.

→ ↳ So here the product development involves continous interaction between developer and customer.

↳ Biggest advantage.

CPMCD




*Agile Testing Methods

→ ① Scrum
② Crystal
③ eXtreme Programming [XP]. → used when customer requirements are constantly changing
④ Lean Software Development.

⑤ Feature Driven Development [FDD].
⑥ Dynamic Software Development Method [DSDM]

* Agile model → used for unstructured projects
↳ A]. when it is used?
↳ ① Frequent changes are expected in project.
② Highly experienced team.
③ customer availablity is guranteed.
④ Project size is small.

B] Advantages of Agile Method
↳ ① Frequent Delivery.

C]. Disadvantages of Agile

Unit-2 starts → REQUIREMENT ENGINEERING

As a whole it means:

↓
A constraint / property / function / feature that the system must fulfill.

↳ Requirements for a product are defined, managed and tested systematically.

* characterstic of a good Requirement.
→ ① Simple, clear, unambigous.
② Correct
③ Easily understandable.
④ Should be able to test.
⑤ complete
⑥ consistent
⑦ Traceable

*why is getting Good Requirements Hard?

* Requirements Engineering Tasks.

and they can be
its example linked list fashion.

* FAT File System
  ↳ File Allocation table
    ↳ its variants: FAT 8, FAT 12, FA
      FAT 32, VFAT.

* Indexed allocation.

28/02/2024
Wednesday

SE

* Requirement Engineering Tasks
→
① Inception → getting basic understanding of problem.
② Elicitation → Draw requirements from stakeho

③ Elaboration → Creation of analysis model
↓
covers information, behavioral and functional aspects.

④ Negotation → Agree on a model suitable for developers and customer.

[3 are remaining] → categorized requirements into suitable subsets.

↳ Relations among requirements are identified.

Q.] Difficulties in Elicitation?

→ ① Boundary of project not defined properly.
② Unecessary technical details by custome confuse us.
③ sometimes customers omit detail that is believed to be obvious.
④ Some requirements might conflict with others.

⑤ Specification → Documentation of the system requirement

⑥ Requirements validation → formal technical review mechanism.

⑦ Requirement management →
↳ used to identify, control and track requirements and changes as projects proceeds.

⊛ Traceability tables
→
① Features traceability table → shows how requirements relate to customer observable features.

New Assg? → Traceability table

PAGE NO. 42
DATE :

② Source traceability table → identifies source of each requirement.

③ Dependancy traceability table → indicate relations among the requirements.

④ subsystem traceability table
↳ requirements

[Incomplete] ⑤ Interface traceability table

* Functional Requirements depends on Business requirements.

Testcases

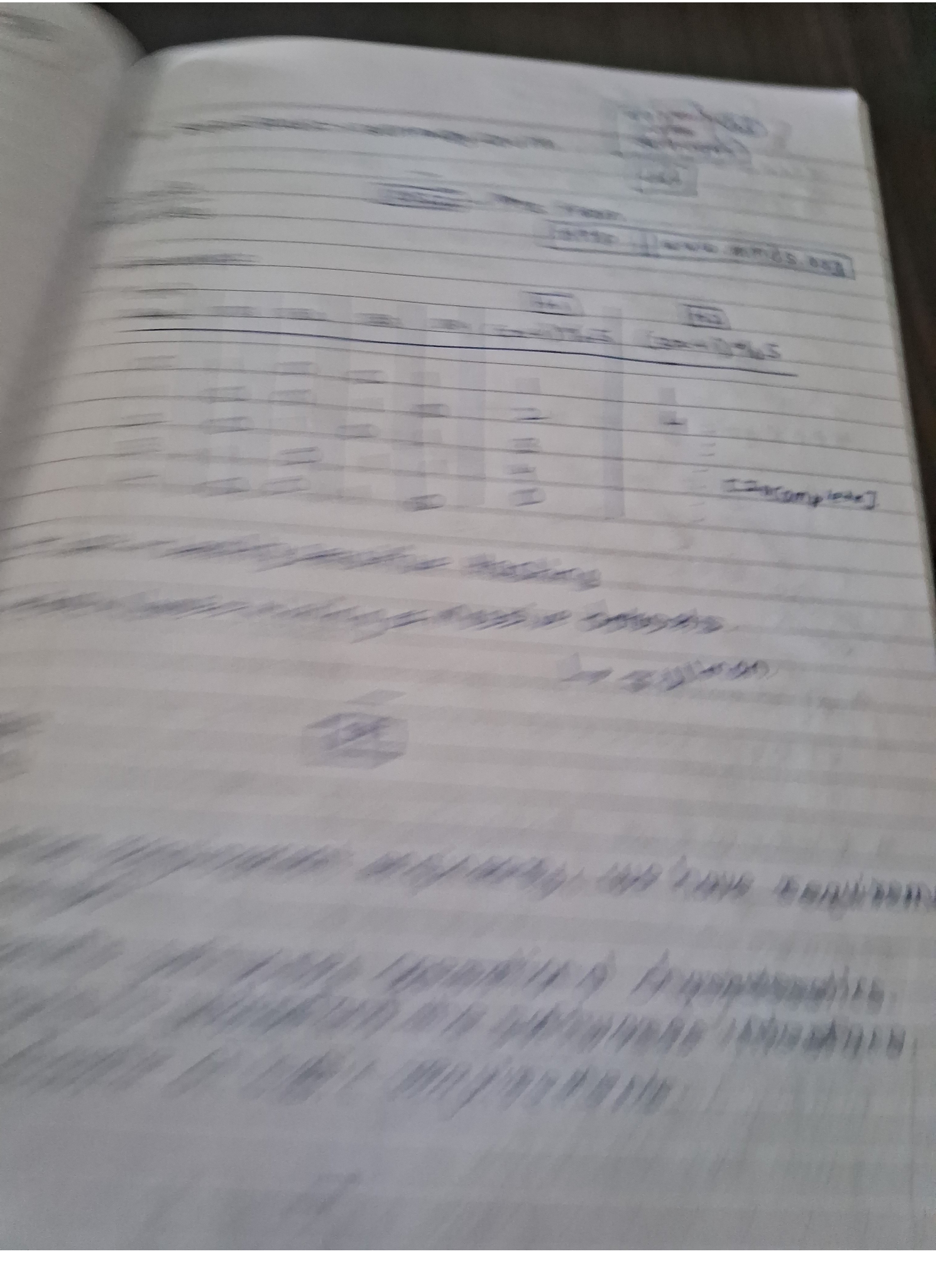
Finished this unit

—

OS

9/02/24
Thursday

* Disk space Allocation for File systems
→
① contigous allocation scheme.
↳ But poblems of fragmentation
Good for
if you want th +
your dis



coupling ∝ 1/cohesive

PAGE NO. 46
DATE :

*Analysis Modelling principles

→ ① Information domain
↓
our DFD's covers this parts

② Function domain
↓
our UML's cover this.

