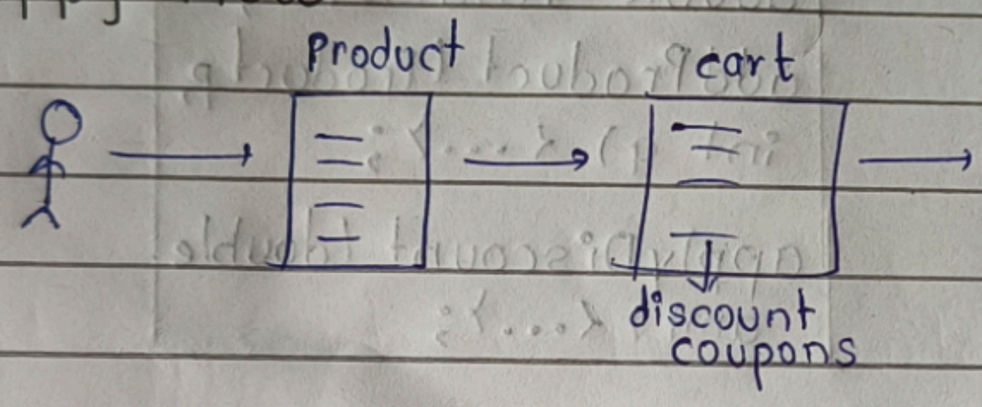
Lecture 24 : Build Discount Coupon system

Page No.
Date

Functional Requirements

1. We can add new coupons at runtime
2. Both cart level and product level discounts
3. Support multiple coupon types (seasonal after, loyalty discount, Banking discount, etc)
4. Support both flat and percentage discount
5. One coupon can/cannot be applied on top of other coupons.
6.

Happy Flow



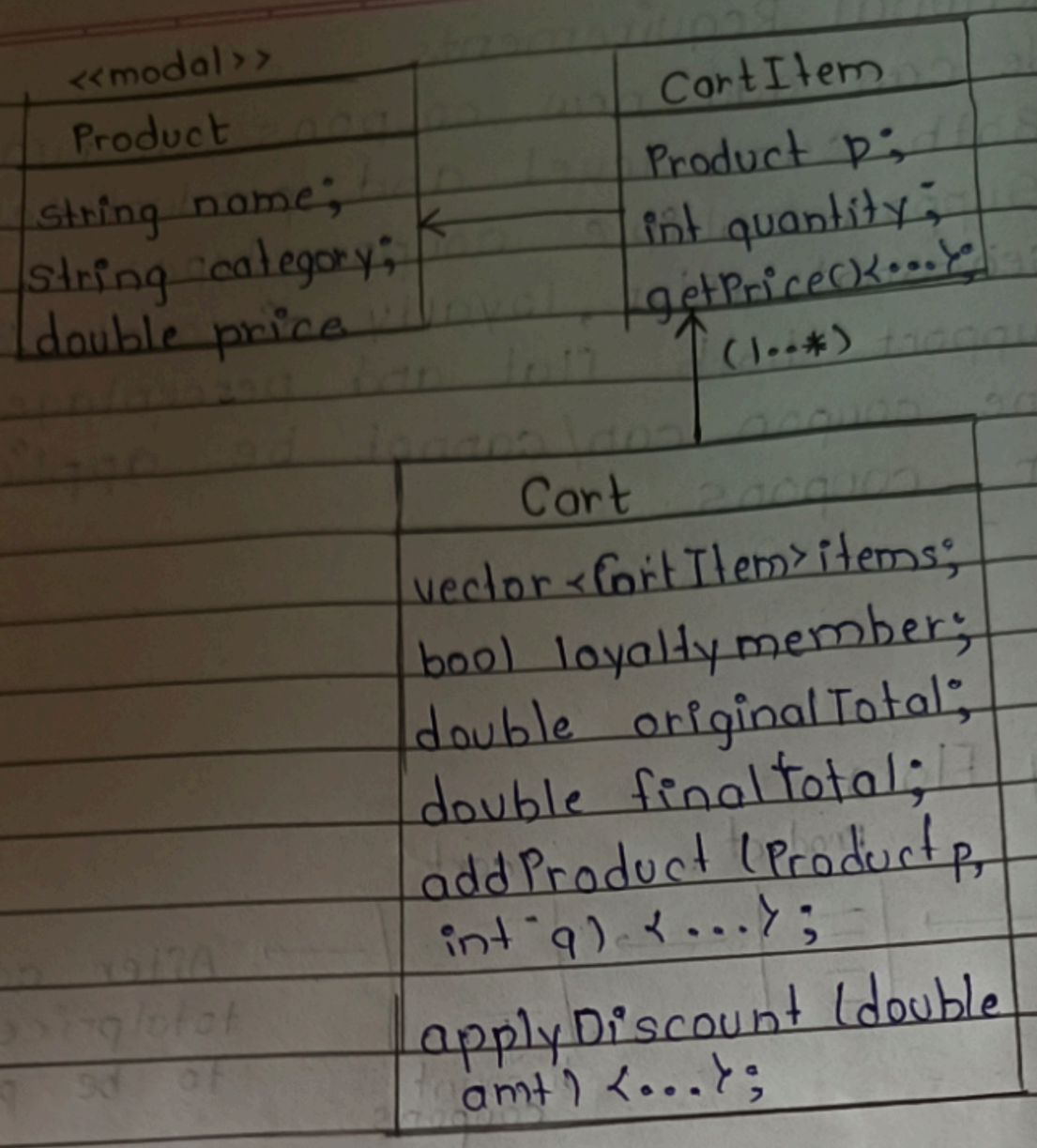
- Plug and Play model
- No need to make user class as it will handled by application which will integrate them

