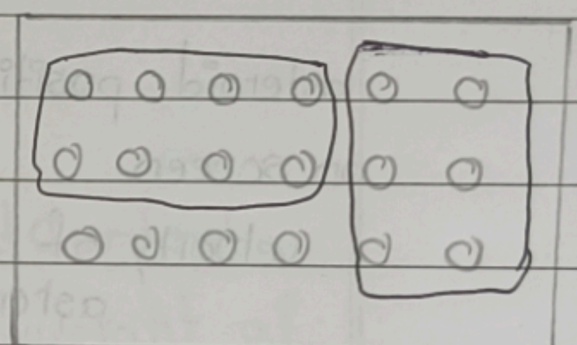
Lecture 30 : Flyweight Design Pattern

Introduction

- Let's assume hum bahoot saare objects create kar rahe for some random games or usecase and itne saare objects lete hai bahoot jyada RAM; and RAM is limited. And RAM will not be able to handle all objects.

- Aur Flyweight Design Pattern comes to rescue.
  ↳ and flyweight pattern bolta hai reuse karo already made objects.

- Flyweight pattern that is used to minimize memory usage by sharing common parts of state between multiple objects instead of storing all data in each object.



Game Example

Description : You are developing a game. You have space-ship which shoots laser and many asteriods are also coming and you have to shoot that asteroids.

- No. of asteriods depend upon level of difficulty.
If easy : 3-4 asteroids
if hard : much more asteroids.

- Ab yeh saare asteroids screen par ayenge and if ram is not optimize our game will crash.

Solution: To start reusing a single object instead of creating new ones each time.

- If we split object into two parts - intrinsic (shared) and extrinsic (unique) properties to reduce memory usage.

properties	Asteroid
All have limited values	int length; int width; int weight; string colour; string texture;
asteroid position on screen	→ int posx, posy;
velocity of asteroid	→ int velx, vely;

- Let's assume asteroid properties value range as -
  length - 10, 20, 30
  width - 10, 20, 30
  weight - 1, 2, 3
  color - Red, Green, Blue
  texture - soft, Hard, mix

  } using these combination properties are made

- no need to recreate them instead reuse them by just making diff objects

- Now, we are generating asteroid at any place & velocity and only this part is varying in all objects.

- And as we have fixed properties and fix options of range so these properties will be same for multiple objects.
- Dividing these properties in two parts

1. Intrinsic Property : Same property ; which can be reused . In our game its - length, width, weight, color, theme.

2. Extrinsic Property : not same for any asteroids :. can't be reused . In our game its - velocity, position

- Seperating these properties -

Asteriod Flyweight
int length;
int width;
int weight;
string colour;
string texture;

'has-a'

↳ Reusable

Asteroid Context
AsteroidFlyweight af;
int posx;
int posy;
int velx;
int vely;

'has-a'

Flyweight Factory
map <string, AsteroidFly weight> pool
Asteroid Flyweight (l, w, wt, col, texture)

→ This checks in map if that object exists if not then create it & if yes then give it

- We write logic in this way -

[10, 20, 30] len
[10, 20, 30] width
[1, 2, 3] weight
[R, G, B] colour
[H, S, SH] texture

3 objects
Total combination if made new 10L ⟹ [3L, 3L, ~~3~~L]

⊙ So, we will optimize ~~10~~L → 3L in flyweight object
- Now, how we will determine if we have created these 3 objects

Map<key, AsteroidFlyweight> pool;
↓ unique value
string → [length + '|' + width + '|' + weight + '|' + 'colour' + '|' + texture]

key = "10|10|1|Red|Hard"

- Now we can store this key as string and all corresponding asteroid this key.

Map →		
10|10|1|Red|Hard		A1
20|20|2|4|Green|Soft		A2
30|30|3|Blue|SH		A3

- objects
the valu
objects.

standard

Extrinsic
Flyweight

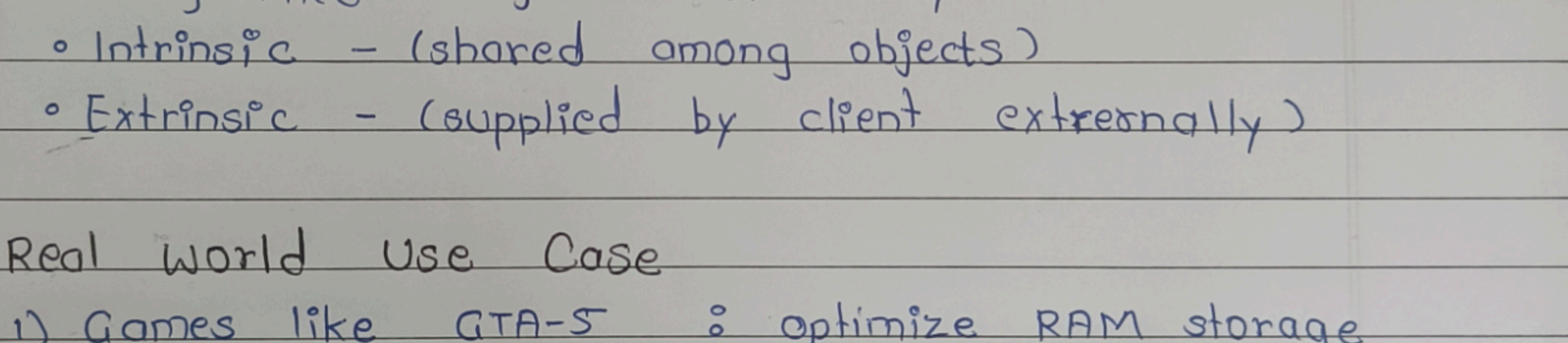
Extrins
pr

standard
Flyweigh
fine gr
◦ Intrin
◦ Extrin

Real W
1) Game
2) Usec

- Objects must be immutable. ⟹ If we change any of the value then change is done in all 3L or 4L objects.

standard UML



standard Definition

Flyweight uses sharing to support large number of fine grained objects effectively.

- Intrinsic - (shared among objects)
- Extrinsic - (supplied by client externally)

Real world Use Case

1) Games like GTA-5 : optimize RAM storage
2) Used in text editors
   ↳ to provide properties to characters.