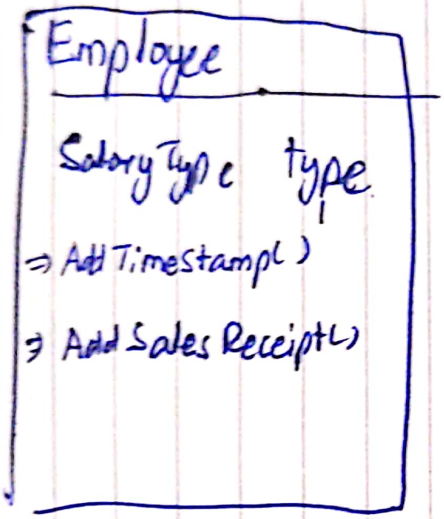
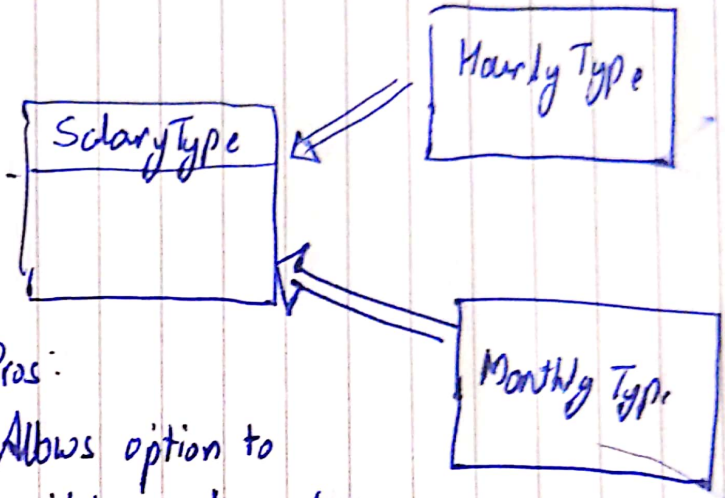
Method 1





Pros:

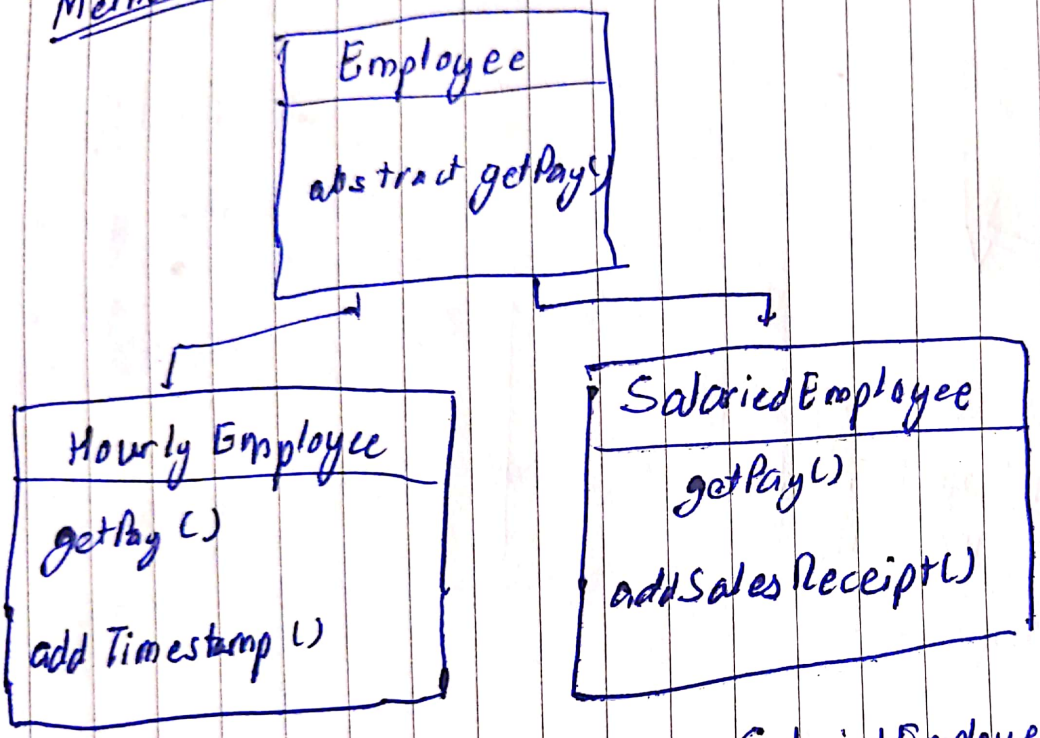
⇒ Allows option to switch employee type by directly changing Salary Type object

Cons:

⇒ Unnecessary methods.
For Hourly employee it doesn't make sense to have addSalesReceipt since it is only for Salaried Employee. Method should not be there in its object.