UML Diagram

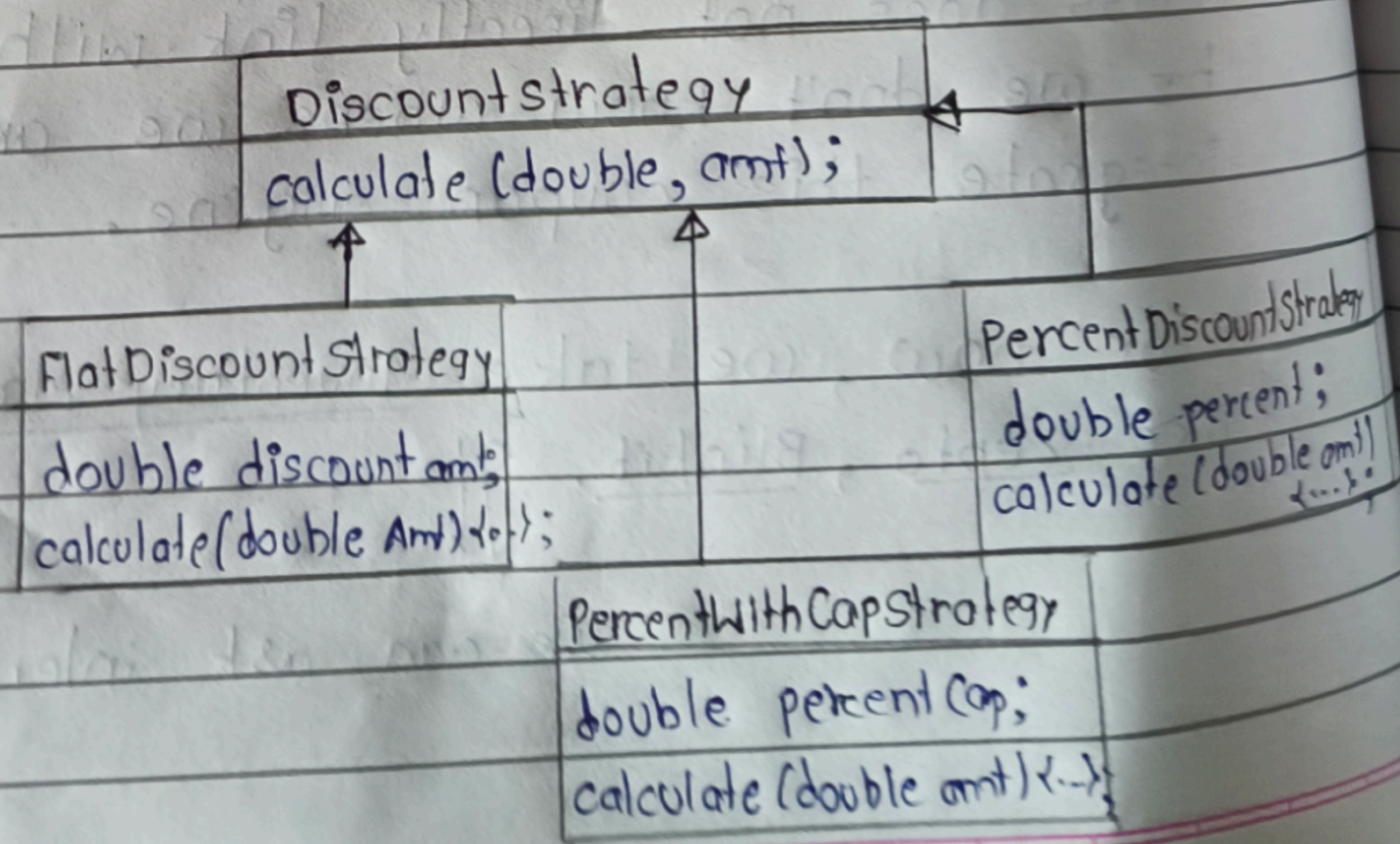
- We will create generic product class as this class does not directly link with this application bcz we don't know where we are going to integrate this coupon engine.

