Progetto P2 - Note di sviluppo

Idea di progetto

Tool di tracking dell'inventario di un personaggio per un gioco di ruolo stile D&D. Dovrebbe garantire funzionalitá di creazione di un inventario, modifica ed eliminazione dello stesso.
Chiaramente un inventario viene associato ad un personaggio, ma deve essere discusso se si vuole dare una qualche forma di rappresentazione a questo aspetto nel programma finale o se si vuole semplicemente ignorare tale aspetto.

Gerarchia di classi

2 Classi base astratte pure:

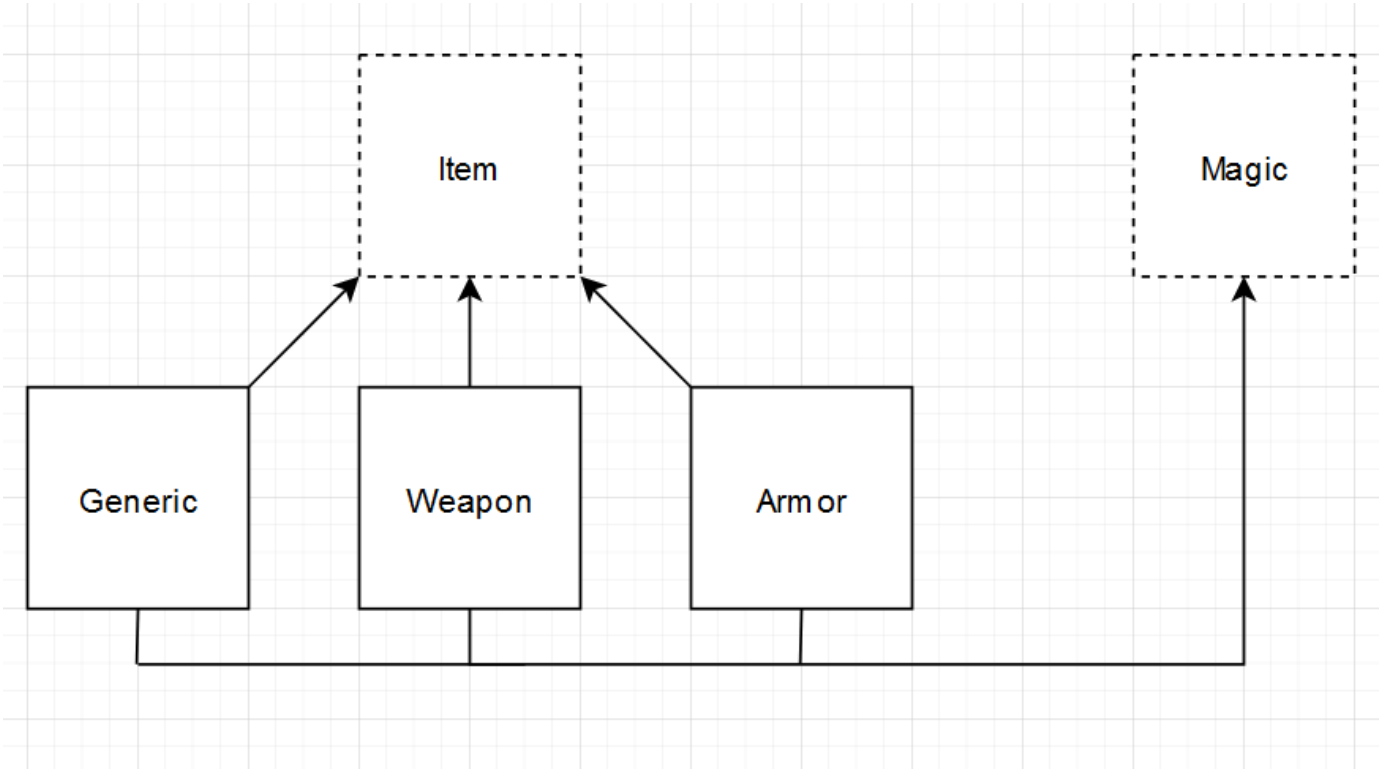
- Item
- Magic

Poi da **Item** derivo altre classi:

