=>

IS -> IOLPAJ

Core Java => 10 -> 8 & above

↳ JDBC -> Fundamental

↳ JDBC -> How do not -> loading
getConnection
Statement
result -> Query
next
close

=> CRUD -> JDBC :- 100% :- 60-70% | sufficient

=> JEE -> Servlet | JSP

↓

What is servlet
why servlet
=> life cycle of servlet
=> MVC -> 1 project

=> { JSP -> added advantage }

↓

tag => Basic =>

{ Servlet }

=> core java => strong

{ Advanced (JDBC + Servlet + JSP)
Intermediate
1 Project }

{ Advance }
Framework

⇐ Strong Fundamental | Foundation on Framework & Spring Framework

⇒ ⇒ Spring Framework :-

⇒ Spring Core
Spring AOP
Spring WebMVC
Spring JDBC
Spring RFS
Spring ORM ⇒

Spring Framework
⇓
Spring Boot
⇓
Developing Spring Application in simple way/form

⇐ 2002 → 0.9v ⇒ Alpha ⇒ Rod Jhonson
2004 → 1.0 (Production releases).
2006 → Spring 2.0
2009 → Spring 3.0 ⇒ ⇒ popular
⇒ Spring 5 ⇒
Dec 2022 Spring 6 ⇒

⇒ SpringBoot = 3

⇒ Framework :-

↳ To make our life easy
↳ semi Develop software

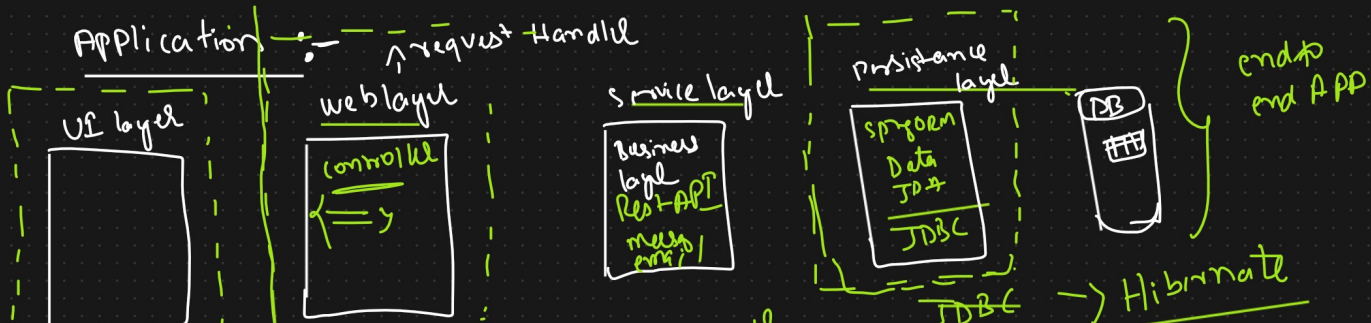
⇒ JDBC ⇒ Load Driver, Register
get Connection
create stmt
rs ← exe Query
close

⇒ USA → 1 company
India → 2 company
learner → intern

⇒ Java Appl ⇐ ⇒ DB

⇒ Boiler plate code.

JDBC
⇓
⇒ Hibernate :- ⇐



=> JDBC, Servlet -> web layer -> Persistance

=> Framework -> ORM framework (eg: Hibernate -> Persistance)
-> web framework -> web layer -> Request Handler
eg :- Struts

=> Servlet :- Forms :- Registration, login => fetching Data from UI

=> servlet => request handler
-> web layer
=> req. getParameter } boilerplate

=> Strut outdated :-

ORM framework =>
web framework =>
Both

=> Spring Application Development Framework

=> Develop => Handling Req (web layer)
Business logic (service layer)
Persistate

APP :- End to End

=> spring -> end to End Application {of Distributed App}
⇓
Restfull }

=> { : Mytra :-> Dependent on other App
↓
=> makemytrip => IRTC => => }
google

=> Spring is Framework => collection of framework
⇓
Modules -> Based

{ => spring core, spring DAO, ORM }

=> spring MVC }
⇓
spring REST

=> { Versatile -> framework.
loosecoupled framework
{ non invasive } }

create -> Servlet
=> X

=> class mysrvlet implement servlet }
{
}

class myservlet extend HttpServlet } or
{
}

=> spring versatile => integrated framework }

=> spring core } Backbone of spring framework

↳ IOC -> inversion of control
↳ DI -> Dependency injection
↳ field injection
↳ setter injection
↳ constructor injection,

