i) Intro Spring Boot

ii) How to start coding

iii) Req. gathering BMS

iv) Class diagram ⟷ model coding

v) ORM introduction

Java 17 +

IntelliJ ⇒ Community edition

Postman

DB ⇒ SQL → MySQL, PostgresSQL . . . .

↓

IDE ⇒ DBEAVER

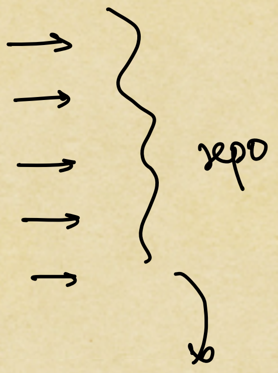
Spring boot ⇒ helps creating BE server very very easily

$$\begin{tikzcd}\text{Java EE [Servlet + JSP + JDBC]} \arrow[r] & \text{Spring} \arrow[r] & \text{SpringMVC} \arrow[r] & \text{SpringBoot} \arrow[d] \\ & & & \text{3.0}\end{tikzcd}$$

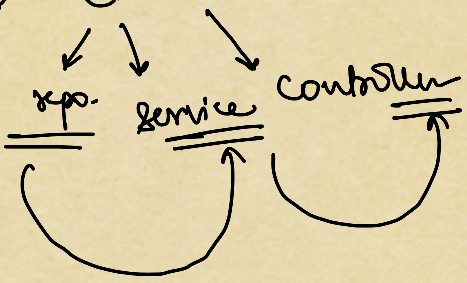
$$\begin{tikzcd}[column sep=small] {} \arrow[d] & {} \arrow[d] & {} \arrow[d] \\ \text{BE} & \text{FE} & \text{SQL queries} \end{tikzcd}$$

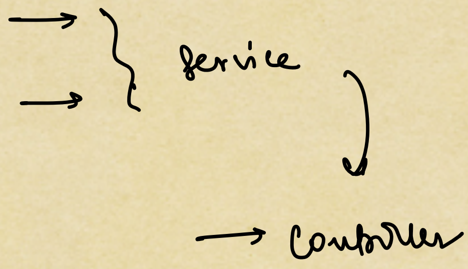
Spring Boot ⇒ IoC container
↓
* Inversion of control

* Dependency Injection



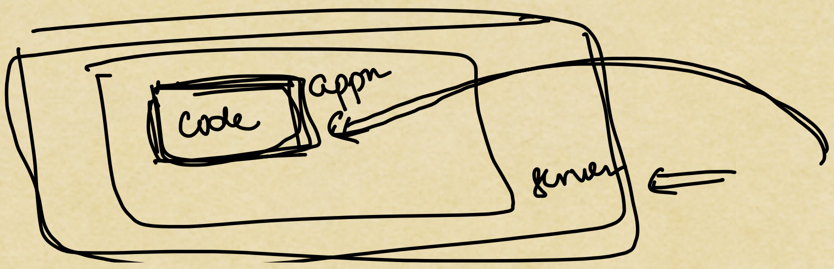
* Car





* 100 entries
↓
300 repo
service
controller

easily allows to create a BE server



⇒ <u>Requirement gathering</u> :-

i) User can book tickets for movie.

ii) User can select the seats for movie

iii) User can select their own location

iv) User can select their choice of theatre & shows.

v) Each theatre can have multiple [$\underset{=}{\text{audis}}$] $\to$ auditorium and can show multiple movies at the same time.

vi) User can do payment and recieve tickets for their respective booking

vii) Diff. types of seats can be present ⇒ PLATINUM, GOLD, SILVER

viii) User's can see all movies running in their region and for a particular user can see all possible shows in all possible theatres.

ix) Auditorium's can have multiple features like IMAX, 3D, 2D, DOLBY etc.

x) Movies can have multiple features like IMAX, 3D, 2D, DOLBY etc.