③ Behavioural domain
↓
interaction with external users.

④ Divide and conquer

Analysis Rules of thumb

* Analysis model → Flow oriented
→ class based

* Structured analysis — using DFD's

* Object oriented analysis — using UML's

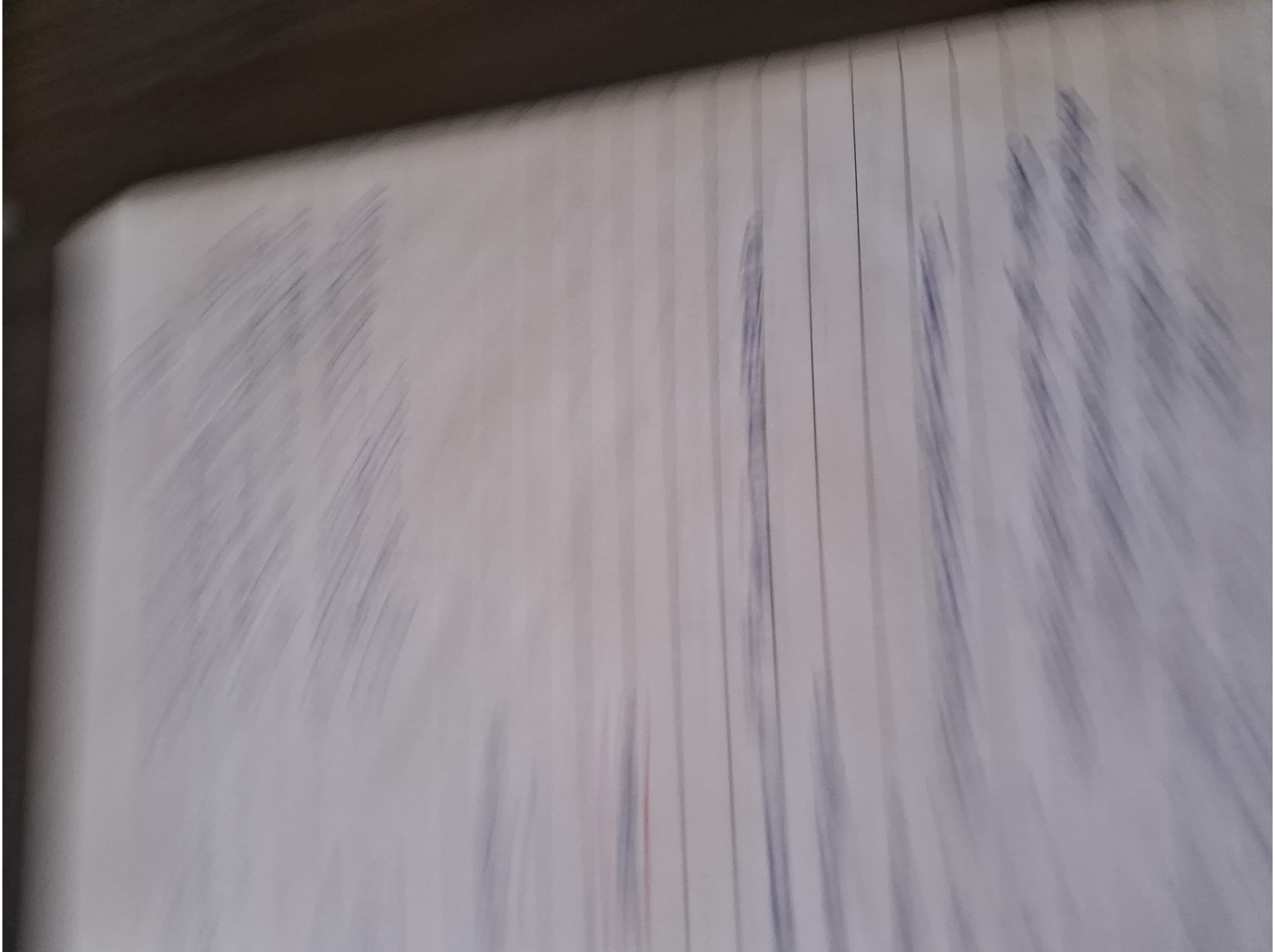
* Elements of Analysis model → Behavioural based