- Weapon
- Armor
- Generic

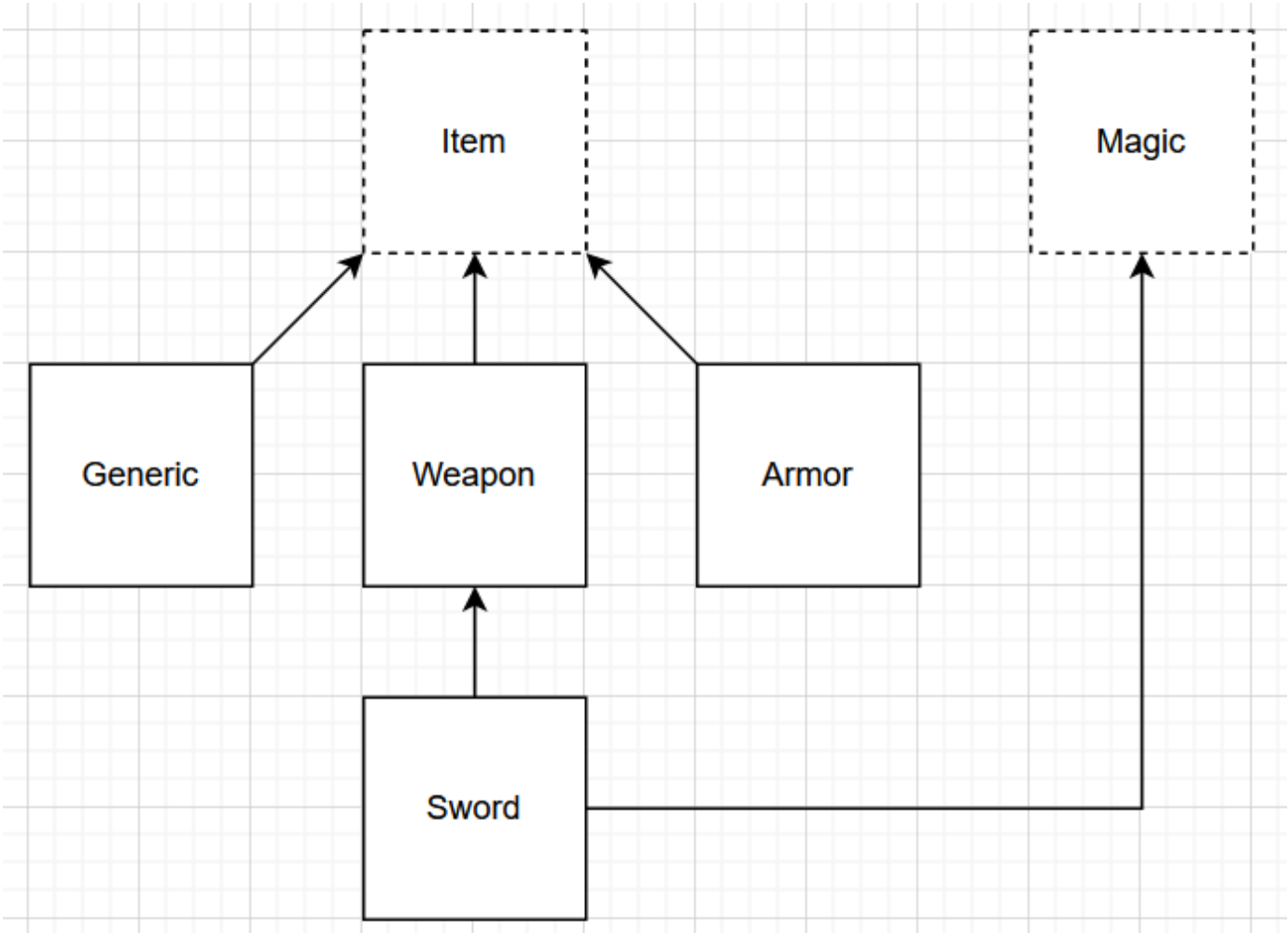
> **Problema**
>
> Primo punto critico: **come vogliamo rappresentare i singoli oggetti?**
> Si distinguono due casi (non possibili ma considerati):
>
> 1. Un arma é rappresentata come istanza di `Weapon`
> 2. Un arma é rappresentata come istanza di un `classe derivata da Weapon` che rappresenta l'arma specifica (Quindi una classe `Sword` che deriva da `Weapon`)

