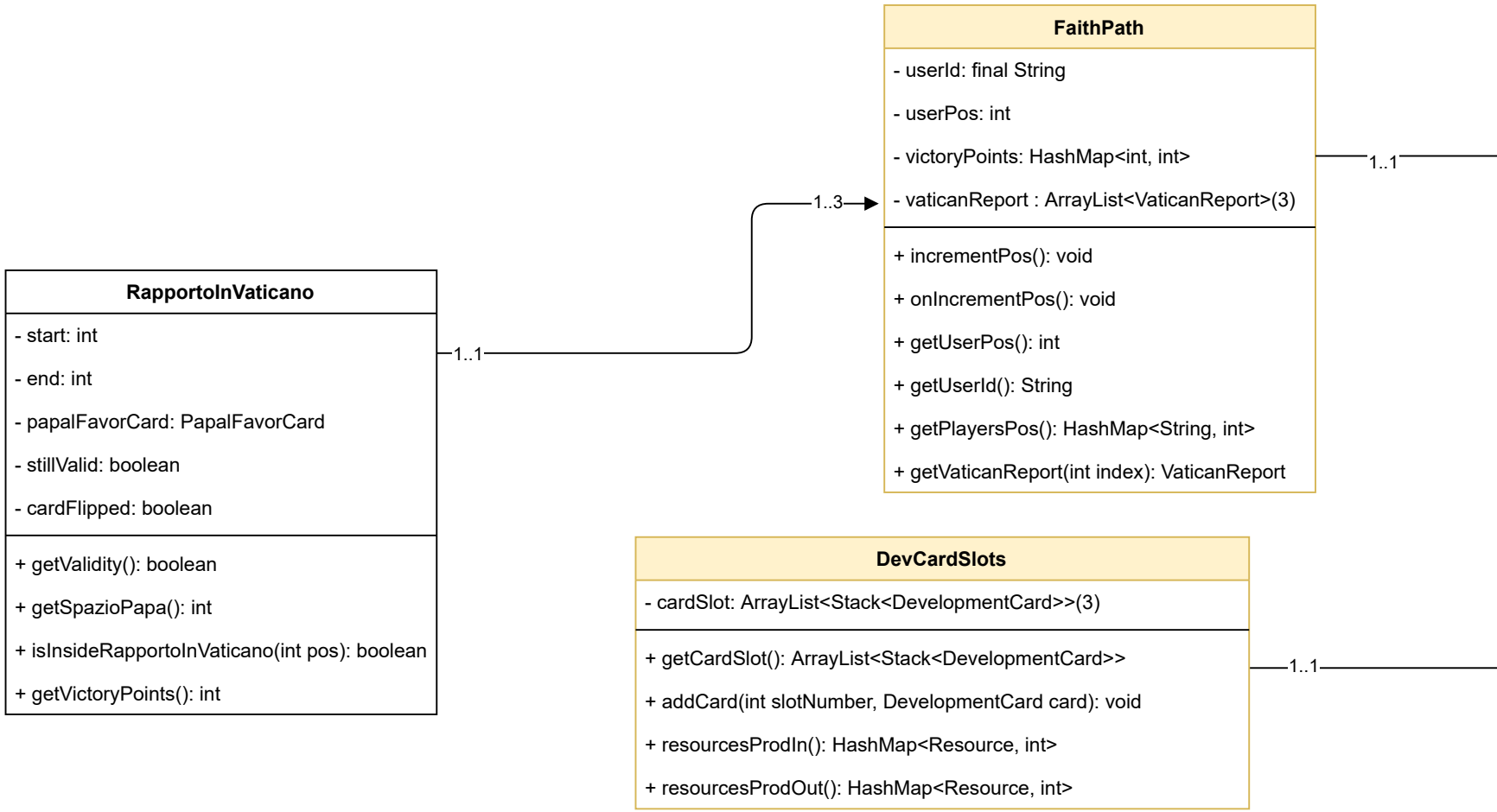
OnMarblePickUp viene usato dentro
il metodo ***insertMarble***



Implementazioni future:

- rendere i modificatori dei metodi nelle classi depositi privati.
- Factory method per l'inizializzazione della carte.
- Observer pattern nell' MVC.
- Inizializzazione del gioco da file.