=> spring DAO | spring JDBC => Persistance layer }

=> spring ORM => Data in form of object }

=> spring web mvc :- web Application } MVC => SpringREST Distributed APP

=> spring AOP => separate BL &

=> Security -> mini }

↳ Reactive programing -> springframework 5

⇓

Java }{ works Based on events }

spring core } { ... } => spring App => springboot

spring framework core }

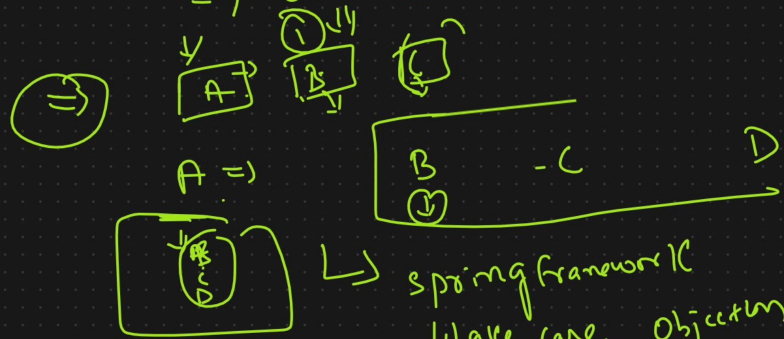
=> IOC
Dependency injection }

=> IOC -> Decoupling object Dependency }

```
class Demo1
{
    m2()
    {
        m1()
    }
}

class Demo2
{
    m1()
    {
    }
}
```

=> Two options => create object
=> extend Demo2



-> spring framework
-> take care Objection } IOC

Spring IOC -> telling -> takecare -> component -> Objects

=> setter injection }
constructor
field injection X

{ IOC } => In way to him to manage Dependency }

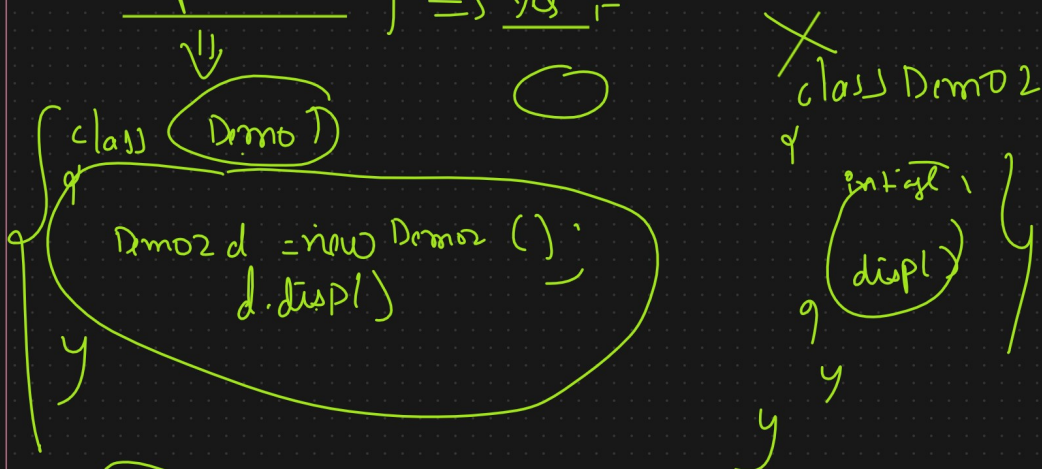
=>

```
class Demo
{
}

class Demo
{
}
```

=> Strategy DP => we will infor -> IOC }

Composition y => Yes :-



=> IOC => y Strategy Design Pattern y

=> DP => y rep -> Best practice -> Solve all problem -> App Develop

=> (oft) => => 3 major DP

- Creational DP => objects rat.
- Structural DP
- Behavioral DP

CDP => Singleton, Factory Design Pattern y

Structural DP => flexibility. => modification => App should not be affected y

Behavioral DP :- change y Algorithm During Runtime without affecting Application

=> y Strategy DP

=> DP -> universal => not just for java

=> IOC -> will create object -> No Dependency of objects

=>

=> Pre-requisites =>

=> Framework =>

=> Types of framework =>

=> To Develop App -> layer => which layer can be used for what ?

=> Spring -> Basic -> Framework

-> Modules -> Spring Framework

=> End to End App | Distributed

=> Learning Spring =>

-> spring core =>

-> IOC

-> DI

=>

=> Object Dependency

-> composition

=> This problem will be resolve by spring framework

{ IOC -> DI }

Spring (or) Spring Framwork => same

To Develop spring easily => springBoot

-> spring core -> one of module <=>

-> IOC -> why ?

{ java project } -> { complications }

=> spring core -> All about -> managing Dependency among classes in App -> with loose coupling

↳ promote -> loose coupling

=> Java class -> IOC -> manage ?

↳ IOC -> To manage -> Develop class -> strategy Design Pattern

=> We have to create / Develop our class -> strategy DP

=> Strategy Design Pattern Pattern Rules.

- => Favour composition not inheritance.
- => Always -> develop -> code to interfaces instead of concrete implementation
- => code -> should be open for extension & -> should -> closed for modification

=> Ask -> IOC -> Create & manage App -> Dependencies.

