Chain Responsibility

.. loose Coupling in software design is achieved using chain of responsibility. Then the object in the chain will decide themselves who will be processing the request is required to be sent to the next object in the chain or not

The object in the chain will decide themselves who will be processing the request and whether the request is required to be sent to the next object in the chain or not

→ chain of responsibility reduce the coupling degree. Decoupling it will request the sender and the receiver

→ Simplified object. The object does not have need to know the chain structure.

→ Enhance flexibility of object assigned duties By changing the numbers within the chain or change their order, allow dynamic adding or deleting responsibility.

→ Increase the request processing new class of very convenient.

Disadvantages:-

→ The request must be received not guarantee

→ The performance of the system will be affected,

Scanned with CamScanner

but also in the code debugging is not easy
may cause cycle call.

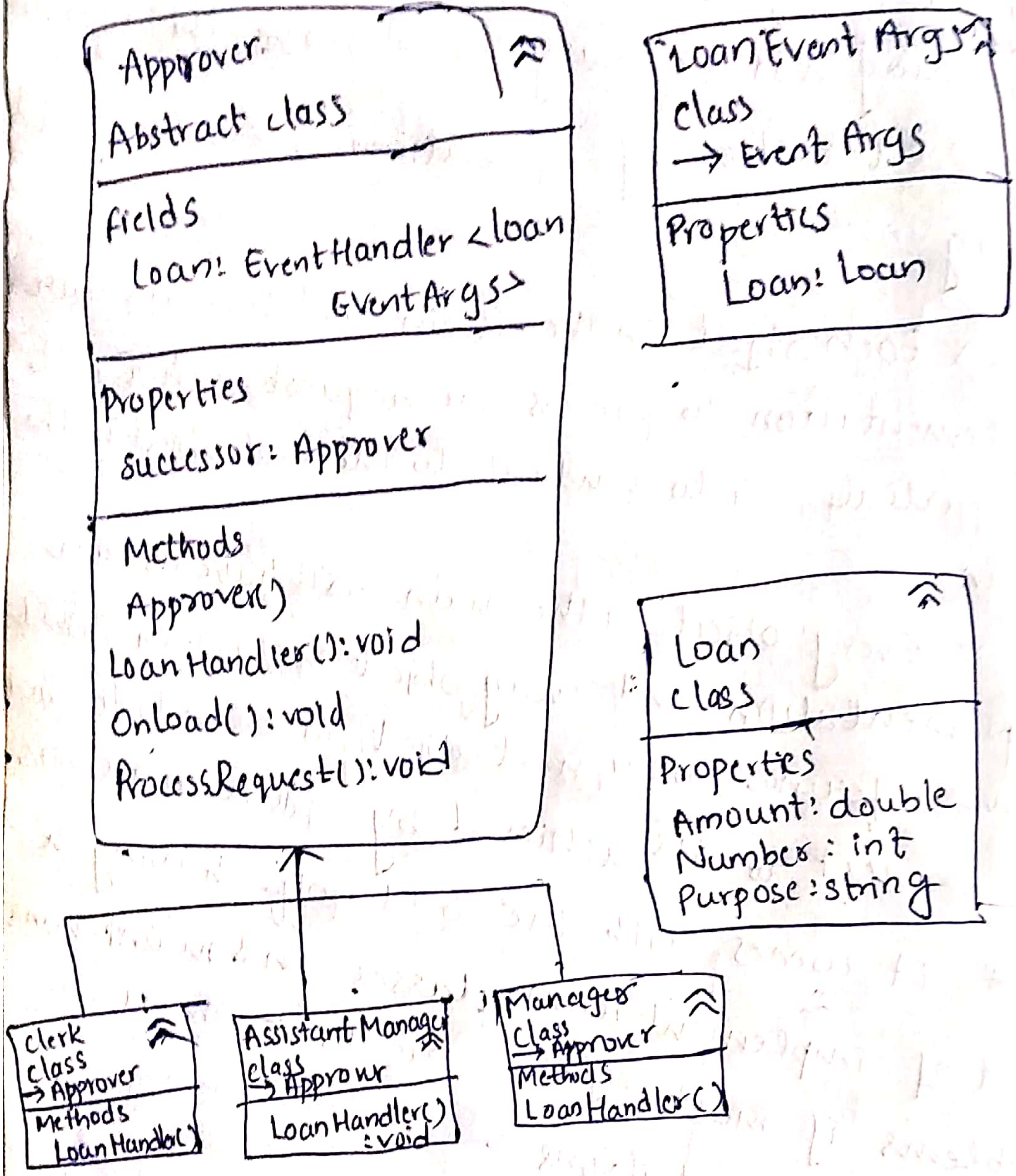
→It may not be easy to observe the
characteristics of operation, due to
debug.

* Each object in the chain will have it's own
implementation to process the request, either full
or partial or to send it to the next object in the
chain.

* Every object in the chain will have it's own
implementation and every object in the chain should
have reference to the next object in chain to forward
the request to, it's achieved by java composition

* It comes with the trade-off having a
lot of implementation classes and maintenance
problems if most of the code in common in
all the implementations.

class Diagram of chain of responsibility



Approver
Abstract class
Fields
Loan: EventHandler <Loan EventArgs>
Properties
successor: Approver
Methods
Approver()
LoanHandler(): void
Onload(): void
ProcessRequest(): void
LoanEventArgs
class
→ EventArgs
Properties
Loan: Loan
Loan
class
Properties
Amount: double
Number: int
Purpose: string
Clerk
class
→ Approver
Methods
LoanHandler()
Assistant Manager
class
→ Approver
LoanHandler() : void
Manager
Class
→ Approver
Methods
LoanHandler()