- For now, we take Inventory Management application like zepto, Blinkit.

We can ask interviewer too.

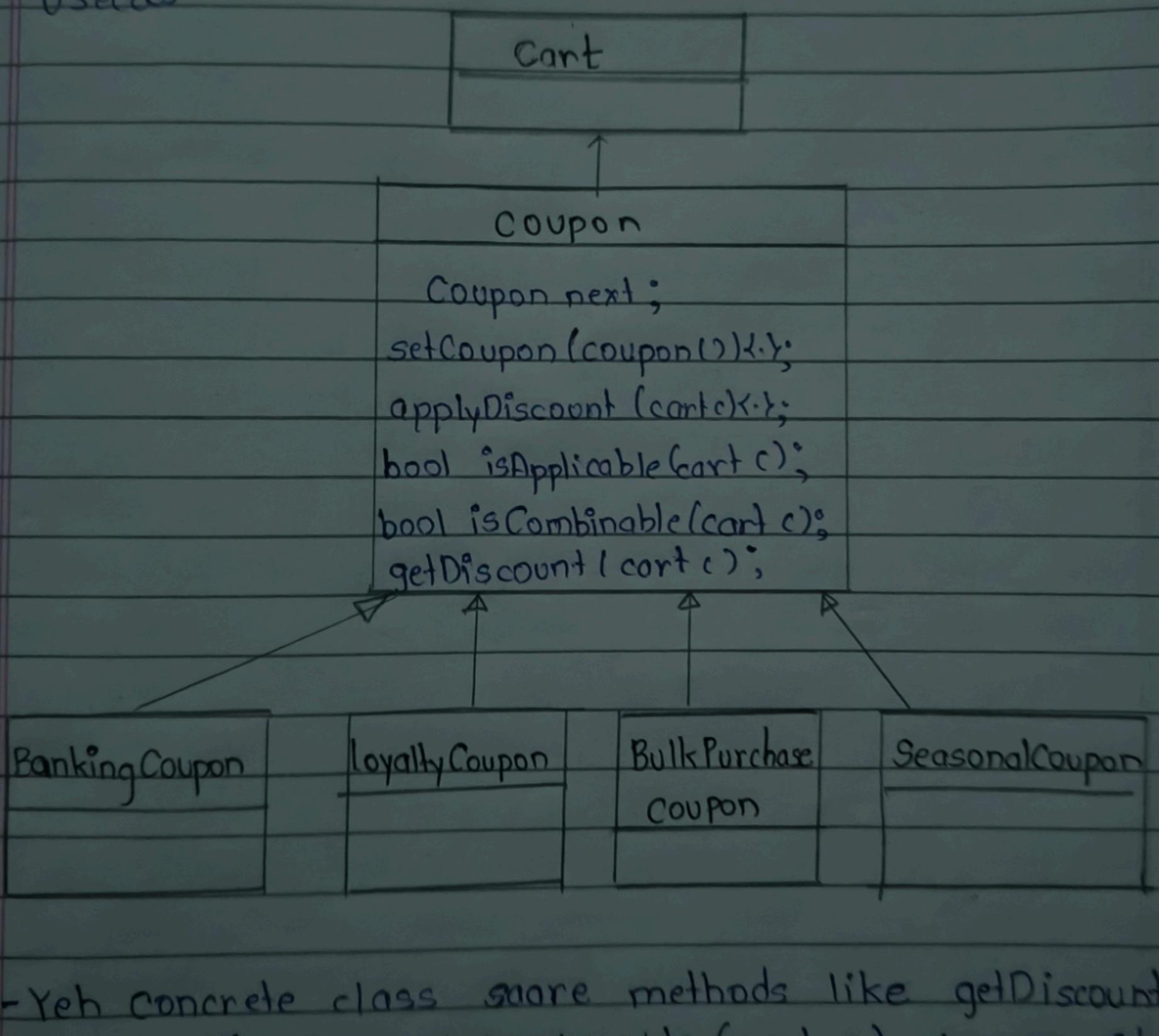


⊙ Now, hum Strategy Design Pattern ~~here~~ ~~to~~ use karege to create different types of discounts jaise flat/%. Client can use alternatively this strategy



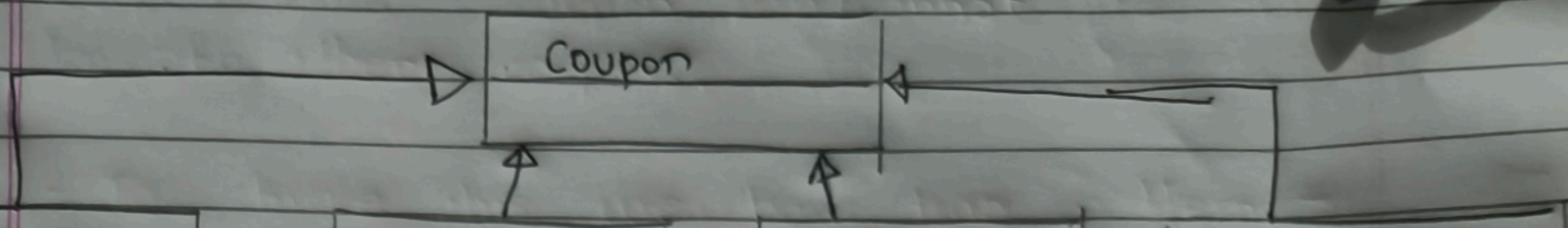
- Let's create our coupon class
Coupon class : coupon can/cannot be applied on top of other coupons.

- To achieve this we have two patterns -
  - Decorator - already ex dekha hai
  - Chain of Responsibility (COR)
    ↳ isliye hum ye choose karege to understand its usecase:



Yeh concrete class mere methods like getDiscount(cart c), isApplicable(cart c), isCombinable(cart c) ko override karega.

Now let's see Coupon Types.

