Progetto P2 - Model V 0.3

Convezioni

Nomenclatura

Possibili convenzioni di nomenclatura:

- I nomi assegnati alle classi sono in lingua inglese con lettera maiuscola. (es: Item)
- I nomi assegnati alle variabili sono in lingua inglese composti unicamente da lettere minuscole. Se composti devono essere intervallati dal carattere '_' (es: weapon_name);
- I nomi dei metodi in lingua inglese. Se il metodo e' composto da più termini il primo ha lettera minuscola, tutte le iniziali dei successivi hanno lettera maiuscola (es. setName);
- Devono essere dichiarate le direttive d'uso che si vogliono usare per i metodi appartenenti a un namespace (es. se voglio non dover riscrivere std::endl ogni volta che lo utilizzo, devo notificare che viene dichiarata quella direttiva d'uso e in quale file);
- TUTTI i nomi che vengono utilizzati devono essere quanto più concisi possibile, prendendo come lunghezza indicativa tra i 5 e i 7 caratteri (non tassativo ma utile)

Sintassi e highlight nei documenti

- I nomi delle classi vengono evidenziate in modo da rendere chiaro che si parla di classi (es. `Item`).

- Per ogni classe sono definiti Campi dati e il significato degli stessi
- I campi dati delle classi sono presentati in forma tabulare
- Le descrizioni per i campi dati sono presentate in forma tabulare espansa

Cambiamenti rispetto alla versione precedente

Nella V02 della gerarchia c'erano alcuni problemi fondamentali di struttura e contenuto per i dati:

1. Le classi `Armor` e le sottoclassi di `Weapon` contenevano una grande quantita' informazioni simili, portando a ripetizione del codice in fase di definizione. Per ovviare al problema, vengono disposte come figlie di un unica interfaccia che racchiude quelle caratteristiche comuni, delegando alle sottoclassi l'onere di implementare aspetti piu' specifici. Questo permette inoltre di unificare tutti quegli oggetti che sono utilizzati come armamentario/equipaggiamento sotto un unica interfaccia, alleggerendo la gerarchia.Questo cambiamento comporta una serie di vantaggi:

- viene introdotta una distinzione tra oggetti *riproducibili* e *unici*. Per esemplificare: una spada lunga e' un arma generica, sia che essa possieda proprieta' magiche sia che non ne possieda. Una spada che esiste come oggetto unico e dotato di proprieta' uniche puo' essere introdotta attraverso l'implementazione di una sottoclasse che ne incapsuli i comportamenti unici.
- nel caso si desideri introdurre una nuova tipologia di oggetti che ricadono nella categoria degli equipaggiamenti sara' sufficiente introdurre una nuova sottoclasse di `EquipItem` che ne incapsuli le caratteristiche.

2. `Generic` (sottoclasse di `Item`), nella gerarchia precedente fungeva da interaccia per eventuali sottotipi di oggetto (tipi specializzati). A conti fatti pero' tutto quello che deve fare e' permettere la definizione di oggetti che non hanno proprieta' particolari al di fuori dell'essere esse stesse un oggetto. In questa iterazione non e' piu' interfaccia ma classe concreta che funge solo da implementazione elementare per la classe Item e che assolve al compito di permettere la definizione di oggetti "elementari", che non hanno alcuna proprieta' particolare (es. sasso, corda, torcia).
3. Come conseguenza del punto precedente, diventava difficoltoso definire in che modo si differenziassero gli oggetti che possiedevano proprieta' magiche (e che quindi assumevano una rilevanza maggiore) da quelli che invece erano definiti senza quelle proprieta'. Per ovviare al problema viene introdotta la classe `MagicItem` che appunto si prende carico di definire le proprieta' generiche di un oggetto magico.