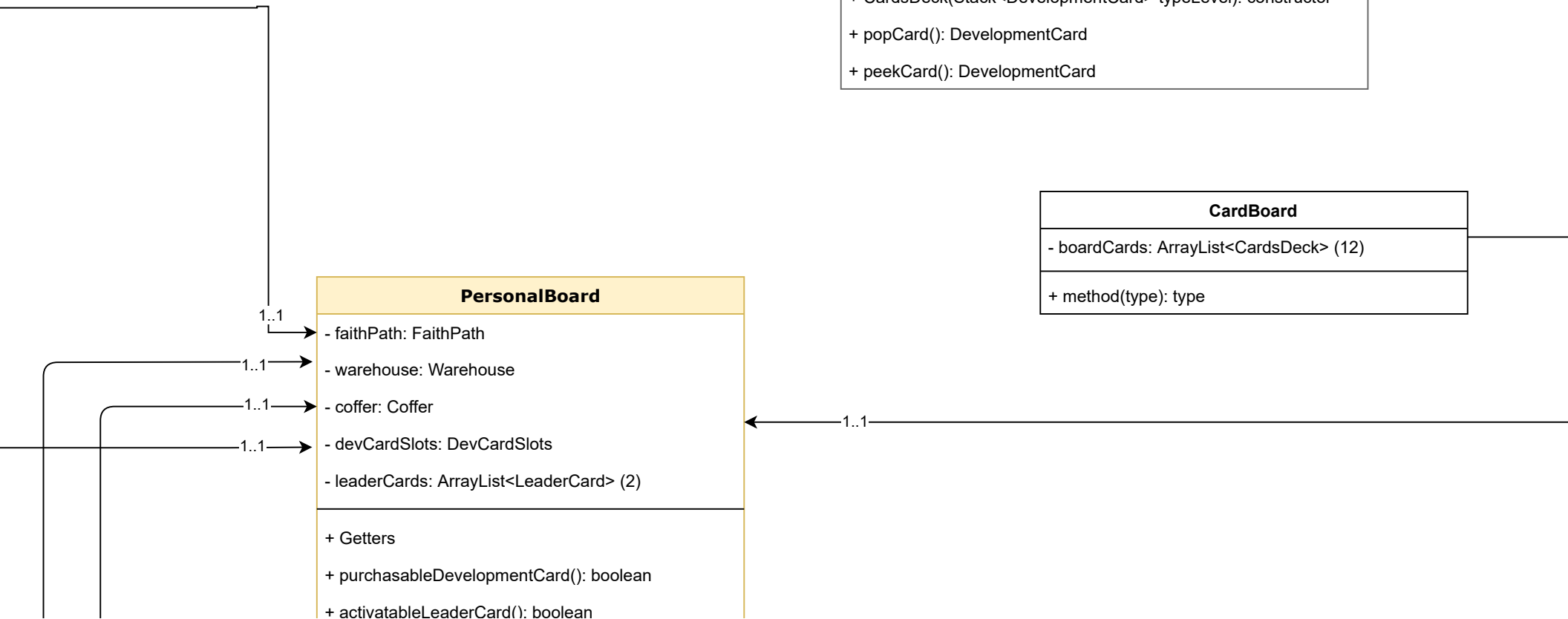
Domande:

- Struttura mercato e Plancia mercato saranno inglobate nello stesso oggetto, giusto?
(Adel)=in teoria si'
- La riserva delle risorse vogliamo renderla un oggetto?
(Adel)=converrebbe perche' ci sarranno metodi che aggiungono/tolgono risorse alla riserva.
- Come rappresentare il potere di produzione delle carte Produzione? ENUM?
- Aggiungiamo un attributo boolean nella classe Player per assegnare il calamaio?
(Adel)=ci sta
-

Abbreviazioni&osservazioni:

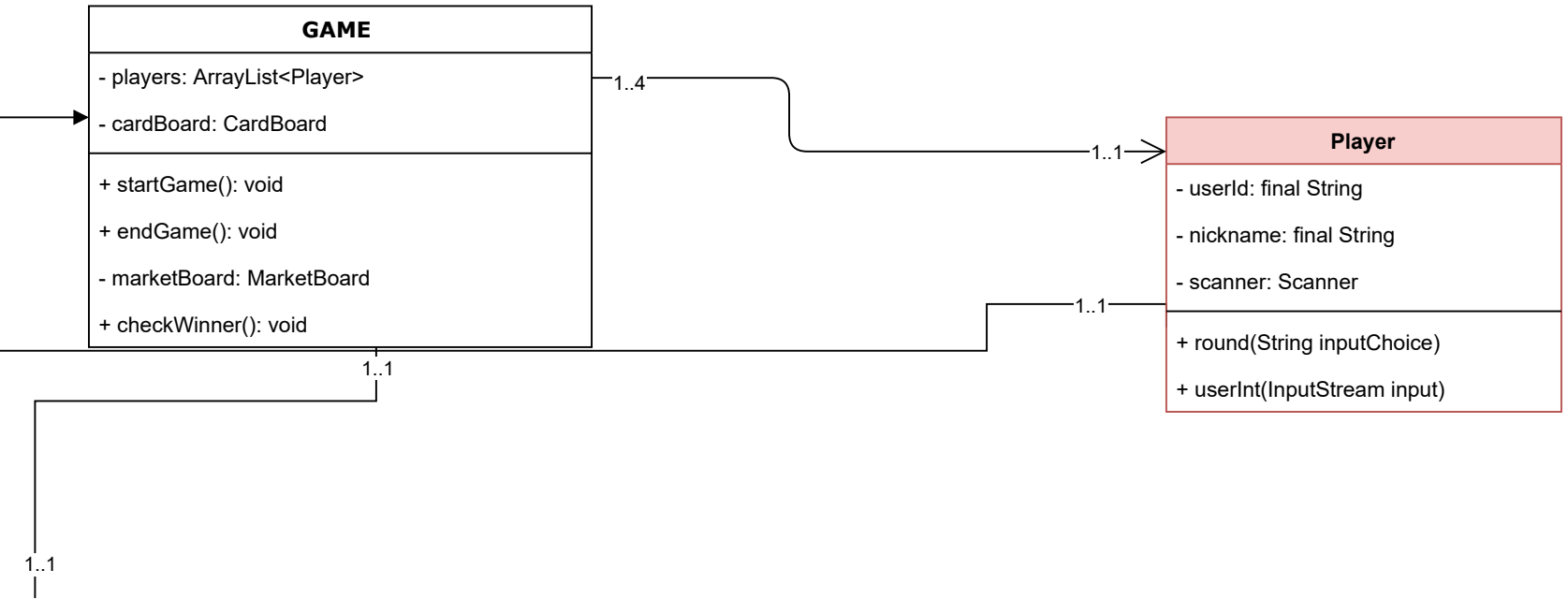
- Marble==Biglia
- DevCardSlots==Development Card Slots
-.