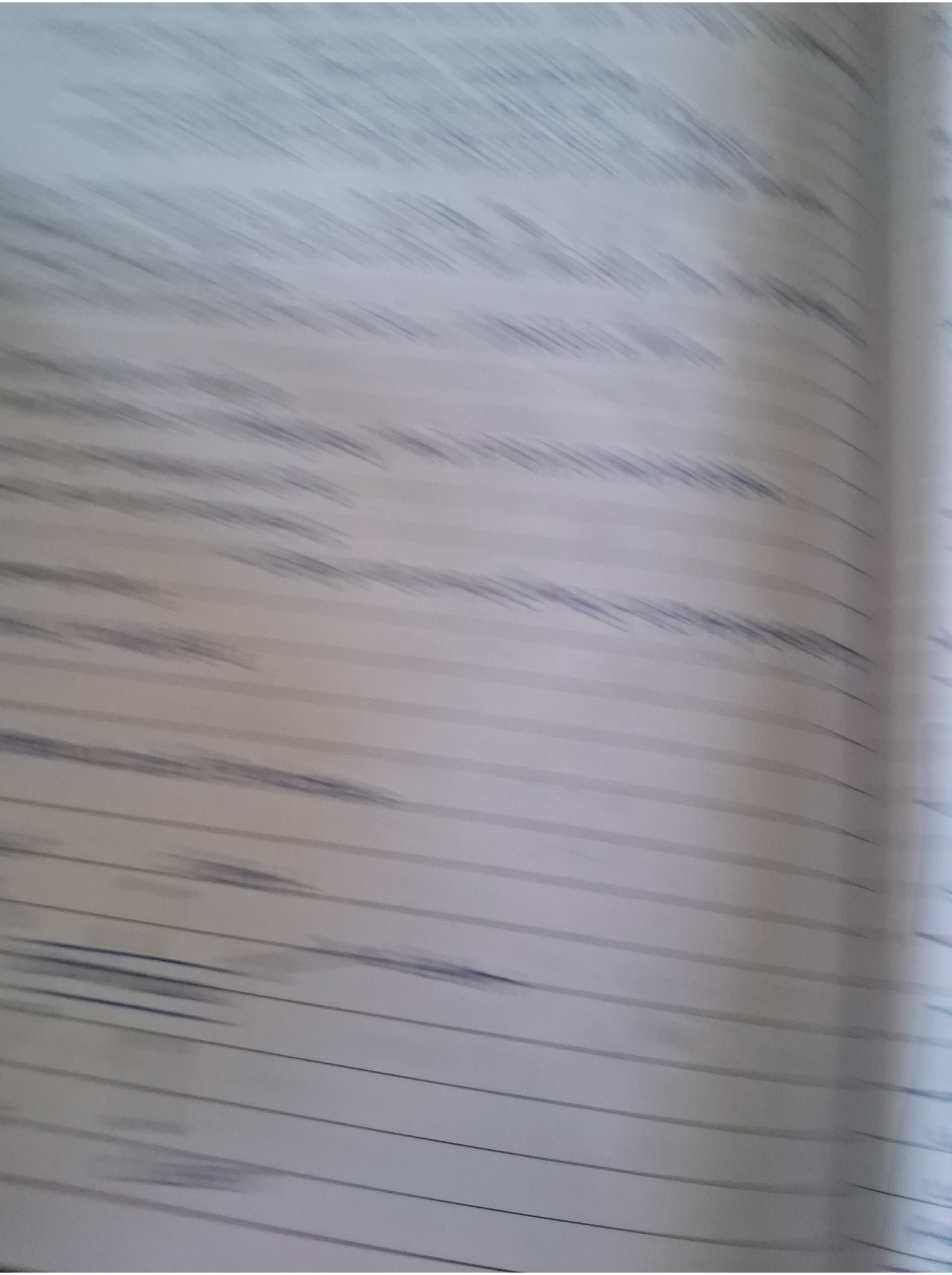
Flow oriented

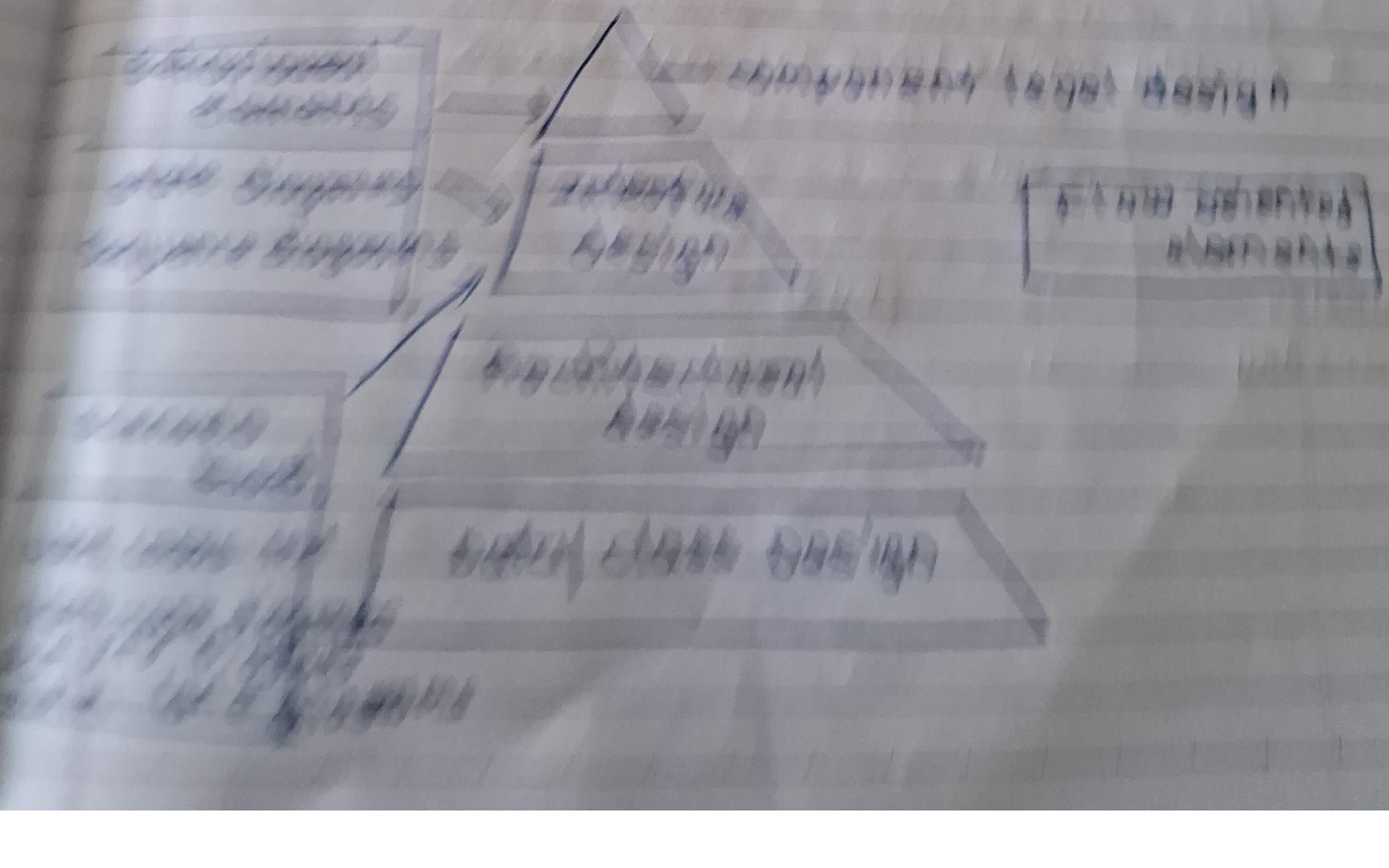
class based

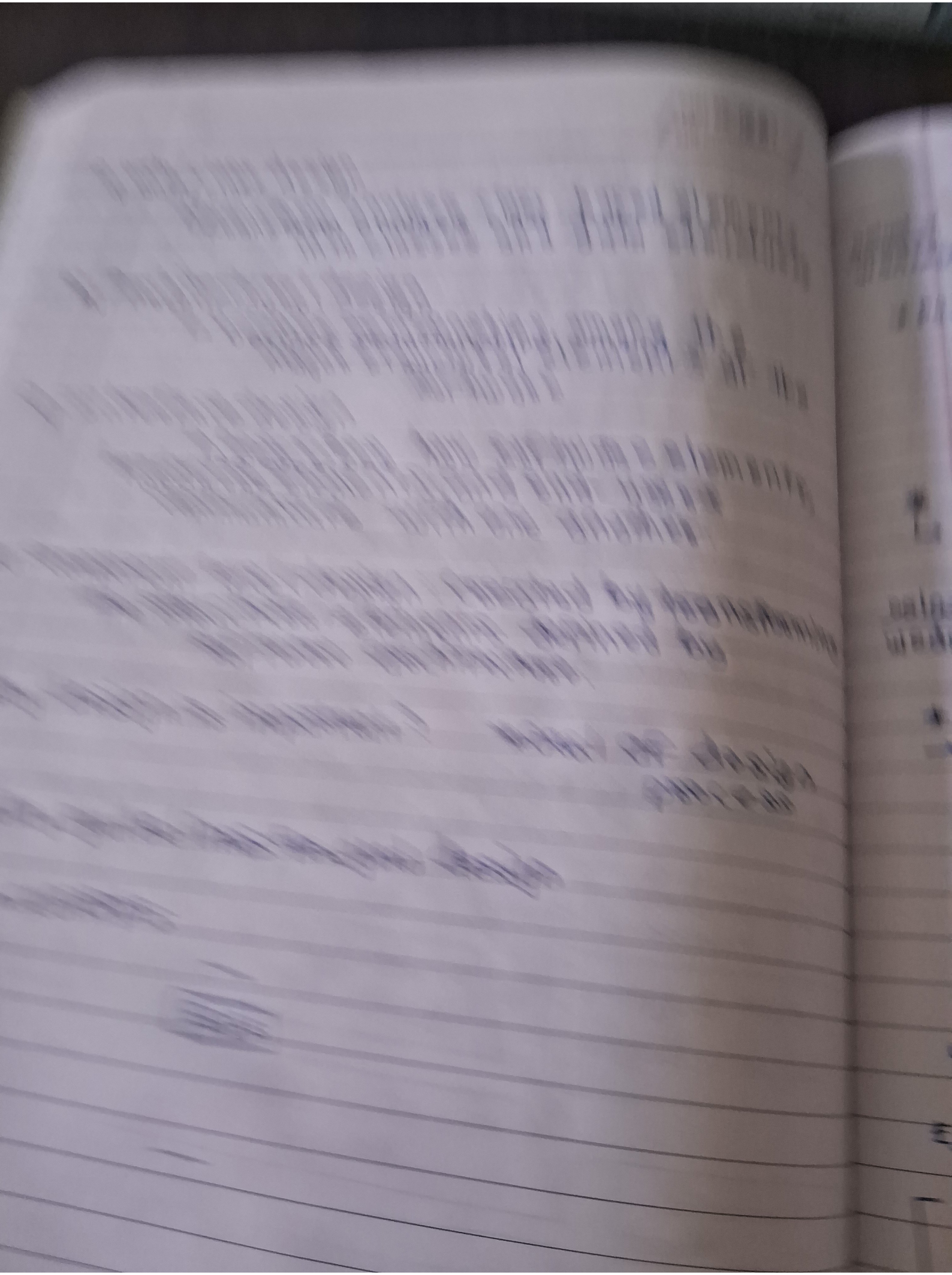
DFD

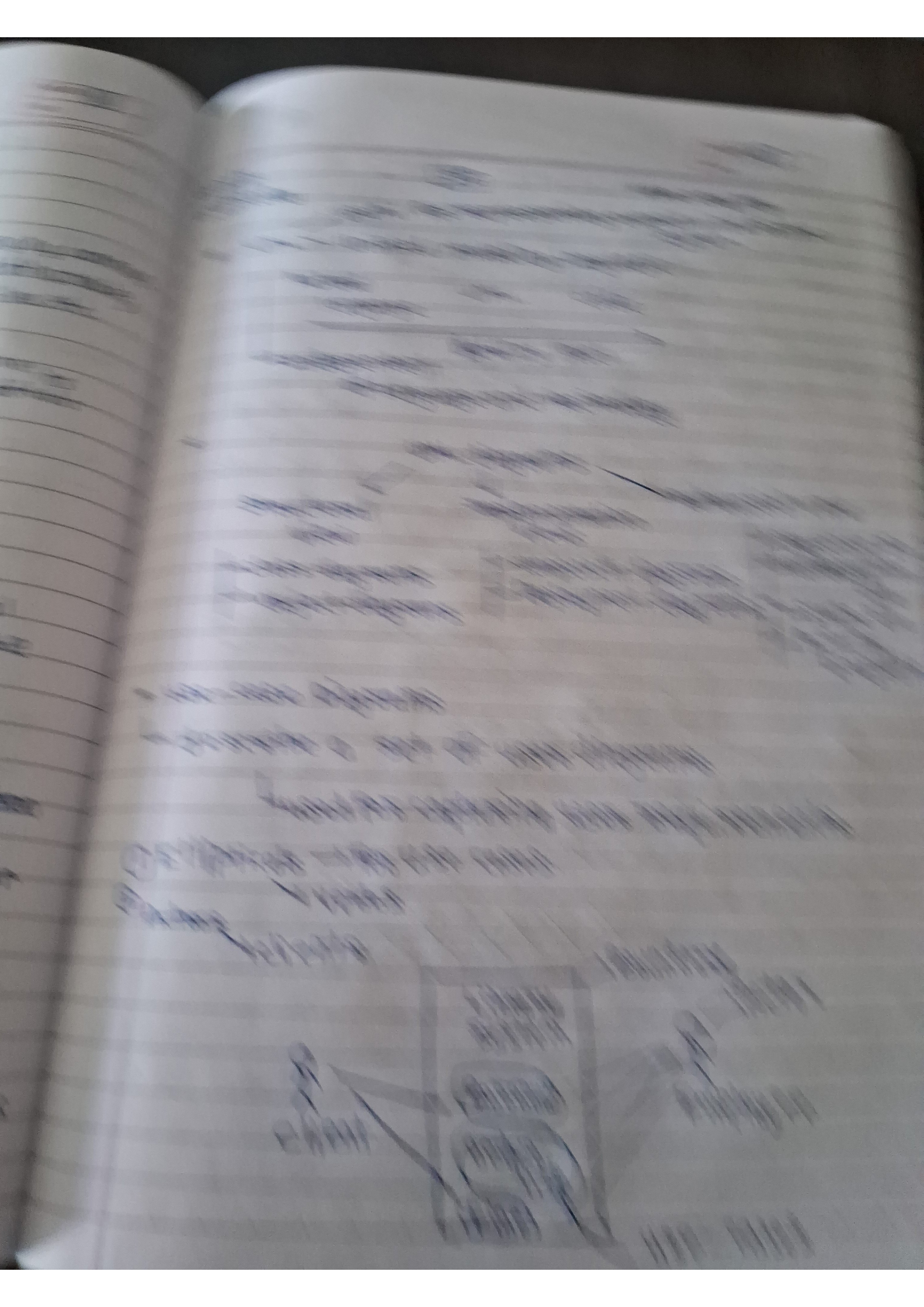
UML