Gerarchia di classi: visione d'insieme

La gerarchia si compone di:

- 2 classi base virtuali pure
  - Item
  - Spell
- 3 classi di prima derivazione
  - Generic (concreta)
  - EquipItem (virtuale)
  - MagicItem (virtuale)

Da EquipItem derivano due classi concrete

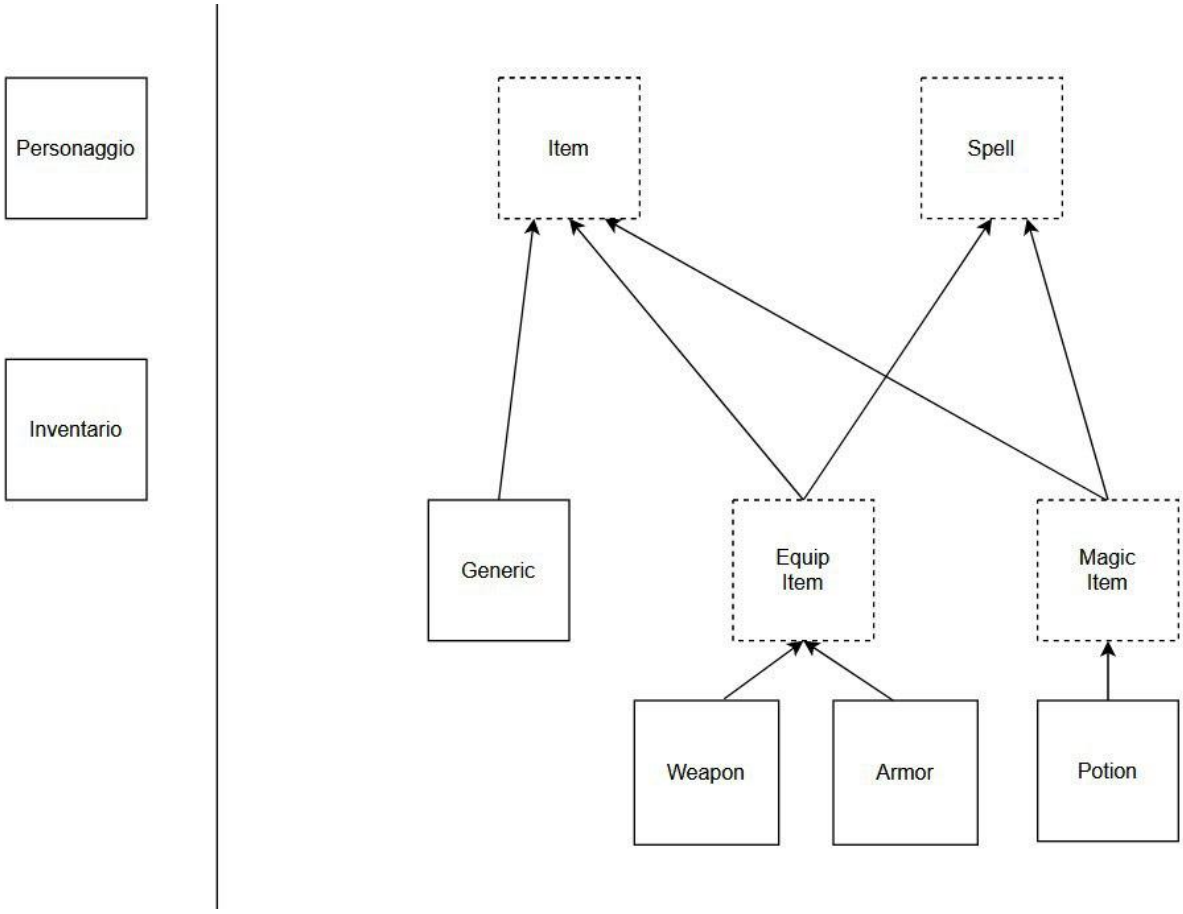
- Melee
- Ranged
- Armor

Da WizardItem deriva una classe concreta

- Potion

> **Sidenote**
>
> La classe **container** e' indicata con il nome temporaneo inventario
>
> La classe **Personaggio** identifica l'oggetto a cui viene associato il container
>
> Non viene specificato nulla di aggiuntivo qui circa queste due classi

Le due classi base permettono la definizione di qualunque oggetto rappresentabile e utilizzabile in un gioco di ruolo moderno.
L'idea fondante consiste nel derivare da queste due classi qualora si volesse creare una nuova categoria di oggetti, oppure rafffinare le categorie gia' presenti implementando classi che sono figlie di EquipItem, Generic o MagicItem.



