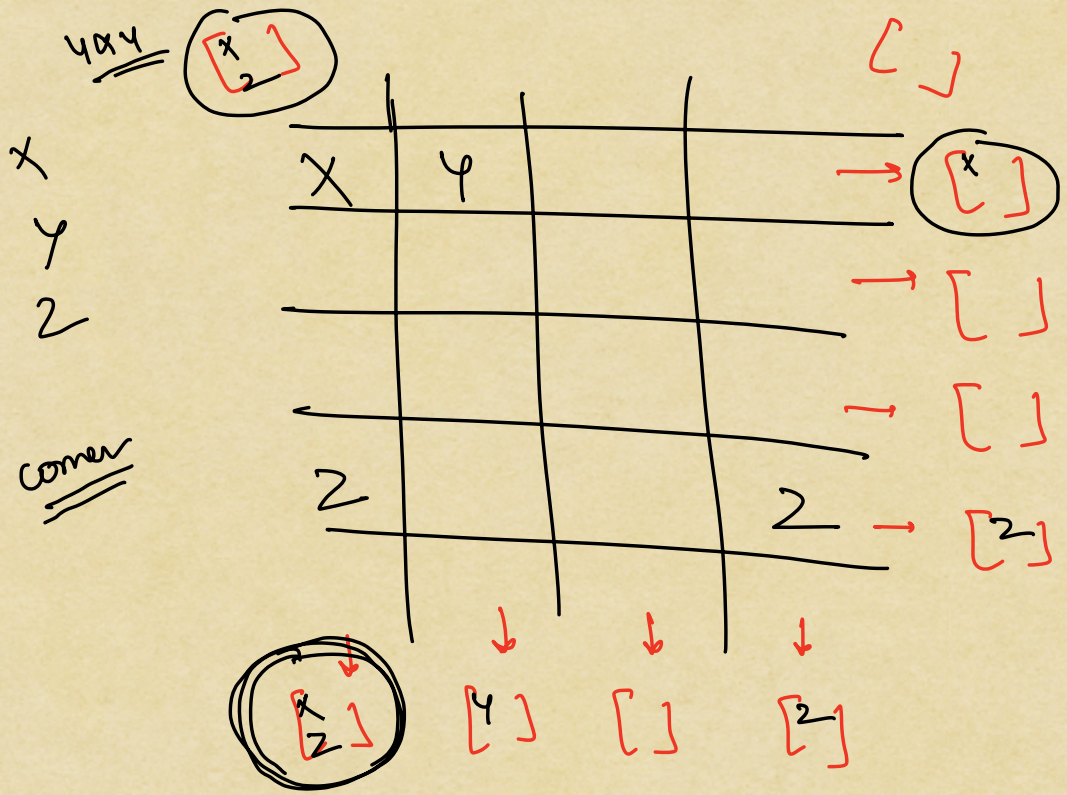


hashmap ⇒ no. of keys = 2 ← Can anyone become a winner

↓
eliminate the hashmap ?

↓
0 ⇒ draw

Set<Hashmap> ← Winning Strategy

↓ if(set.size=0) draw

↓ update hashmap ⇒ no. of keys

↓ >1

⇒ Parking lot — (S/w system to maintain parking lot)

VVV Popular Interview Question

Overview
- Know
- Don't know

⇒ entities only ? → ✗

⇒ design the entire working system ? → ✓

⇒ Interact with system ? → Command line

⇒ persistent ⇒ Yes → [in-memory]

⇒ REQUIREMENT GATHERING :-

⇒ Multiple floors

⇒ There is an entry at parking lot

⇒ there can be multiple entry gates but 1 entry gate per floor

⇒ A ticket is given at gate.

⇒ Ticket contains information — vehicle no, date time of entry, vehicle type, parking spot number

floor + spot

⇒ Different vehicle types will have different parking spots.

⇒ Entry gate will have an attendant

⇒ Parking spots might have some properties depending on their type :-

eg ⇒ EV vehicle type will have charger

Luxury vehicle type will have additional security

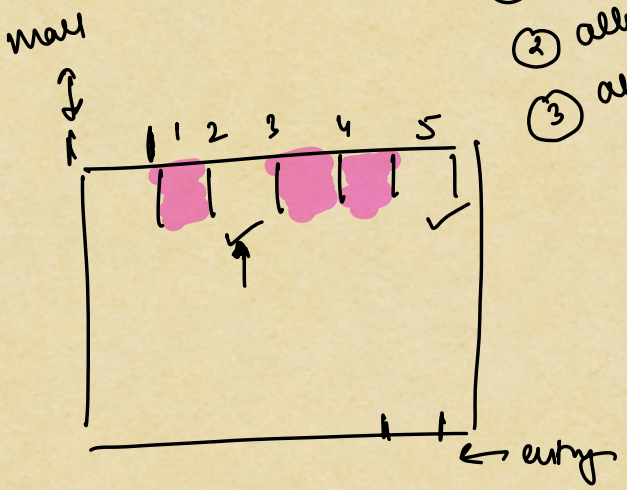
⇒ Spot allocation strategy [Strategy Design Pattern]

⇒ how to allocate a spot

to car spot available

+ car

① allocate randomly
② allocate serially
③ allocate closest to gate



⇒ Exit gate present for vehicles to exit.

⇒ there can be multiple Exit gates but 1 Exit gate per floor

⇒ attendant present at exit gate

⇒ ticket is given and bill is generated

⇒ payment for bill can be via multiple modes

- → Cash
- → UPI
- → Card
- → fastag

⇒ bill generation ⇒ ⇒ [Strategy design pattern]

bill should be generated based on time and vehicle type

↓

algorithm ⇒ baseRate

↓

hourlyRate * no. of hours

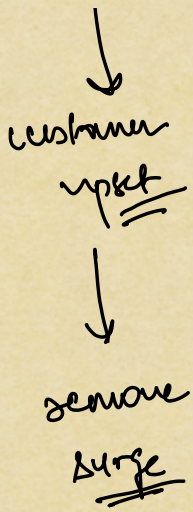
	Base Rate	Hourly Rate
Car	50	30
Bike	20	10

↓

Surge multiplier ⇒ if(filledCap > 50) {

0.5 ⇒ x ⇒ x * 0.5x

⇒ 1.5x



if (price >x) {

0.x ⇒ x ⇒ x + 0.xx
⇒ 1.xx

>90

0.9 ⇒ x + 0.9x
⇒ 1.9x

⇒ class diagram = ?!