Nel caso 1, quello che otterremmo sarebbe una gerarchia come quella rappresentata nell'immagine qui soto



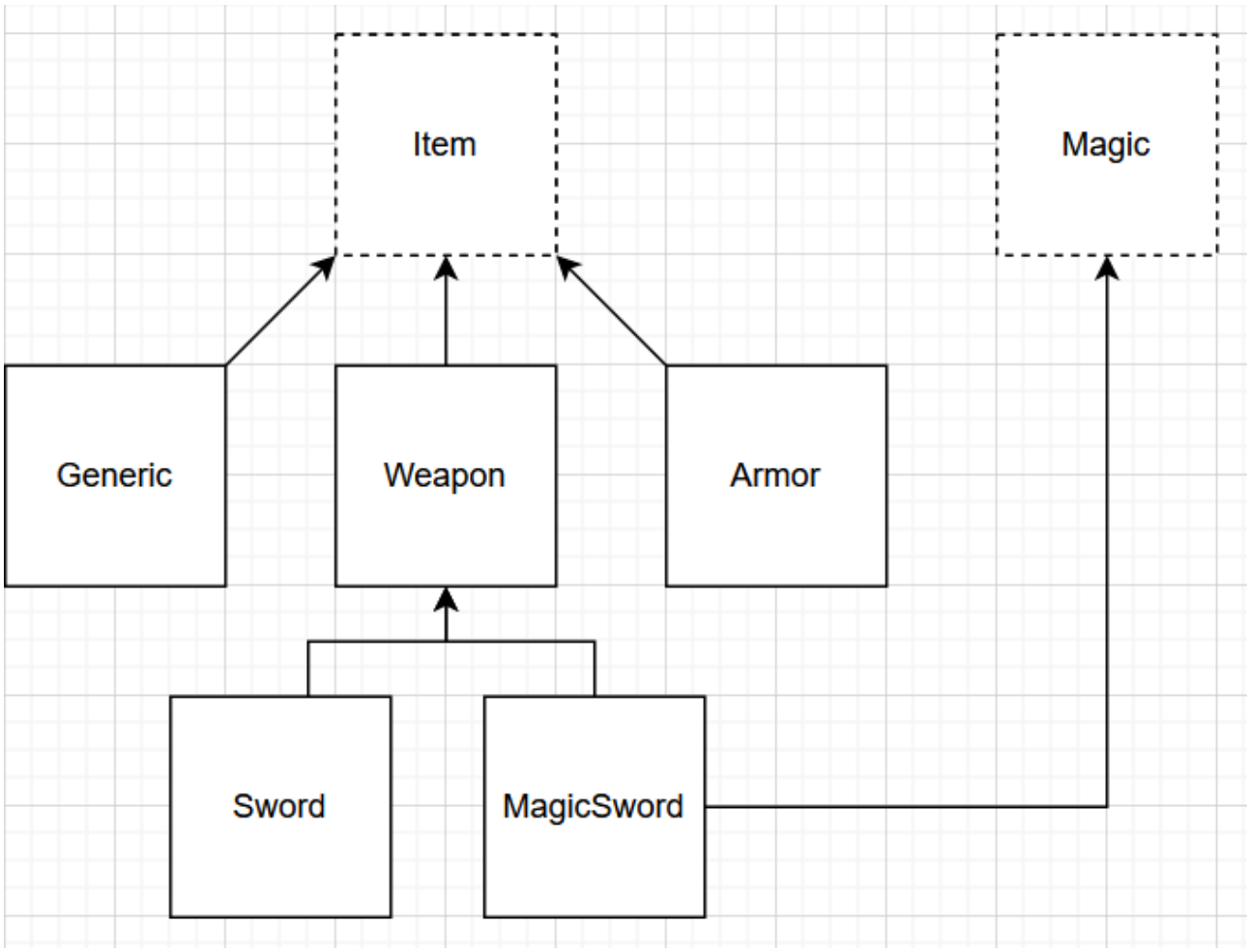
Nel caso 2 avremo invece una gerarchia differente, che cresce esponenzialmente con l'aumentare degli oggetti che vengono rappresentati in quanto per ogni oggetto rappresentabile deve essere creata una classe che estenda la gerarchia.