⇒ Change not needed. Eg:- Manager, Sales Representative are of type salary Employee.
- Their type doesn't need to change.
Subtraction is not necessary.

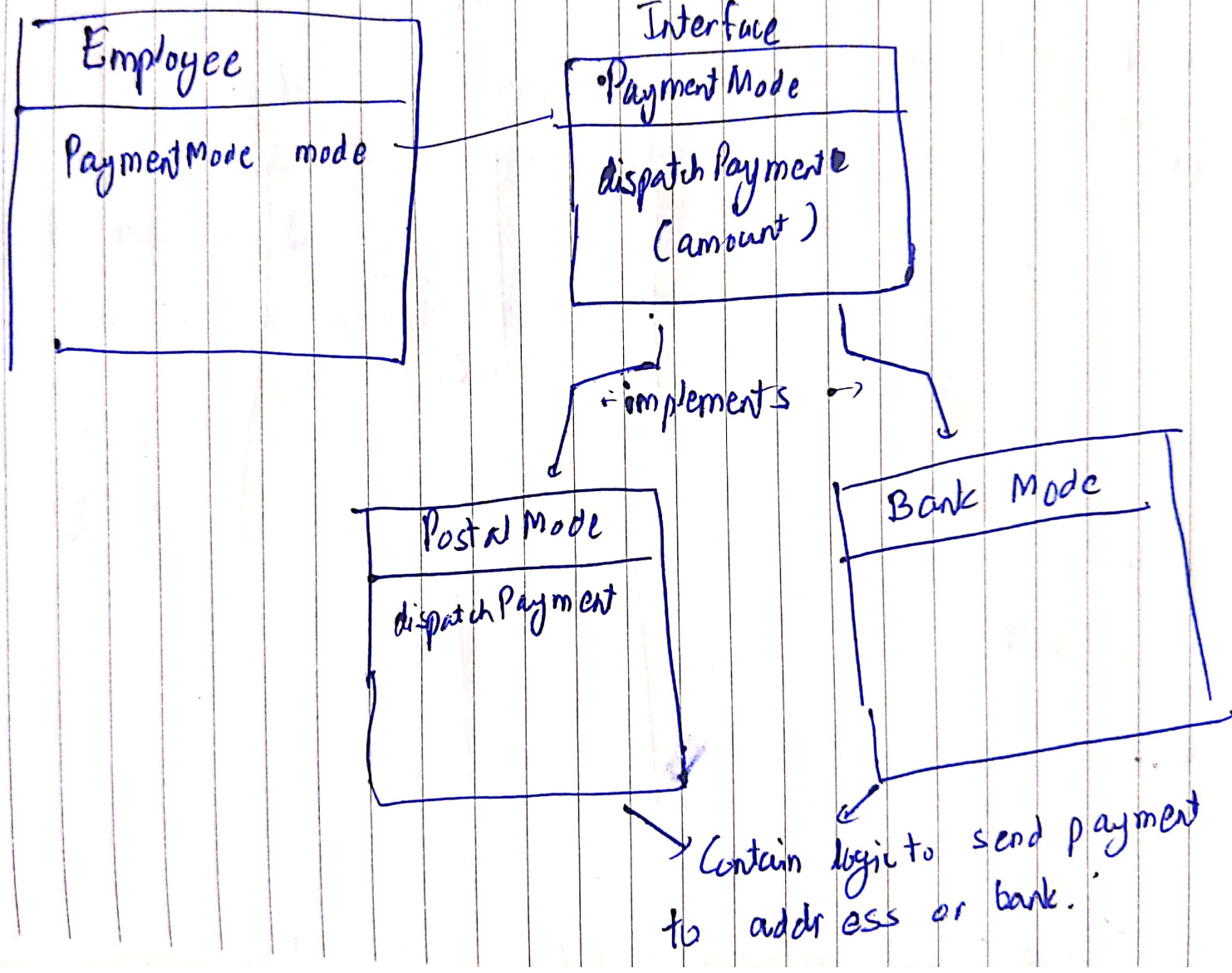
Method 2



Methods are at proper places. SalariedEmployee doesn't have addTimestamp() & Hourly doesn't have addSales