Two types of lines used → Association [ ——— ]

→ ~~Specialisa~~ Generalisation [ ——→ ] points to parent use case

Imp other two lines

a) Include:

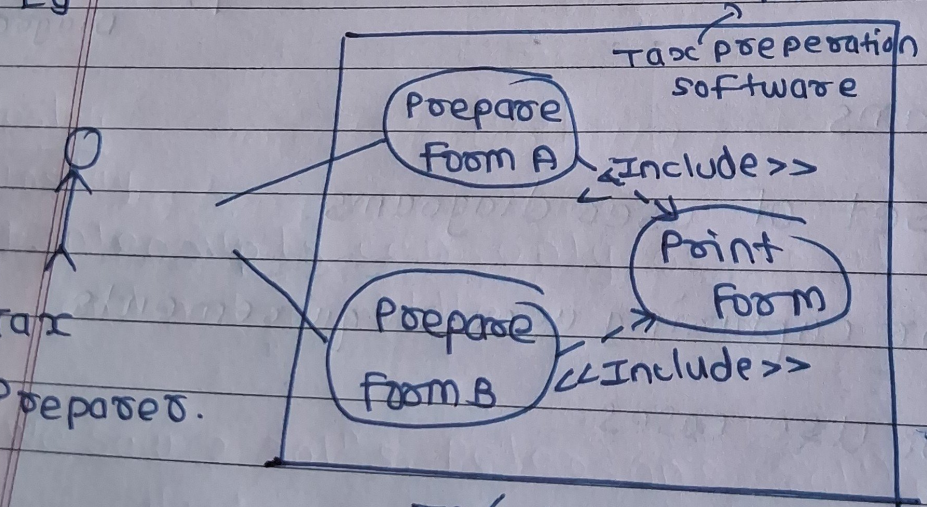
Base case <<include>> ▶ pointing to include use case.