αi) we will only support online payments
(UPI, CARD)

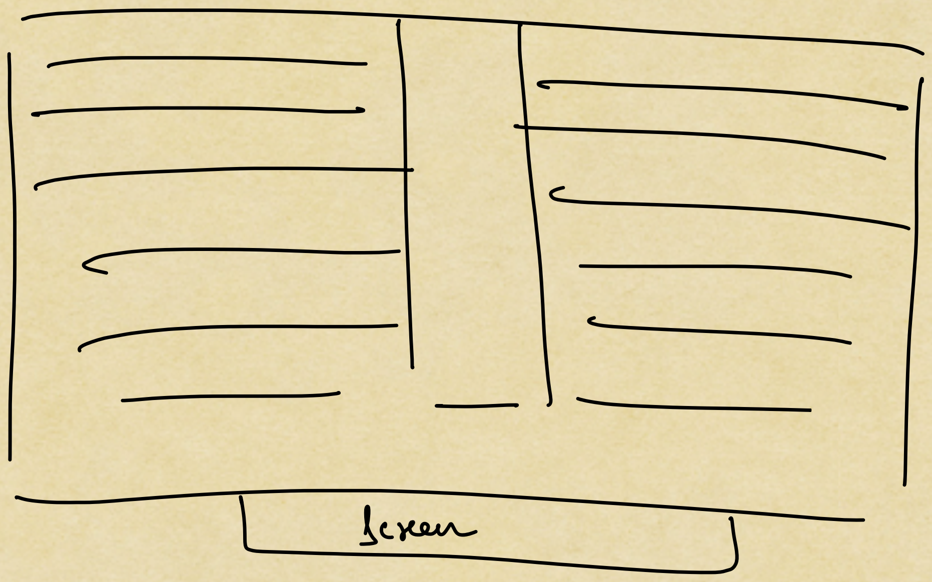
αii) we can have partial payments
or
multi-transaction payment

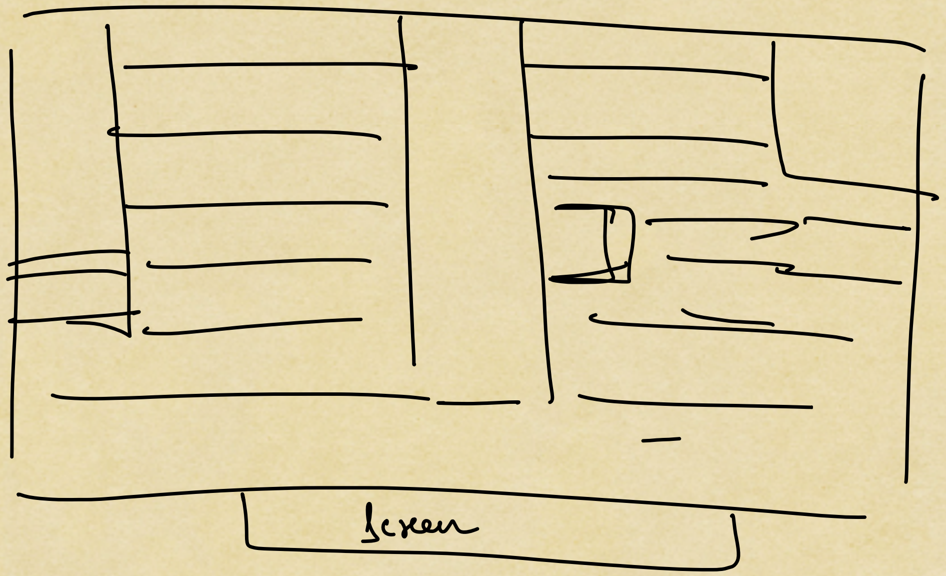
$$1000 \nearrow \underbrace{\overset{\text{Wallet}}{500} + \overset{\text{UPI}}{500}}_{\text{2 transactions}} \underbrace{\longrightarrow 1000}_{\text{single payment}}$$