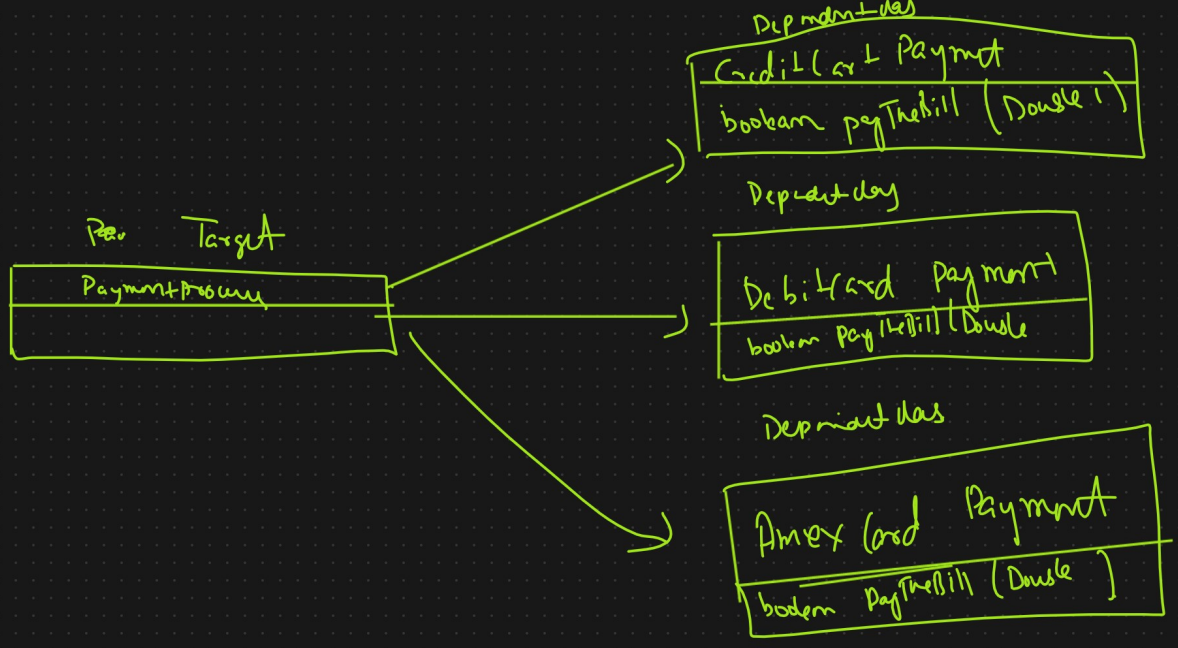
=> Requirment :-

Restaurant :-

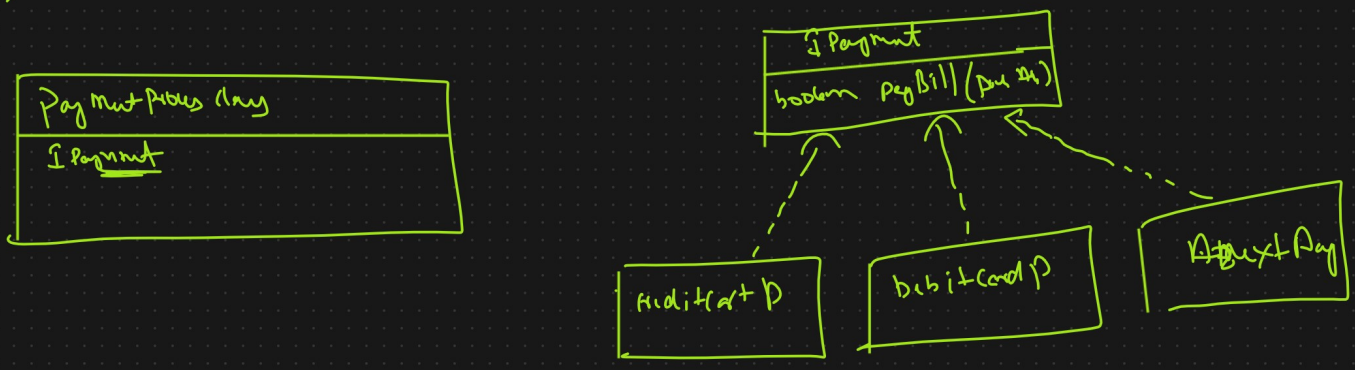
App -> program

IOC -> DI

=> IOC


Dependent class
Credit Card Payment
boolean payTheBill (Double)
Peer Target
PaymentProcessing
Dependent class
Debit Card payment
boolean payTheBill (Double
Dependent class
Amex Card Payment
boolean payTheBill (Double)

=>


PaymentProcessing class
IPayment
IPayment
boolean payBill (Double)
CreditCardP
DebitCardP
NetBankingP