↳ used when we have similar behaviour across various use cases.

b) Extend:

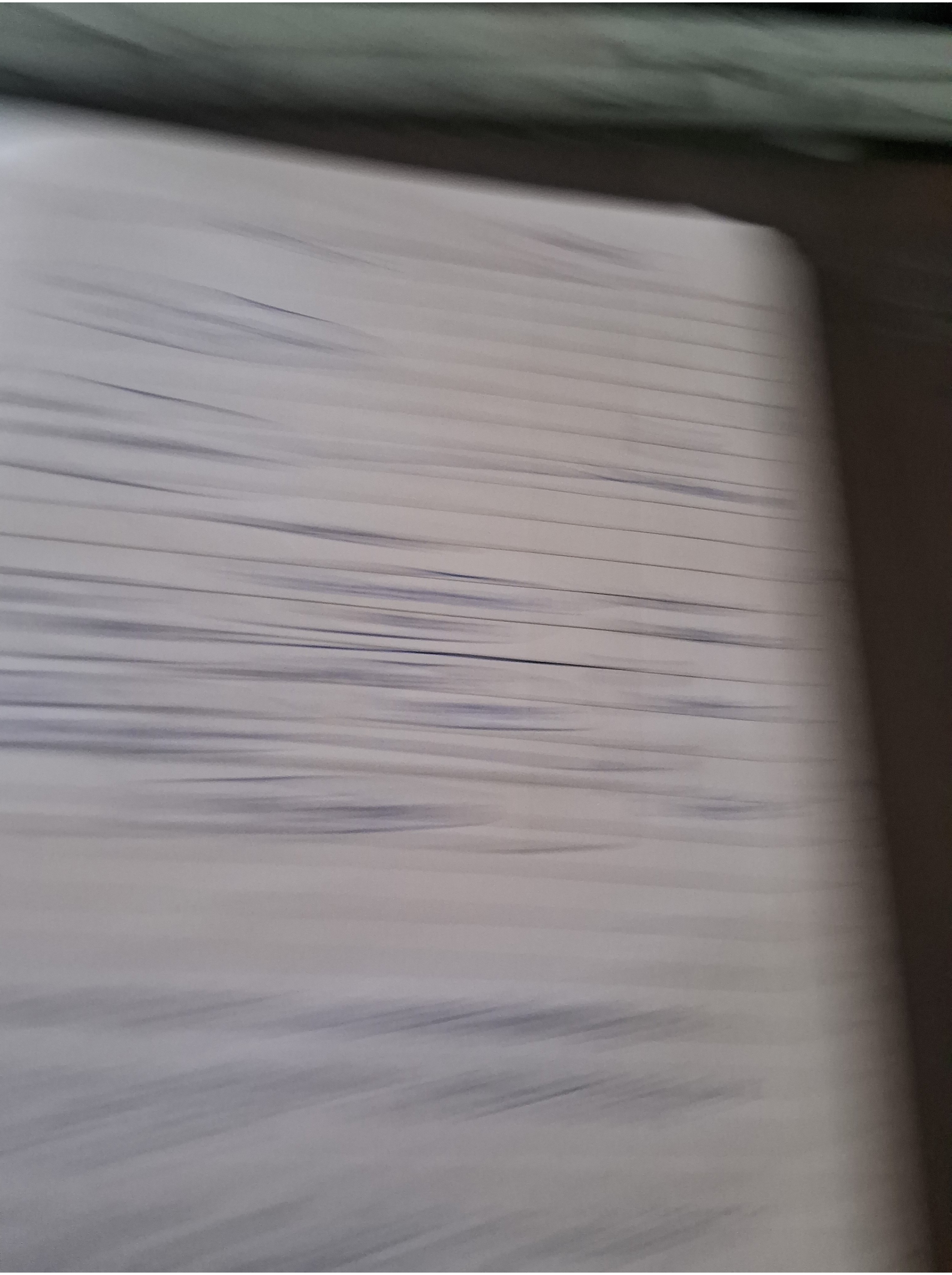
<<extend>> - - - - - →

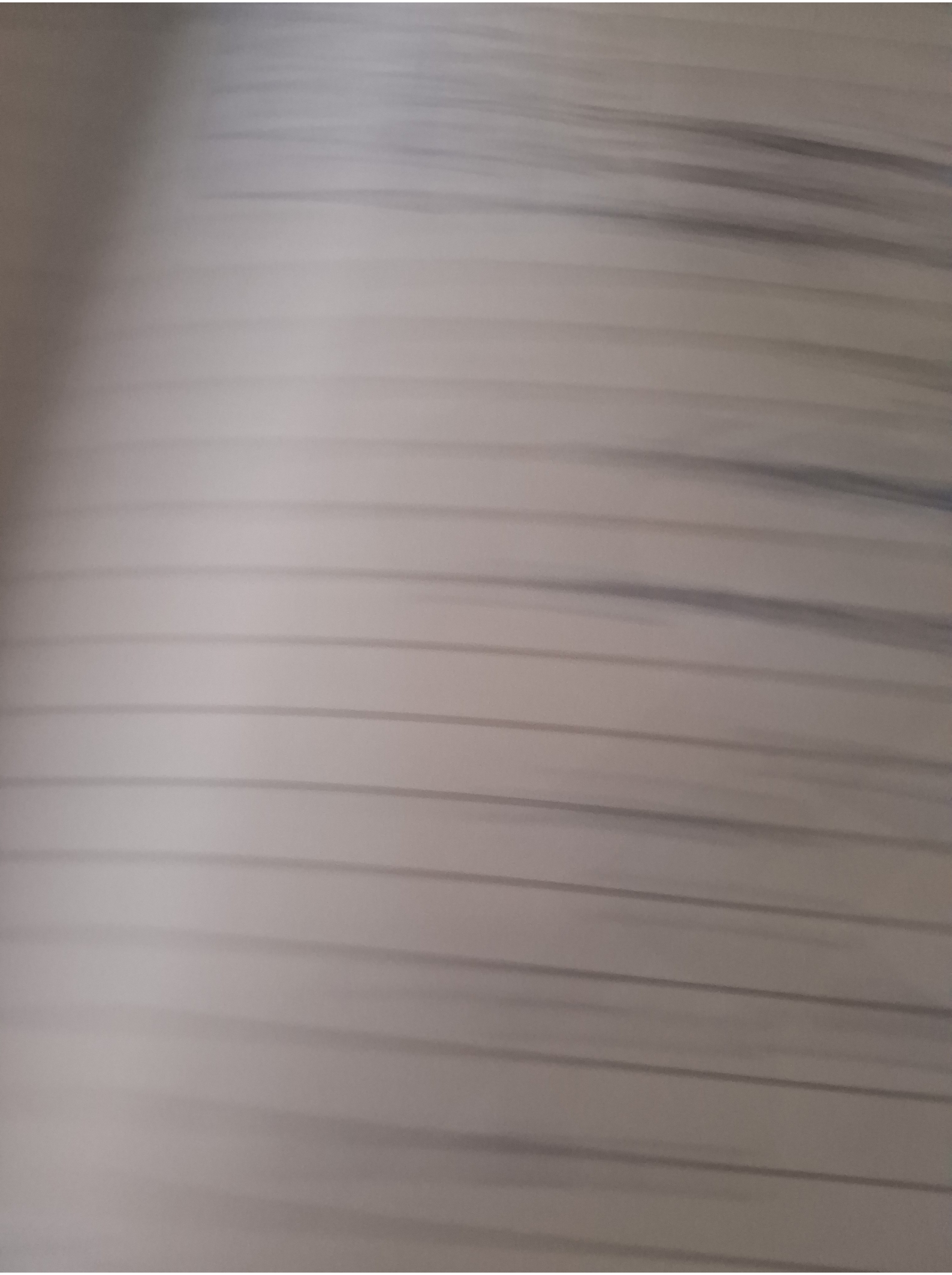
Eg:




Tax preparation software ~~product~~.

→





* components used to model: source code files, packages, versioning

SE

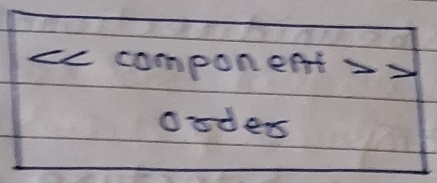
Monday

① user view → use's.
② structural view → class diagrams.
③

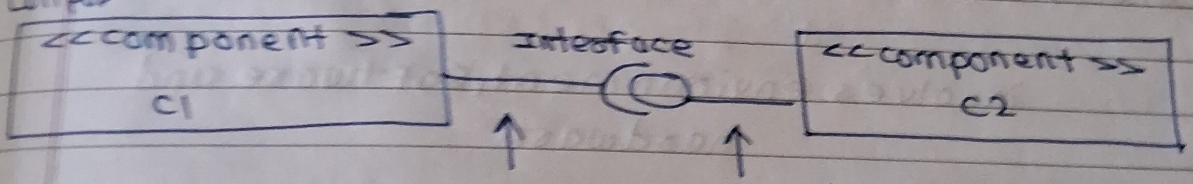
* Component Diagrams → inherited from class diagrams.

Used for depicting components, interfaces and dependancy, aggregation, constraint, generalization, association and realization relationships.

* Components →



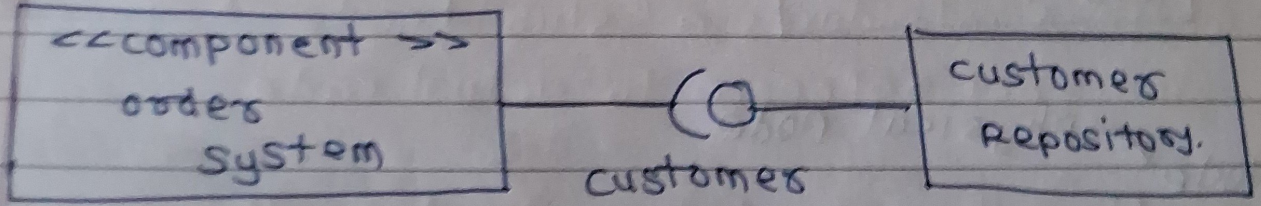

Lollipop →




Required interface

provided interface.

Eg: Interface [Order System].




* Posts.

Mandatory relationships → composition.
optional relationships → Aggregation

PAGE NO. 99
DATE :

Dependancies are shown with dotted lines.

Eg ----->

Eg:
* <<components>>
Item.java

----->

<<components>>
Loan.java

Dependant component

who is going to provide dependancy

* Also can be used to model the database.

* Dependancy can be unidirectional or bidirectional

* deployment diagram
↳ shows execution architecture of diagram.
↓
involves environment, software and hardware.

↳ Middlewares, software and hardware node.

Eg:

<<artifacts>>
boot, club, app

Deployment specification

* Node → Physical node
↳ Execution environment node.

13th – 14th April Viva SE

Viva centered.

UML, ERD's, DFD's

PAGE NO. 80
DATE :

* Imp UML diagrams.

| | | |
|---|---|---|
| ① Use-case diagram | → | captures user view |
| ② class diagram | | Structural view |
| ③ Sequence Diagrams | → | [Dynamic view] |
| ④ State diagrams | → | [Dynamic view] |
| ⑤ Activity Diagrams | → | [Dynamic view] |
| ⑥ component diagram | → | [static view] |
| ⑦ Deployment Diagram | → | [static view] |

–

Disadvantages

RE

① site specific.
② Low energy density.
③ Transmission cost.

26/03/2024
Tuesday

Unit 5 Tidal Energy

ESE pattern → MCQ, Numericals, Long and short answers.

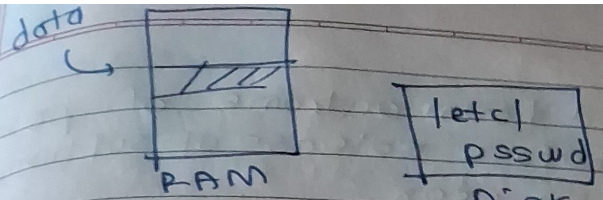
*connection of tides with gravitational field.

Topics to Study

① 71% of earth is water → <5% usable water
→ ~66% sea water.

② Energy associated with oceans → tides energy, waves energy, temperature diff.

③ Tidal Energy advantages and disadvantages




SE

Key process indicators
[KPI]

Calculus
Tuesday

Unit 4

* Software metrics → Qualitative
→ Quantative

* Estimates for cost
↙ ↘
Schedule Estimates | Duration Estimation

3 type of metrics
① Product metrics
② Process —11— } → quantitative measures
③ Project —11—

Project metric ↑ | Process metric ↑

* Reasons of measure
① To characterize.
② To evaluate.
③ To predict
④ To improve.

tactical ~~strategy~~ → short time
strategic → long term.

* Process metrics
↳ ↳ Data collected across all projects over long periods of time.

Mistake made by developer → errors.
↙
If detected
by user; ~~testing~~ and
~~solved~~ reported.
↳ defects.