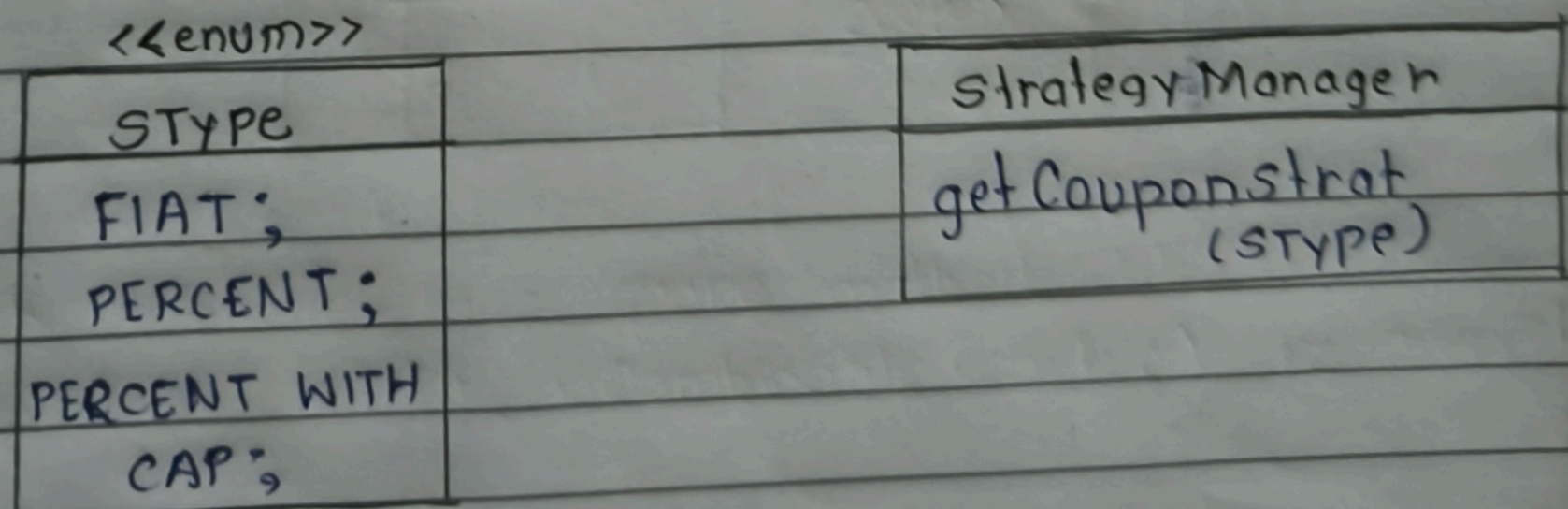


BankingCoupon	LoyaltyCoupon	BulkPurchaseCoupon	SeasonalCoupon
String bank; double minspent; percent, off; DiscountStrategy ds; isApplicable(cartc) {...} iscombinable(cartc) {...}	double percent; DiscountStrategy ds; isApplicable(cartc) {...} isCombinable(cartc) {...}	double threshold; double flatoff; DiscountStrategy ds; isApplicable(cartc) {...}; isCombinable (cartc) {...};	String Category double percent; DiscountStrategy ds; isApplicable(cartc) {...}; isCombinable(cartc) {...};

→ Ab hum different variables in different coupon so that hum usse different strategy mein use kar skte hai.

→ Coupons (Conke concrete class) 'has-a' relation with Discountstrategy.

→ We make Strategy Manager (Factory Pattern) which talk with different strategies.