-Nel Warehouse:
moveResource((int)fromStorageNum,(int)toStorageNum):boolean
//{sposta risorsa tra gli scaffali all'interno del magazzino}-->Prende due interi, il numero di scaffale da cui spostare una risorsa e lo scaffale a cui spostare.



Domande al prof:

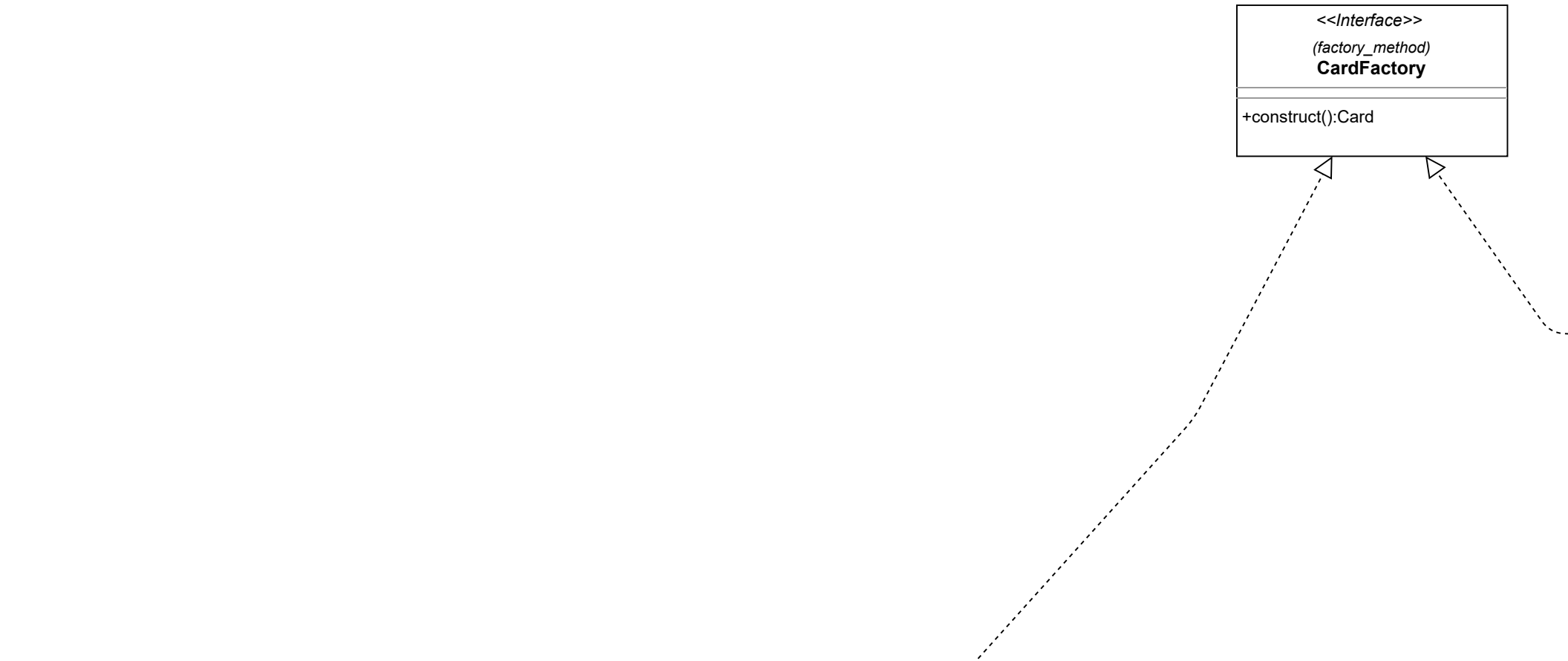
- Quando si pescano le carte leader, una volta che ne ho scelte due, le altre vanno inserite direttamente nel mazzo, che a sua volta viene rimescolato o tutti pescano 4 carte contemporaneamente e poi tutti ne scartano 2?
- quando iniziare a pensare a MVC