PAGE NO. 97
DATE :

* Process Metrics
→ Following are derived from outcome of process
① Errors detected before release of software
② Defects reported by user
③ Work products delivered
④ Human effort expended
⑤ Time expended
⑥ Conformance to Schedule
⑦ Time and effort to complete each generic activity

* Project metrics
→ Used from past similar projects used
to estimate time and effort
→ Project metrics:
Productivity, Errors, Defects created, review hrs, function points, delivered source lines of code
→ Errors encountered during each generic activity

lines of code is direct measure of product metrics

organic mode
↳ small project, small team

→ semi-detached mode
↳ little complex projects

embedded mode.
↳ Developed within a tight set of hardware and software constraints.

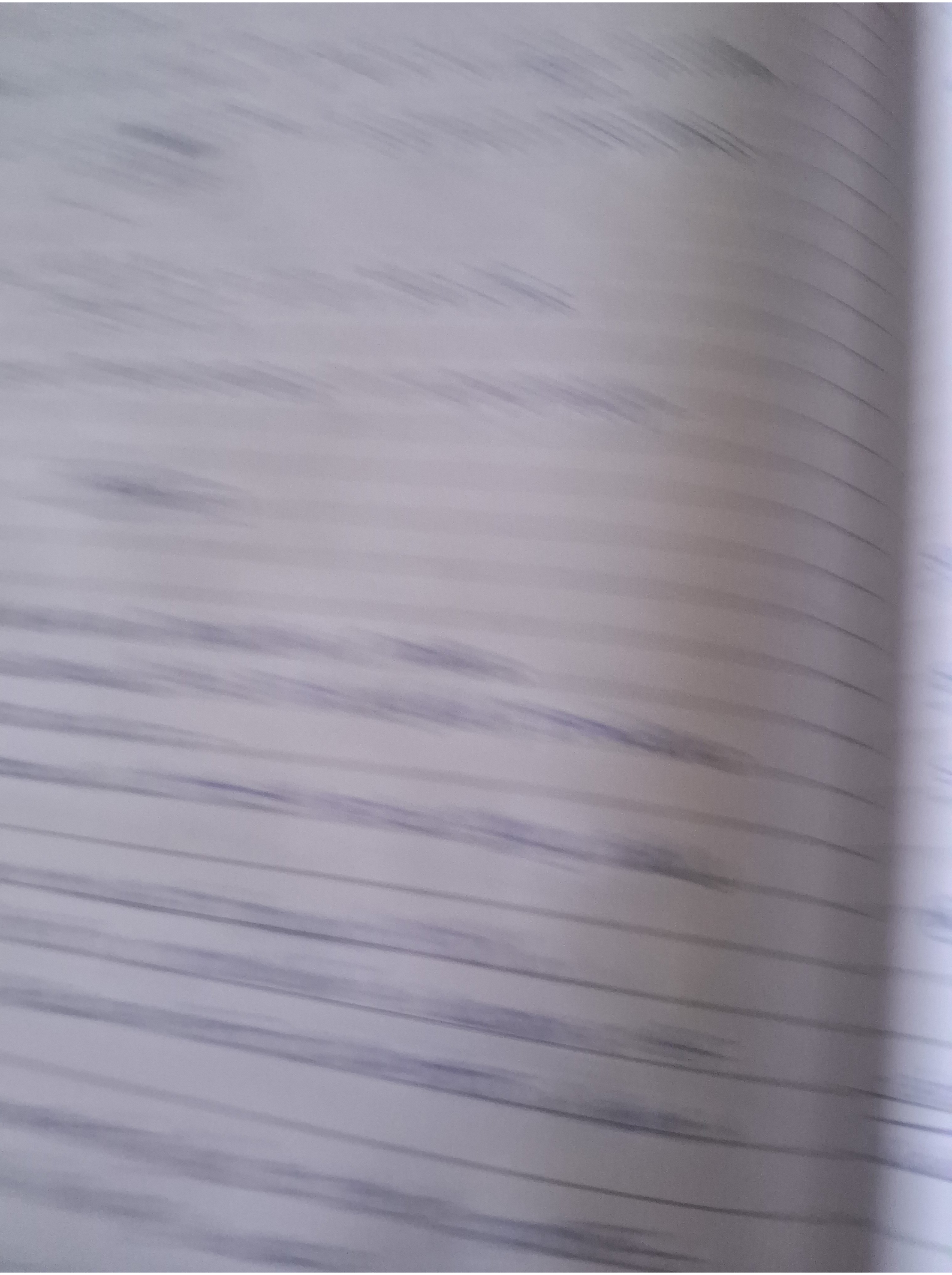
basic cocomo
Effort = $a_1 \times (KLOC)^{b_2}$ per month.
Time for development = $b_1 \times (Effort)^{b_2}$ months.

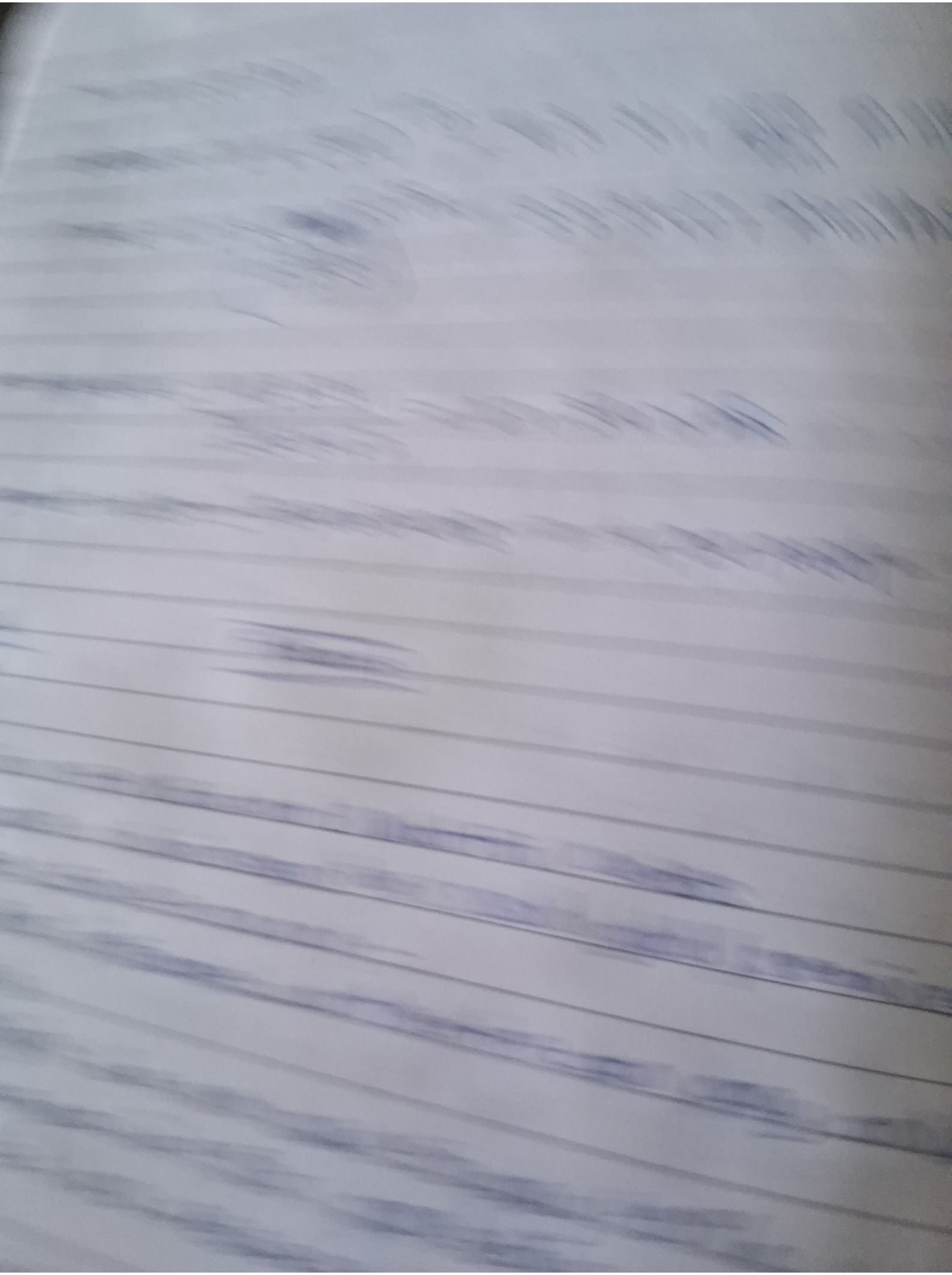
organic : Effort = $2.4 (KLOC)^{1.05}$ PM
semi-detached : Effort = $3 (KLOC)^{1.12}$ PM
Embedded. Effort $3.6 (KLOC)^{1.20}$ PM
↳ thousands line of codes.