⚠ **Anche le classi che derivano di `Item` sono virtuali in questo caso, errore nel grafico**



Caso “extra” possibile

Ci sarebbe anche un caso 3, non menzionato prima, che consiste nel dividere gli oggetti magici da quelli non magici e che diventa problematico se si decide di procedere con la gerarchia del tipo 2.
Per esemplificare: una spada potrebbe essere magica cosí come potrebbe essere semplice. Se fosse semplice, avremmo una classe `Sword` che deriva solamente da `Weapon`, mentre se fosse magica avremmo un altra classe `MagicSword` che andrebbe a derivare sia da `Weapon` che da `Magic`.
⚠**Anche le classi che derivano di `Item` sono virtuali in questo caso, errore nel grafico**



Una rappresentazione di questo porterebbe inesorabilmente alla necessitá di creare coppie di classi per ogni oggetto che si vuole andare a rappresentare, andando a raddoppiare le classi presenti nella gerarchia e la complessitá generale.
Per questo motivo non ne ho parlato prima, ma rimane comunque una soluzione possibile se discutendone la si ritiene necessaria/utile

---

Struttura interna delle classi

Dati propri

A prescindere da quale sia la gerarchia scelta, questa sezione contiene un punto di partenza per decidere come rappresentare internamente gli oggetti di ciascuna classe. Non é definitiva e bisogna discuterne in maniera

approfondita.
La struttura data a questa documentazione é la seguente:

- definizione del nome dei campi dati e del tipo in forma tabulare
- specifica dell'informazione contenuta dal campo dati in forma testuale

La simbologia utilizzata é la seguente:

- ❓ ⇒ serve a indicare che é necessario discutere l'utilitá o la necessitá del campo dati
- ❗ ⇒ serve a indicare che nella specifica ci sono delle note aggiuntive sulla quale é necessario discutere

Classe Item

Campo Dati	Tipo	Note
name	string	❓
val	int	❗
weight	int	
integrity	int	❓
material	string	

name	Il nome dell'oggetto.
val	Il valore dell'oggetto ❗ richiede anche che venga definita una valuta per esprimerlo
weight	Il peso dell'oggetto
integrity	Un valore che indichi il livello di usura dell'oggetto
material	Il materiale di cui é composto l'oggetto

Classe Weapon

Campo Dati	Tipo	Note
atk	int	
typeDmg	string	❗
typeWeapon	string	❗

atk	Il valore di attacco dell'arma
typeDmg	Il tipo di danno che l'arma procura ❗ richiede che vengano decisi i tipo di danno da includere, in questo caso vengono considerati solo:

- bludgeoning (impatto)
- piercing (perforazione)
- slashing (taglio)

typeWeapon	Il tipo di arma, quindi *Melee* o *Ranged*

Classe Armor

Campo Dati	Tipo	Note
def	int	
typeArmor	string	

def	Il valore di difesa dell'armatura
typeArmor	Il tipo dell' armatura, quindi *heavy* o *light*

Classe Magic

Non c'é ancora stata alcuna riflessione seria su cosa questa classe dovrebbe rappresentare, da discutere.

Metodi

Non c'é ancora stata alcuna riflessione seria su quali dovrebbero essere i metodi che ciascuna classe dovrebbe incorporare, da discutere.
Gli unici metodi che sicuramente verranno inclusi saranno (se ritenuto necessario eplicitarli) *costruttore*, *distruttore* e *costruttore di copia*.
Nel caso in cui si decidesse di andare a utilizzare la gerarchia [2], ciascuna classe andrebbe solamente a implementare i metodi di Weapon.

Container

Per il container si era discusso sull'idea di utilizzare un Vector come struttura di partenza, da andare poi ad ampliare con altre funzionalitá.
Il polimorfismo diventa utilizzabile in maniera molto semplice istanziando il vector al tipo `Item*` .