Classe `Item`

Classe base virtuale pura che rappresenta le caratteristiche generiche di un oggetto.

Campi dati

Campo Dati	Tipo
name	string
value	int
weight	int
material	string

Definizione dei campi dati

name	Il nome dell'oggetto.
val	Il valore dell'oggetto
weight	Il peso dell'oggetto
material	Il materiale di cui é composto l'oggetto

Classe `Spell`

Classe base virtuale pure che rappresenta la magia come proprietà che può essere associata ad un item.

> **N.B.**
>
> Da osservare che la non presenza di proprietà magiche in un item viene rappresentata come una magia nulla e che quindi viene richiesto un costruttore nullo

Campi dati

Campo Dati	Tipo
name	string
type	string
level	int
spell_power	int
duration	int

Definizione dei campi dati

name	Il nome della magia
type	Il tipo di magia (es. Fuoco)
level	Il livello della magia
power	Il danno della magia
duration	La durata della magia

Classe `Generic`

Classe derivata unicamente da Item che rappresenta un item che non possiede proprieta' magiche. Non implementa campi dati propri ma si limita a ridefinire i metodi ereditati da Item.

Classe `EquipItem`

Classe derivata da Item e Magic che rappresenta il concetto generico di equipaggiamento da battaglia

Campi dati

Campo Dati	Tipo
equip_power	int
type	string
rarity	string
equipped	bool

Definizione dei campi dati

equip_power	Il valore di potenza dell'equipaggiamento
type	Il tipo di equipaggiamento, distinguibile in: • leggero • pesante
rarity	La rarità dell'arma che ha come valori possibili: • commmon • rare • epic • legendary

> ! **N.B.**

> Questa classe indica una generalizzazione di un oggetto, quindi non ha senso andare a definire un livello di rarita' **Unique** per identificare un oggetto non categorizzabile (es. Excalibur e' un oggetto unico, non riproducibile). Nel caso in cui si volesse creare un oggetto unico si deve creare una nuova classe che deriva da `EquipItem` e che identifica quello specifico oggetto.

Classe `MagicItem`

La classe MagicItem identifica un oggetto generico che pero' gode di proprieta' magiche e che quindi richiede una specifica rappresentazione.

Campi dati

Campo Dati	Tipo
range	int
description	string

Definizione dei campi dati

range	L' area di effetto della magia
description	La descrzione dell' effetto della magia

Classe `Weapon`

Campi dati

Campo Dati	Tipo
melee_type_dmg	string
ranged_type_dmg	string
min_range	int
max_range	int

Definizione dei campi dati

melee_type_dmg	Il tipo di danno che l'arma e' in grado di infliggere corpo a corpo

ranged_type_dmg	Il tipo di danno che l'arma e' in grado di infliggere dalla distanza, se vi e' modo di farne.
min_range	La distanza minima a cui e' possibile colpire
max_range	La distanza massima a cui e' possibile colpire

Classe `Armor`

Classe derivata da Item e Magic che rappresenta un armatura nella sua interezza. Si occupa solamente di definire i metodi che deriva dalle classi da cui deriva, senza aggiungere informazioni particolari.

Classe `Potions`

Classe derivata da MagicItem e che definisce gli oggetti pozione. Si limita ad implementare i metodi che vengono forniti dalla classe MagicItem.