αiii) we can have multiple transactions under 1 payment

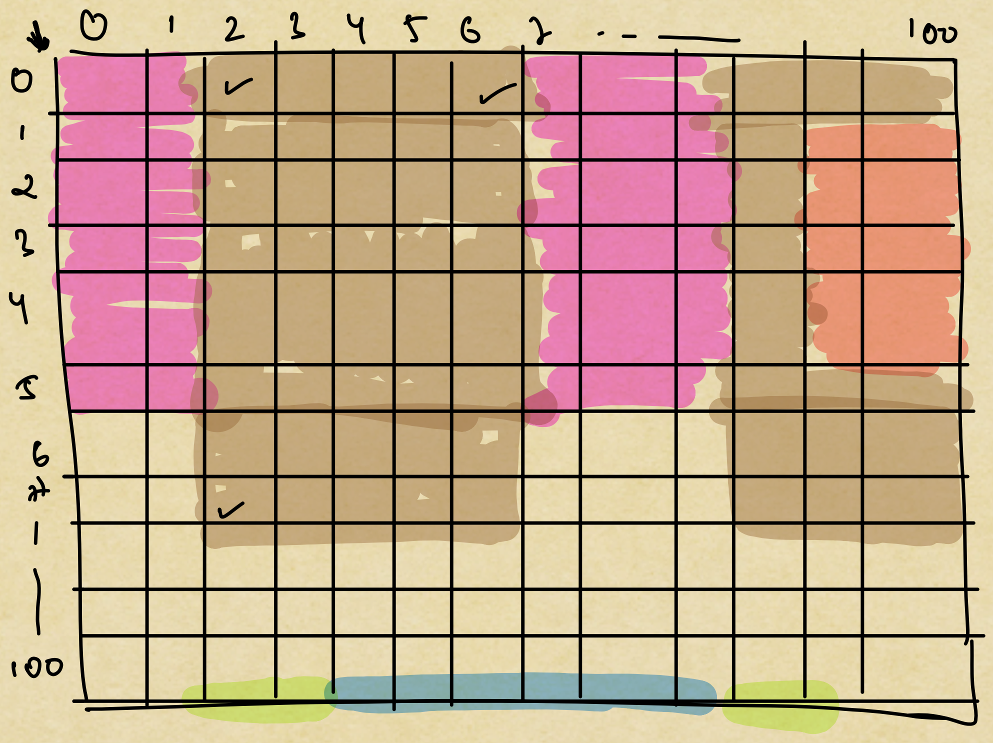
αiv) for movies, we will store certain details,
- name
- description
- actors
- release date
- poster → link to image
- trailer → link to video
- ratings

⇒ <u>Seating arrangement</u>



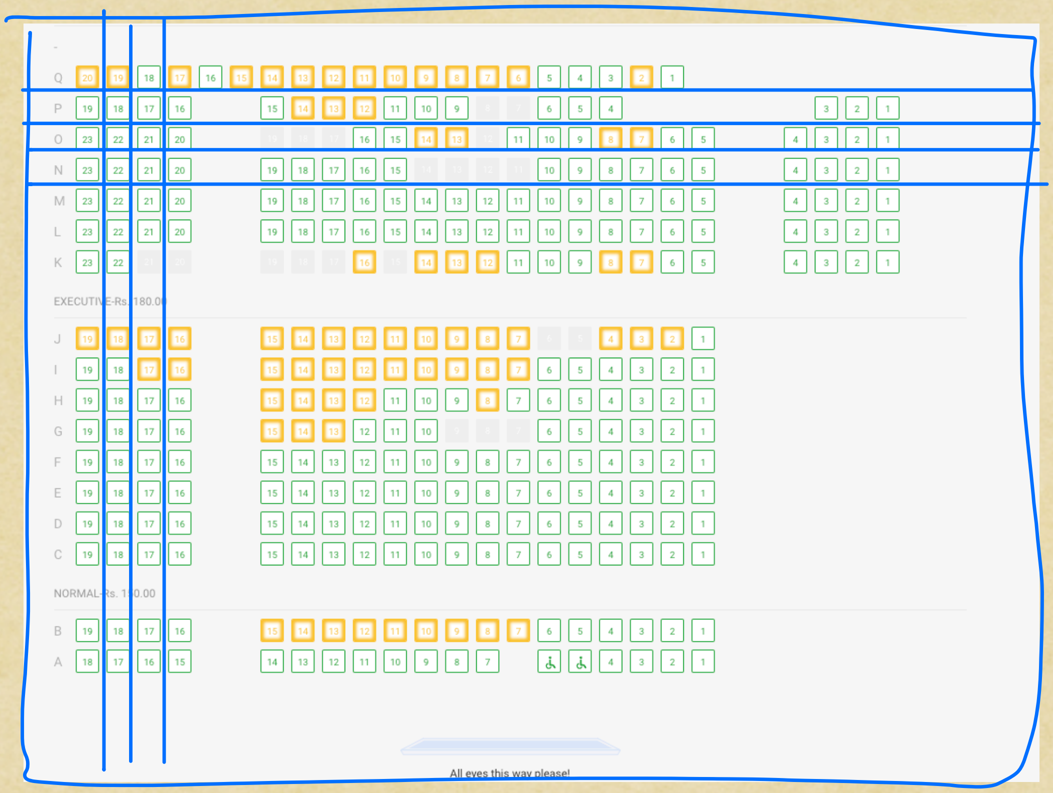


theatre → auditorium



Seat [00 to 06, 02 to 72]