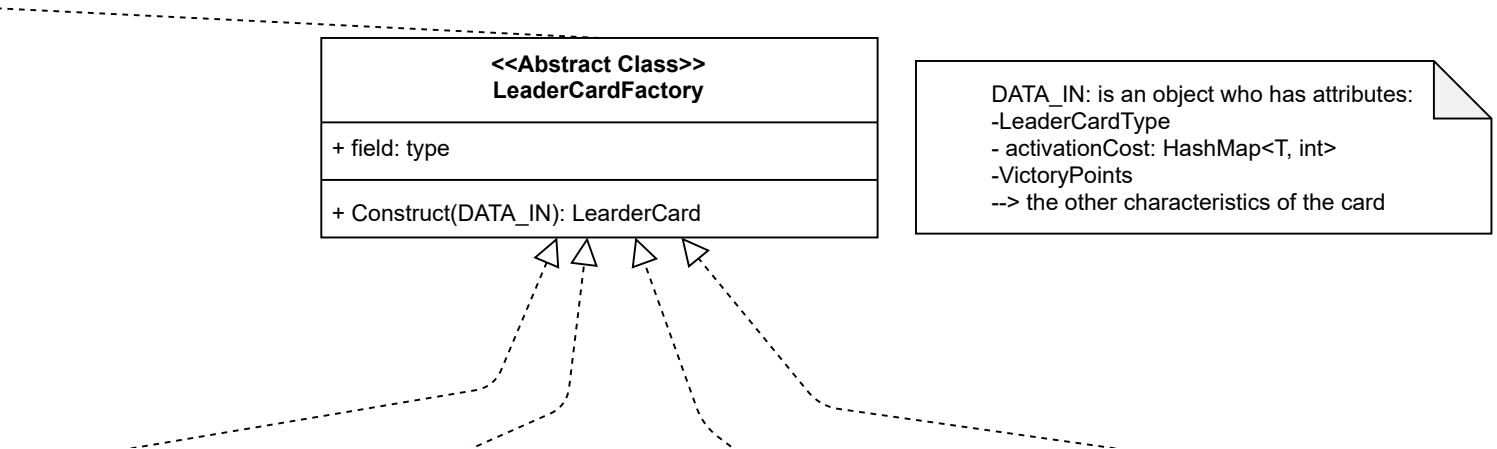


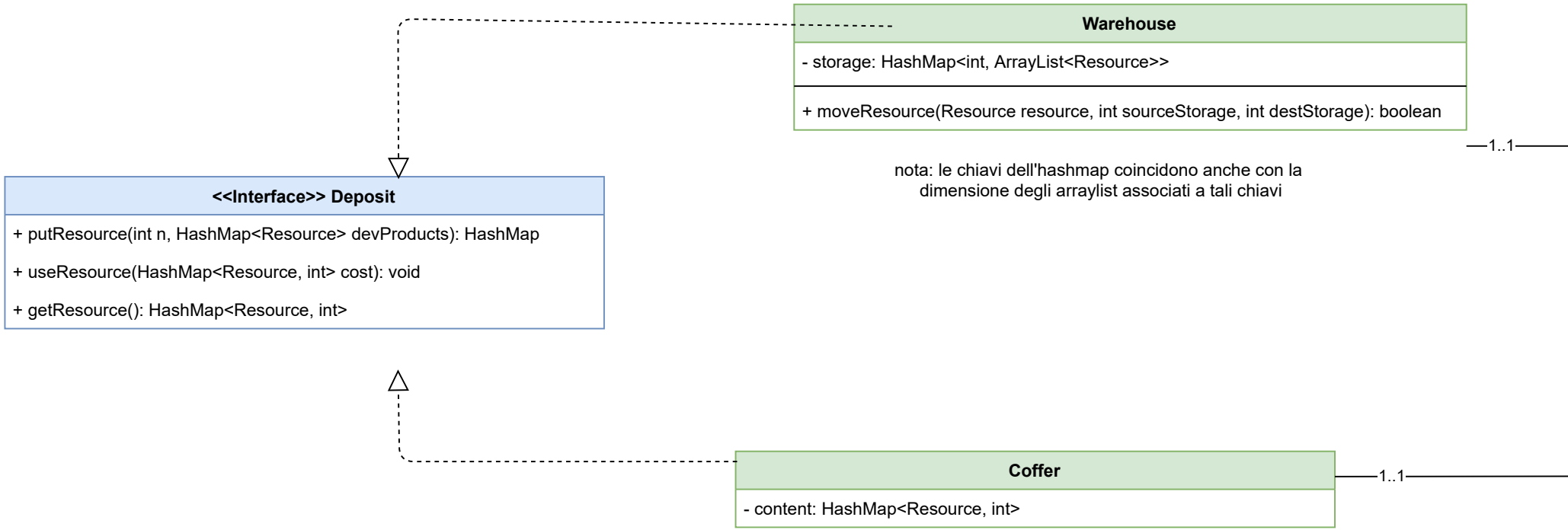
<<Interface>>
CardFactory

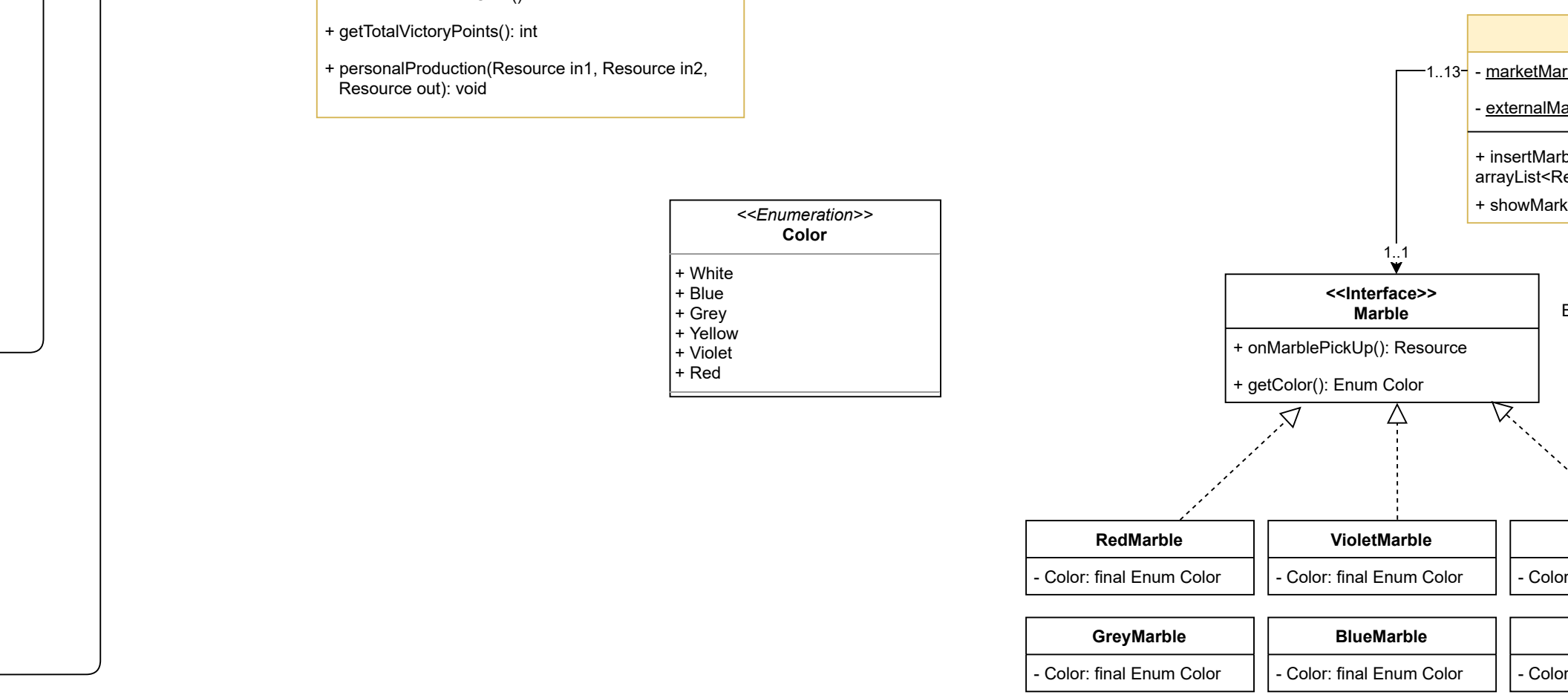
+createDevelopmentCard(int PV, HashMap <String, int> cardCost, HashMap <String, Int> prodIn, HashMap <String, Int> prodOut, String type, String level) : DevelopmentCard

