Flyweight Design Pattern.

=> observe & understand: when to use this pattern:

-> when Memory is Limited
-> when objects Shared Data
① Intrinsic data: Shared Amoung objects & remain same once defined one Value.

② Entrinsic data: changes based on client input & differs from one object to another

-> Creation of object is Expensive

=> How to Solve the issue?

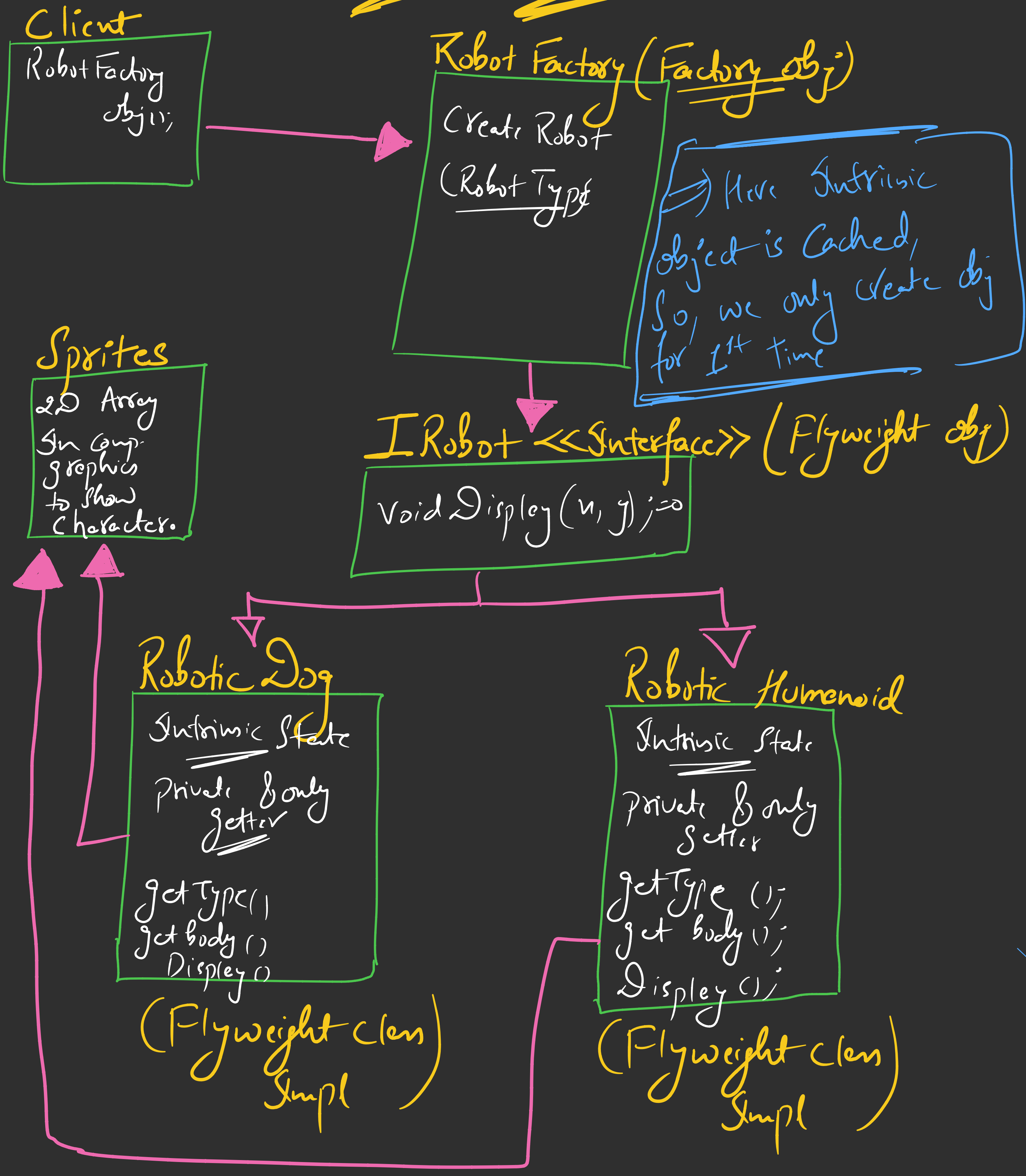
-> From object, remove All Entrinsic data & Keep Intrinsic data [This obj. Called Flyweight object]

-> Flyweight Class Can be Immutable

-> Entrinsic Data Can be passed to Flyweight class in method parameter

-> once Flyweight object is Created, It is Cached & reused wherever required.

UML Diagram


Client
RobotFactory obj();
Robot Factory (Factory obj)
Create Robot (Robot Type)
→ Here Intrinsic object is Cached, So, we only create obj for 1st time
Sprites
2D Array In Comp. graphics to show character.
IRobot <<Interface>> (Flyweight obj)
Void Display(x, y);=0
Robotic Dog
Intrinsic State
Private & only getter
getType()
get body()
Display()
(Flyweight class Impl)
Robotic Humanoid
Intrinsic State
Private & only Setter
getType();
get body();
Display();
(Flyweight class Impl)