FF ⇒ 100 X 100



Q	20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
P	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
O	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
N	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
M	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
L	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
K	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

EXECUTIVE-Rs. 180.00

J	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
I	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
H	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
G	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
F	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
E	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
D	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
C	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

NORMAL-Rs. 150.00

B	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
A	18 17 16 15 14 13 12 11 10 9 8 7 ♿ ♿ 4 3 2 1

All eyes this way please!

⇒ <u>Strings</u>

```
String s = "cat";
     ↓
   s = "dog";

 s = s + "rat"  ⇒ "catrat"
```

```
String s = "ball";
   s = s + "bat";

 print(s)  ⇒  "ball"      ①
               "ballbat"  ②
                 error    ③
```

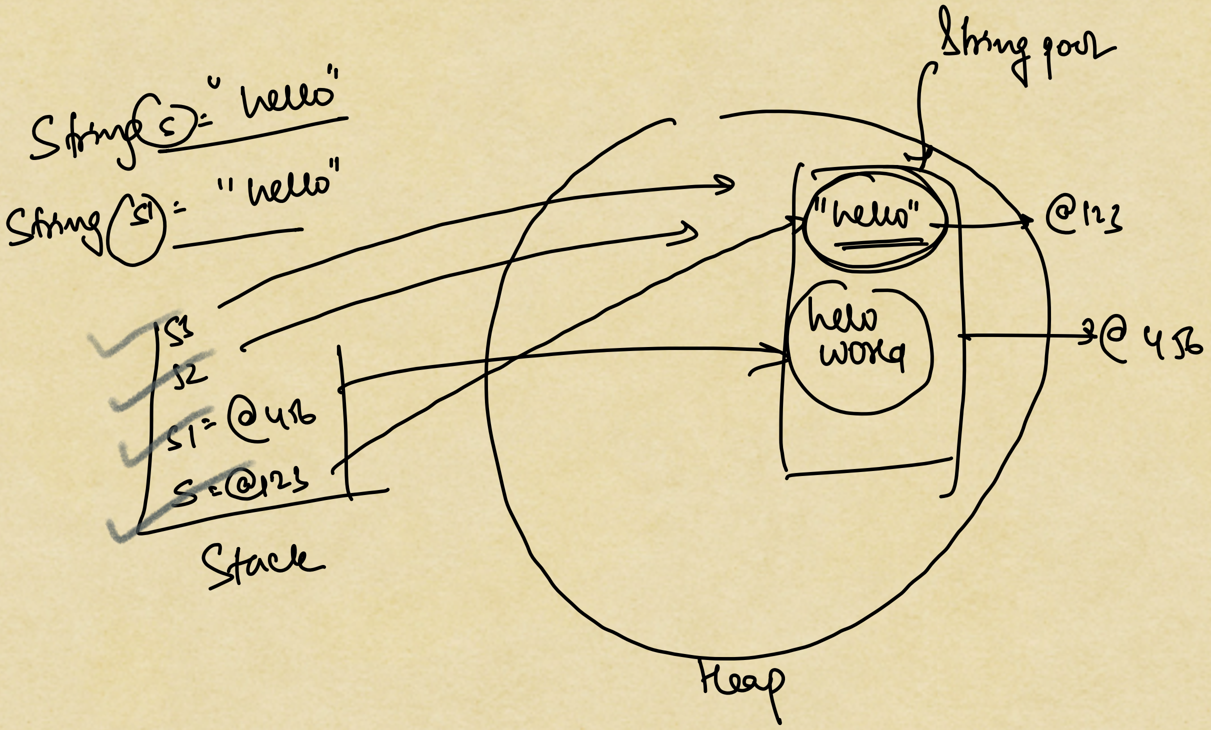
" ⟵———⟶

⟵———⟶

⟵———⟶ "

Common in strings
↓
<u>duplicacy</u>

String pool
↓
memory inside heap



s1 + world = "hello world"

print(s1) → hello world

print(s) → hello.

(s2)

(s3)