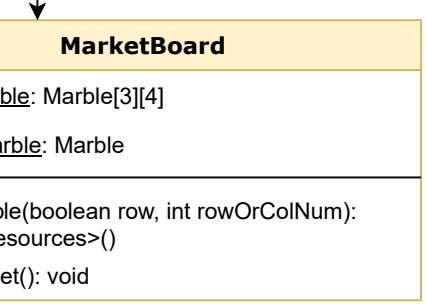
+createLeaderCard(String leaderCardType, int PV, HashMap <String, int> activationCost, String resourceType) : DevelopmentCard



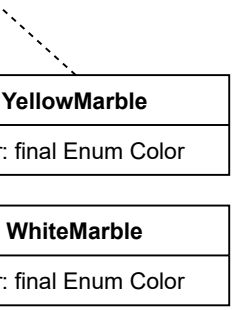

<<Abstract Class>>
LeaderCardFactory
+ field: type
+ Construct(DATA_IN): LearderCard
DATA_IN: is an object who has attributes:
-LeaderCardType
- activationCost: HashMap<T, int>
-VictoryPoints
--> the other characteristics of the card

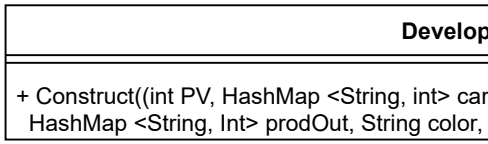

Warehouse
- storage: HashMap<int, ArrayList<Resource>>
+ moveResource(Resource resource, int sourceStorage, int destStorage): boolean
1..1
nota: le chiavi dell'hashmap coincidono anche con la dimensione degli arraylist associati a tali chiavi
<<Interface>> Deposit
+ putResource(int n, HashMap<Resource> devProducts): HashMap
+ useResource(HashMap<Resource, int> cost): void
+ getResource(): HashMap<Resource, int>
Coffer
- content: HashMap<Resource, int>
1..1