Cons:
Subtraction not possible

Payment Modes: 3 Types



Database: Access

Distributed Logic

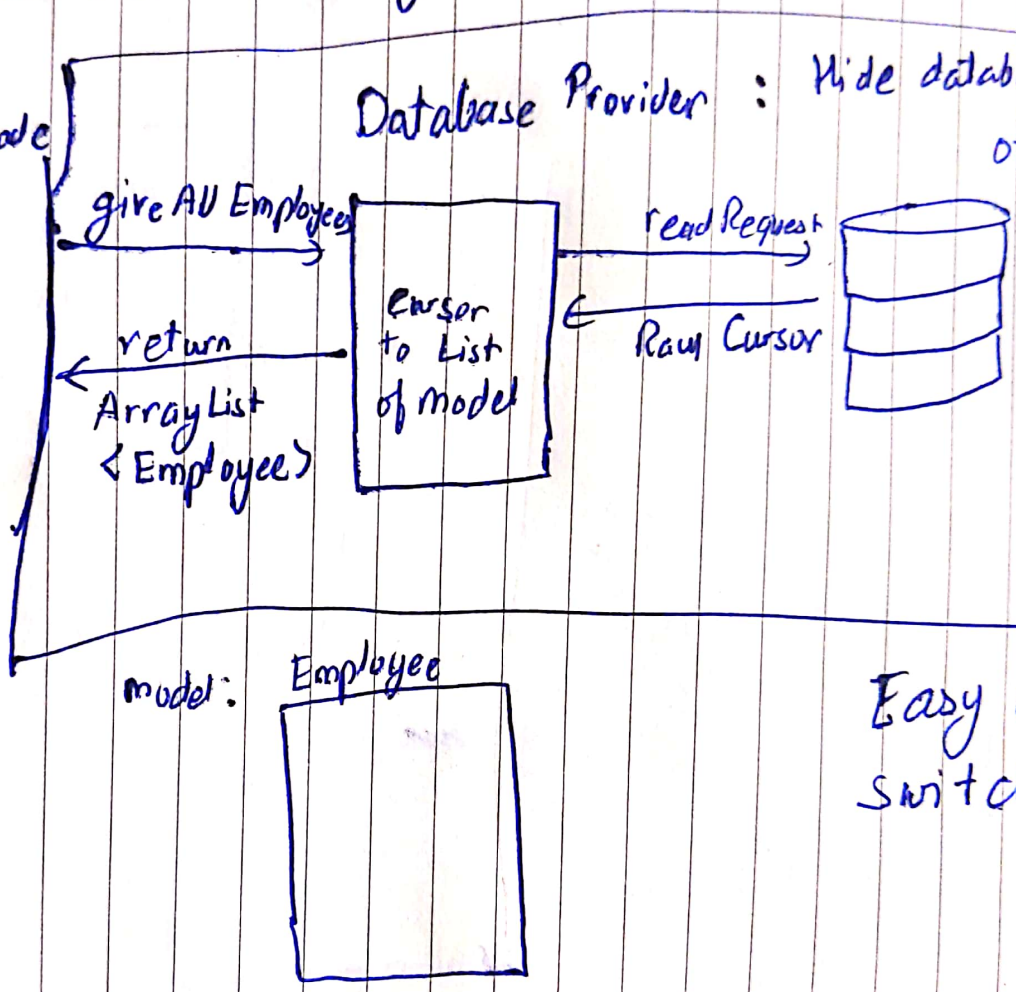
- Every code should write to database on its own if needed.
- If change needed multiple classes needs to change

VS

Centralized Logic.

- One class should write to database only.
- Only one place should be changed for changing database.
- More robust.
- Better approach.

Database Abstraction Layer



model: Employee

Easy database switching.

Stateless Design:

The methods and classes do not keep an ArrayList of Employee or time stamps in memory. Every time we add or delete changes are written to database.

Since data is not saved in classes They are behaviour only.

When displaying employees we fetch data to display & destroy immediately.

Reason: Data can be large. Lakhs of employees can be there.
Better to keep in database only.
Using read and write when necessary state can be managed in database.