organic : $T_{dev} = 2.5 (Effort)^{0.38}$ months.
semi-detached : $T_{dev} = 2.5 (Effort)^{0.35}$ months.
Embedded : $T_{dev} = 2.5 (Effort)^{0.32}$ months.

organic project
32,000 lines of code.
Avg salary of software engineers = Rs. 15,000/- pm.

→



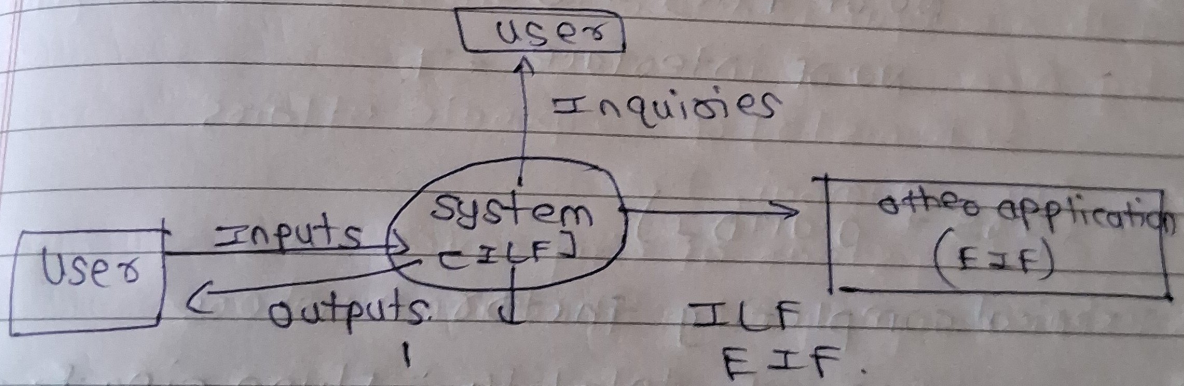


Product metric

* Disadvantages of COCOMO model
1]. Rough estimates can be erroneous from

* Project Metric → Function point analysis
1]. Proposed by Albrecht in 1983
2]. estimate size of software product from problem statement
↳ More the required no. functionalites, the bigger is the project and the more is the no. of files required.

3]. Unit of calculation
↳ function points.




① No. of external entities → Input screen and tables.
② No. of external output → output screen and reports.
③ No. of external inquires → Prompts and interrupt.
④ No. of internal files → Databases and directories
⑤ No. of external interfaces → shared databases and shared routines.

*

① Function points (FP) = UFP * TCF

| Function Units | Low | Avg | High |
|---|---|---|---|
| EI | 3 | 4 | 6 |
| EO | 4 | 5 | 7 |
| EQ | 3 | 4 | 6 |
| ILF | 7 | 10 | 15 |
| EIF | 5 | 7 | 10 |

Eg: if Avg

└ UFP = 4 * No. of inputs +

No. of outputs * 5 +

No. of inquiries * 4 +

No. of files * 10 +

No. of interface * 7.

② TCP = (0.65 * 0.01TDI).

└ Technical complexity factor.

TDI → summation of total 14 influencing factors

③ CAF = 0.65 + (0.01 * F) .... Complexity adjustment factor.

VAF = 0.65 + 0.01 * TDI .... Value adjustment factor

F = [4 * scale]

PAGE NO. (188)
DATE :

rub specifications:
0 = No influence
1 = Incidental
2 = Moderate
3 = Avg
4 = Significant
5 = Essential.

No. of inputs.
UFP: while entering data of an employee, we enter various fields but as it belongs to one employee only. So it is only 1 input.

ex7. calculate the functionpoint, productivity and cost per function for software application with multiple processing factors ~~by using~~ 5, 1, 0, 4, 3, 5, 4, 3, 4, 5, 2, 3, 4, 2 by using following info:

No. of EI (Avg): 22
No. of EO (Low): 45.
No. of EQ (High): 06,
No. of ILF (Avg): 05.
No. of ELF (Low): 02.

person month
↑
PM
Effort: 37 ~~per month~~.
Cost: $7520 per month.

UFP = 22 * 4 + 45 * 4 + 6 * 6 + 5 * 10 + 2 * 5.
= 364.

TCF = 45.

FP = 364 * 1.1 ≈ 400.4
= 400.

→

400 FP in 37 persons month. (Productivity)

↳ in 1 person month = $\frac{400}{37} = 10.81 \approx 11$ FP/PM.

Cost per month = $7520 for 11 FP's.

So for one FP = $\frac{7520}{11} = 683.636$

—

25/04/24
Friday

## Finance

* Features of Limited companies

1. Incorporated associations.
2. Certificate of Incorporation (Birth certificate) with a unique Number is alloted to company by MCA
   ↳ Ministry of corporate affairs.
   Eg: ROC - Mumbai, ~~Roc~~
   ↳ Registrar of companies
3. CIN → company Identification numbers.

Perpetual existence

↳ Akbaldah all of