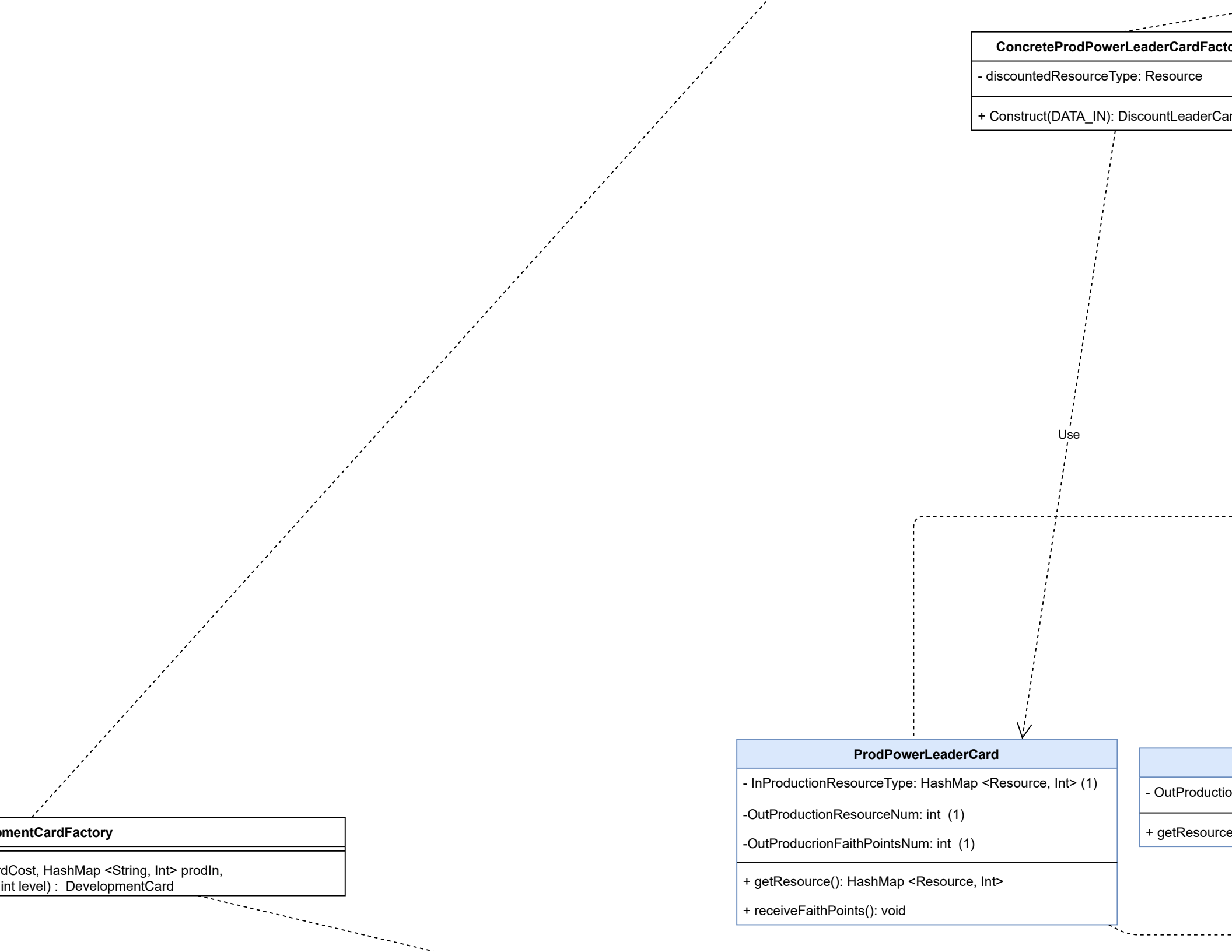

+ getTotalVictoryPoints(): int
+ personalProduction(Resource in1, Resource in2, Resource out): void
<<Enumeration>>
Color
+ White
+ Blue
+ Grey
+ Yellow
+ Violet
+ Red
1..13
- marketMar
- externalMa
+ insertMarb
arrayList<Re
+ showMark
1..1
<<Interface>>
Marble
+ onMarblePickUp(): Resource
+ getColor(): Enum Color
RedMarble
- Color: final Enum Color
VioletMarble
- Color: final Enum Color
- Color
GreyMarble
- Color: final Enum Color
BlueMarble
- Color: final Enum Color
- Color