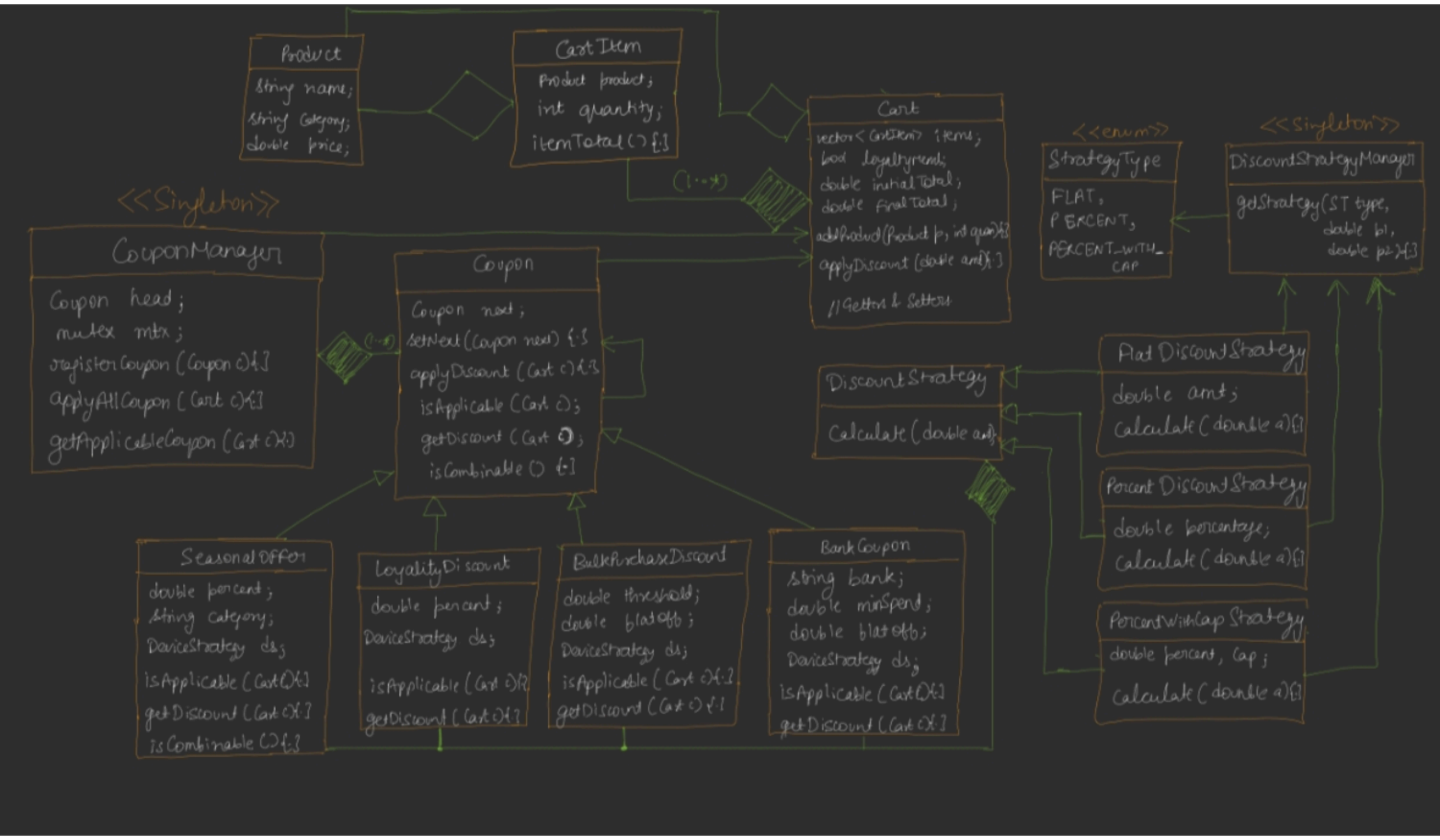
→ Only one class left → which will interact with this whole system.
↳ First point of contact for coupons;

Page No.
Date

Coupon Manager
coupon * head;
Mutex mtx;
registerCoupon(coupon());
ApplyAllCoupon (head) {...};
getAllCou...nt c){...};

thread
sale

(1..*) → Coupon


Product
String name;
String Category;
double price;
CartItem
Product product;
int quantity;
itemTotal() {}
(1...*)
Cart
vector<CartItem> items;
bool loyaltyFriend;
double initialTotal;
double finalTotal;
addProduct(Product p, int quan){}
applyDiscount(double amt){}
// Getters & Setters
<<enum>>
StrategyType
FLAT,
PERCENT,
PERCENT_WITH_CAP
<<Singleton>>
DiscountStrategyManager
getStrategy(ST type, double p1, double p2){}
<<Singleton>>
CouponManager
Coupon head;
mutex mtx;
registerCoupon(Coupon c){}
applyAllCoupon(Cart c){}
getApplicableCoupon(Cart c){}
(1...*)
Coupon
Coupon next;
setNext(Coupon next) {}
applyDiscount(Cart c){}
isApplicable(Cart c);
getDiscount(Cart c);
isCombinable() {}
DiscountStrategy
Calculate(double amt);
FlatDiscountStrategy
double amt;
Calculate(double a){}
PercentDiscountStrategy
double percentage;
Calculate(double a){}
PercentWithCapStrategy
double percent, cap;
Calculate(double a){}
SeasonalOffer
double percent;
string category;
DaviceStrategy ds;
isApplicable(Cart){}
getDiscount(Cart c){}
isCombinable(){}
LoyalityDiscount
double percent;
DaviceStrategy ds;
isApplicable(Cart c){}
getDiscount(Cart c){}
BulkPurchaseDiscount
double threshold;
double flatOff;
DaviceStrategy ds;
isApplicable(Cart c){}
getDiscount(Cart c){}
BankCoupon
string bank;
double minSpend;
double flatOff;
DaviceStrategy ds;
isApplicable(Cart){}
getDiscount(Cart c){}