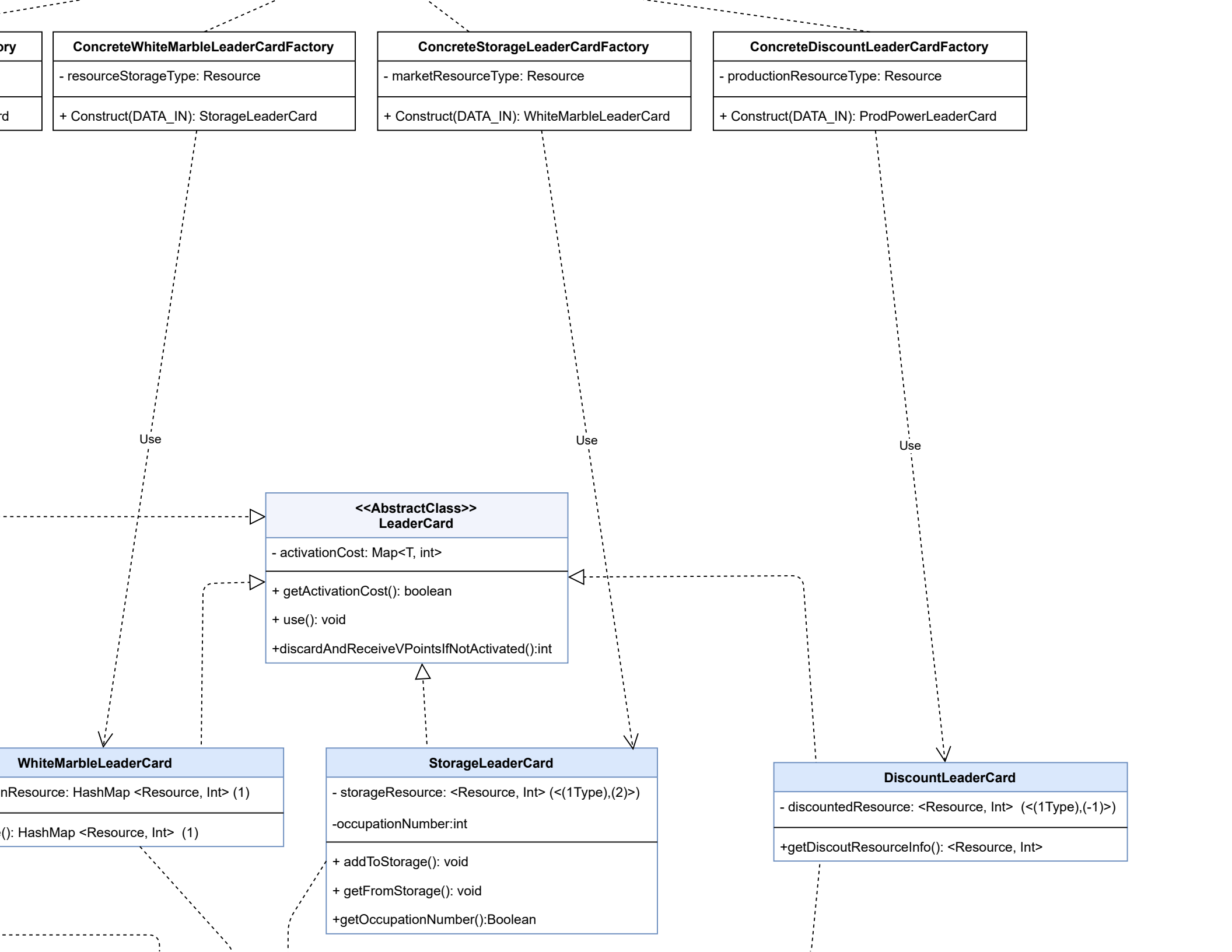


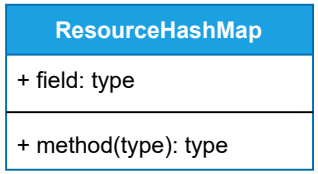
Biosgna aggiungere un attributo player alla funzione insertMarble per far in modo che si sappia chi è che sta aggiungendo aggiungendo la marble.



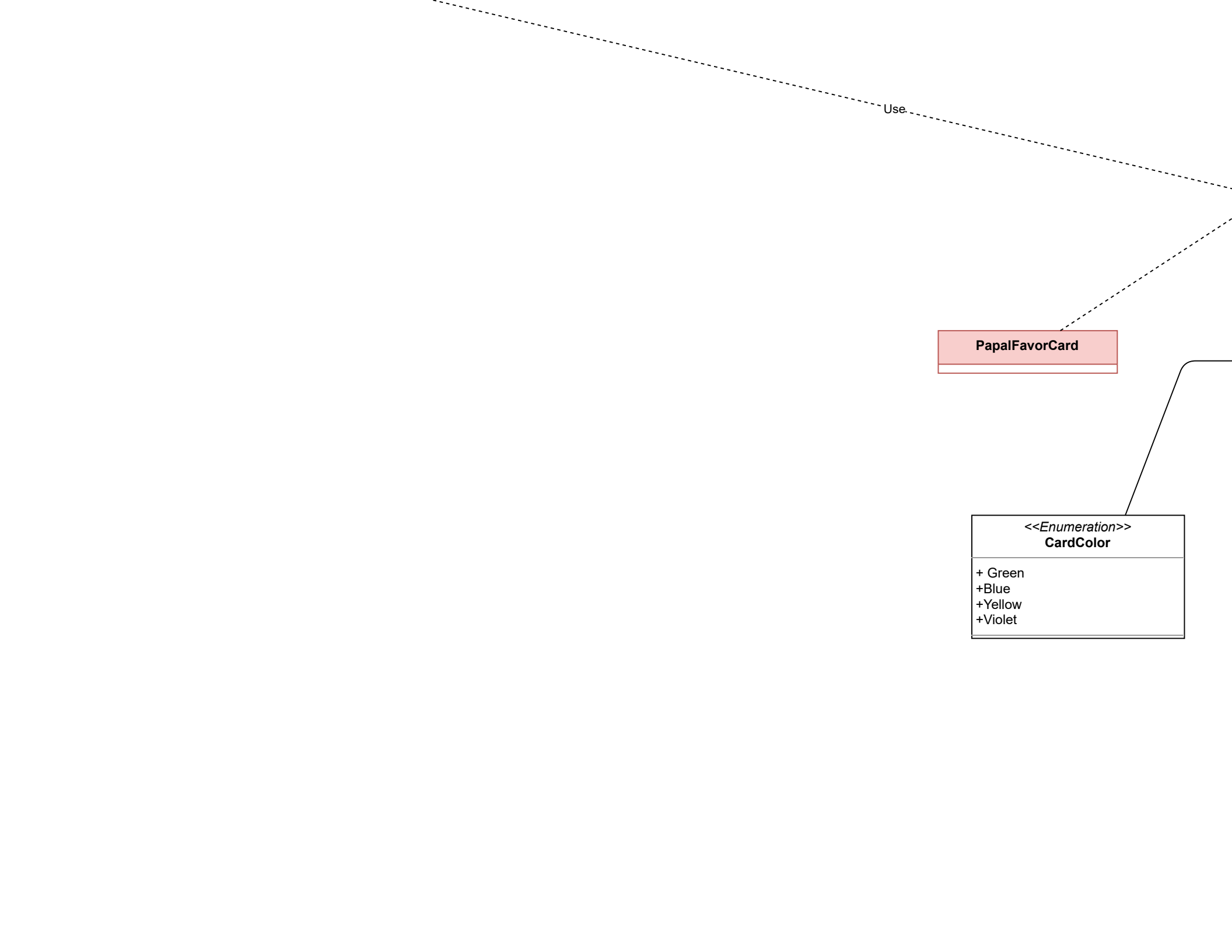


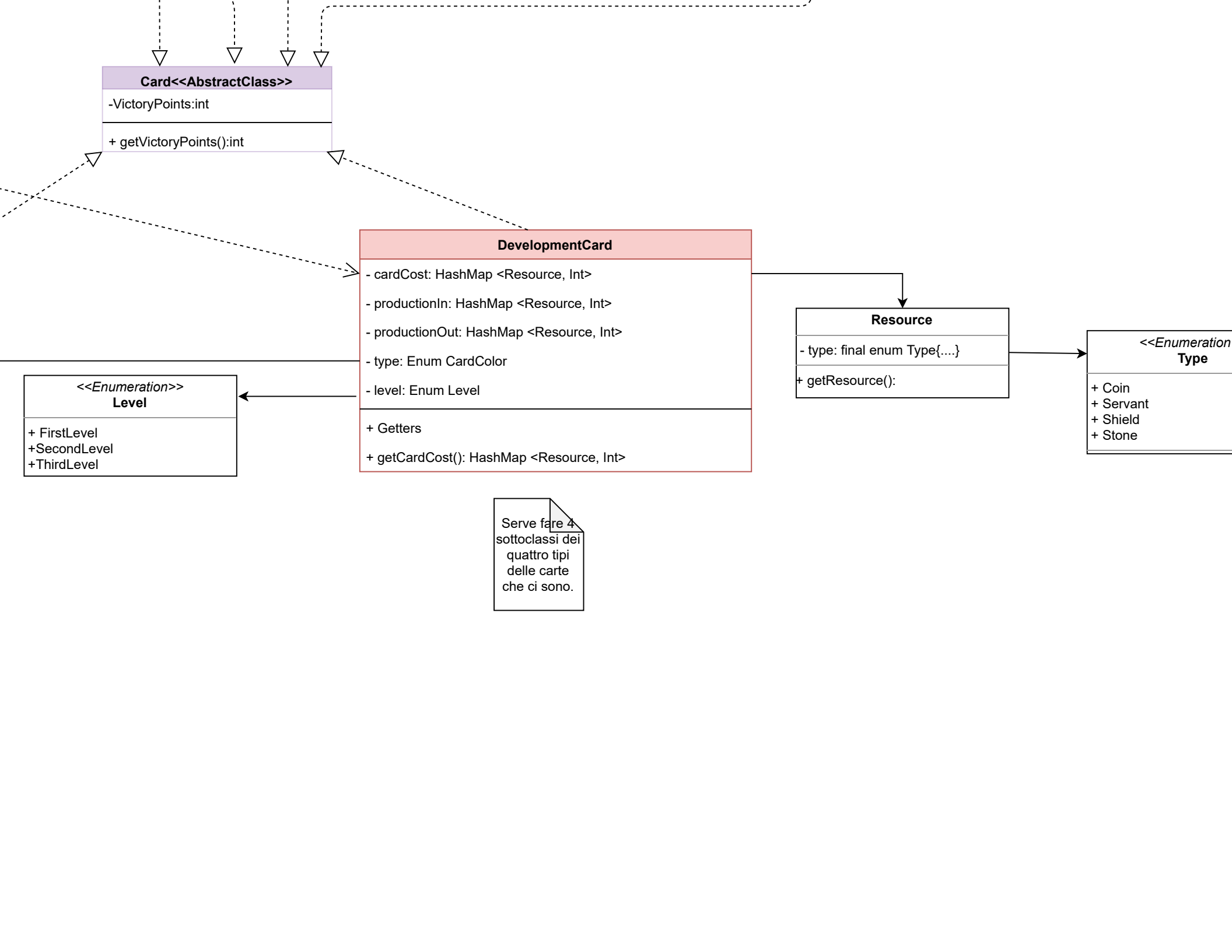

ConcreteProdPowerLeaderCardFacto
- discountedResourceType: Resource
+ Construct(DATA_IN): DiscountLeaderCar
Use
ProdPowerLeaderCard
- InProductionResourceType: HashMap <Resource, Int> (1)
-OutProductionResourceNum: int (1)
-OutProducrionFaithPointsNum: int (1)
+ getResource(): HashMap <Resource, Int>
+ receiveFaithPoints(): void
- OutProductio
+ getResource
mentCardFactory
rdCost, HashMap <String, Int> prodIn,
int level) : DevelopmentCard


ConcreteWhiteMarbleLeaderCardFactory
- resourceStorageType: Resource
+ Construct(DATA_IN): StorageLeaderCard
ConcreteStorageLeaderCardFactory
- marketResourceType: Resource
+ Construct(DATA_IN): WhiteMarbleLeaderCard
ConcreteDiscountLeaderCardFactory
- productionResourceType: Resource
+ Construct(DATA_IN): ProdPowerLeaderCard
Use
Use
Use
<<AbstractClass>>
LeaderCard
- activationCost: Map<T, int>
+ getActivationCost(): boolean
+ use(): void
+discardAndReceiveVPointsIfNotActivated():int
WhiteMarbleLeaderCard
nResource: HashMap <Resource, Int> (1)
(): HashMap <Resource, Int> (1)
StorageLeaderCard
- storageResource: <Resource, Int> (<(1Type),(2)>)
-occupationNumber:int
+ addToStorage(): void
+ getFromStorage(): void
+getOccupationNumber():Boolean
DiscountLeaderCard
- discountedResource: <Resource, Int> (<(1Type),(-1)>)
+getDiscoutResourceInfo(): <Resource, Int>



DATA_IN:might be
arguments(level,producrionIn,productionOut,VictoryPoints) or might be an
object that have there info or also might be just some
string that contain the info and then codificated inside the class while constructing.


Use
PapalFavorCard
<<Enumeration>>
CardColor
+ Green
+Blue
+Yellow
+Violet


Card<<AbstractClass>>
-VictoryPoints:int
+ getVictoryPoints():int
DevelopmentCard
- cardCost: HashMap <Resource, Int>
- productionIn: HashMap <Resource, Int>
- productionOut: HashMap <Resource, Int>
- type: Enum CardColor
- level: Enum Level
+ Getters
+ getCardCost(): HashMap <Resource, Int>
Resource
- type: final enum Type{....}
+ getResource():
<<Enumeration
Type
+ Coin
+ Servant
+ Shield
+ Stone
<<Enumeration>>
Level
+ FirstLevel
+SecondLevel
+ThirdLevel
Serve fare 4 sottoclassi dei quattro tipi delle carte che ci sono.