



UNIVERSITÀ
DEGLI STUDI
FIRENZE



Esercitazione

Generic programming exercise

Obiettivo

- Il progetto CLion fornito contiene classi e scheletri di classi relative al gioco in stile Rogue (<https://it.wikipedia.org/wiki/Roguelike>) della scorsa esercitazione.
- Scopo della presente esercitazione è:
 - Implementare una classe template base che rappresenti un inventario di oggetti dello stesso tipo
 - Es. un inventario di pozioni o di armi
 - Estendere la classe template inventario per rappresentare uno scrigno di oggetti, con posizione e attributo che rappresenta la chiusura/apertura dello scrigno
 - Implementare una funzione generica per il calcolo della distanza L1 tra due oggetti diversi del gioco (es. un GameCharacter ed uno scrigno), a patto che entrambi abbiano metodi per ottenere le coordinate 2D

Schema del codice

- Il programma è composto da 11 classi di partenza. Lo schema del codice delle classi di base è lo stesso di quello della volta scorsa:
- Dungeon crea mappe casuali con stanze, corridoi, scale, porte, etc.
- Weapon rappresenta un arma con forza e magia
 - Bow e Sword estendono Weapon
- GameCharacter rappresenta un personaggio del gioco, ed è composto con Weapon.
 - Orc, Skeleton, Knight e Wizard estendono GameCharacter
- Dice rappresenta un dado
- Potion rappresenta una pozione, con diversi tipi ed effetti

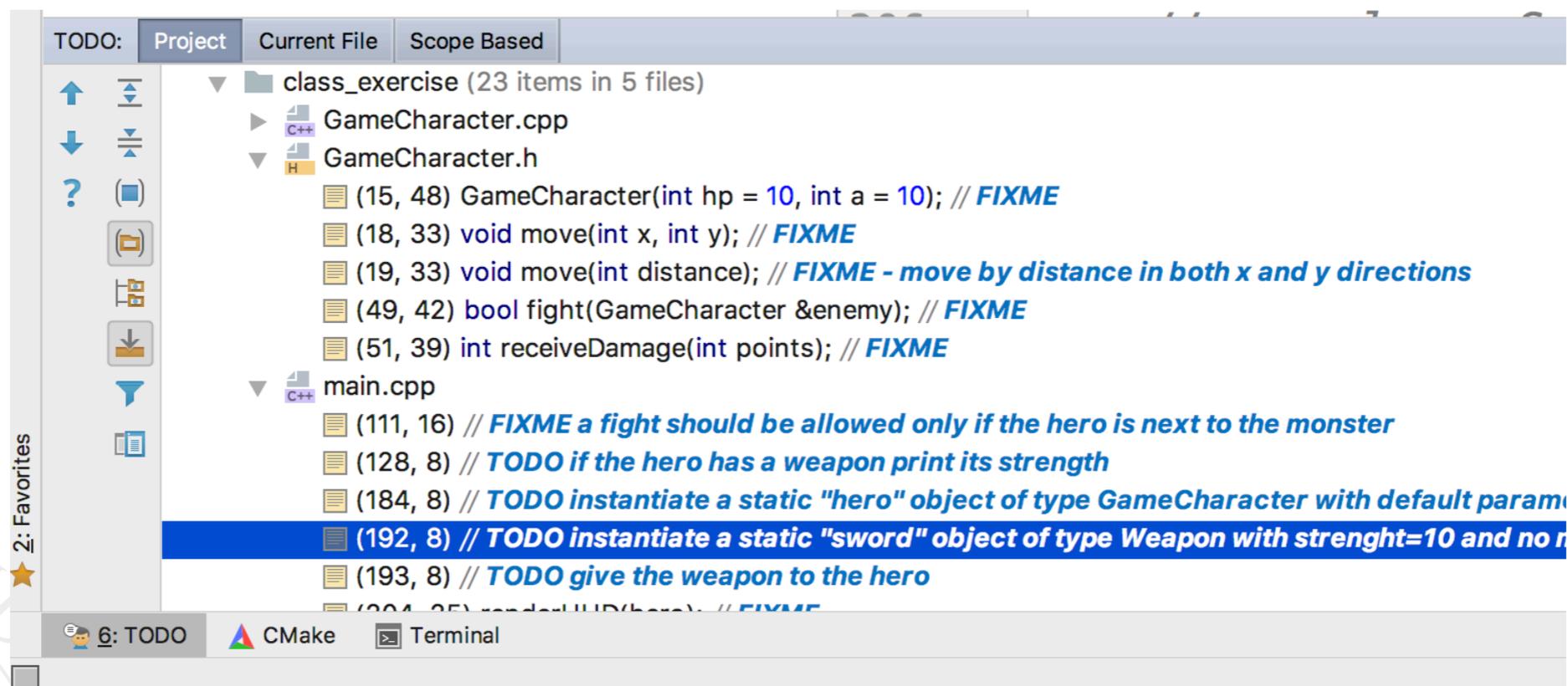
Schema del codice

- In questa esercitazione andremo ad aggiungere due classi: `Inventory` per l'inventario e `Vault` per lo scrigno. `Vault` estende `Inventory`.
- Implementeremo la funzione generica per il calcolo della distanza L1 in `Utilities.h`.
- Modificheremo il `main.cpp` dove indicato per istanziare oggetti generici e invocare la funzione `template`.



Dove modificare il codice

- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con TODO e FIXME
- Per vedere tutti questi commenti selezionare la finestra TODO di CLion



Dove modificare il codice

The screenshot shows the CLion IDE interface. The main editor displays the code in `main.cpp` with the following content:

```
185 // find a legal start position
186 int startX = 0;
187 int startY = 0;
188 setupCharacterCell(startX, startY, map);
189 hero.setPosX(startX);
190 hero.setPosY(startY);
191 // create a weapon and give it to hero
192 // TODO instantiate a static "sword" object of type Weapon with strenght=10 and
193 // TODO give the weapon to the hero
194 // create an enemy with a low grade armor
195 GameCharacter enemy(20, 2);
196 // find monster position not too far from hero position
197 startX += 5;
198 startY += 3;
199 setupCharacterCell(startX, startY, map);
200 enemy.setPosX(startX);
201 enemy.setPosY(startY);
202
203 // render
204 renderHUD(hero); // FIXME
205 renderGame(map, hero, enemy);
```

The TODO list at the bottom of the IDE shows the following items:

- (15, 48) `GameCharacter(int hp = 10, int a = 10);` // FIXME
- (18, 33) `void move(int x, int y);` // FIXME
- (19, 33) `void move(int distance);` // FIXME - move by distance in both x and y directions
- (49, 42) `bool fight(GameCharacter &enemy);` // FIXME
- (51, 39) `int receiveDamage(int points);` // FIXME
- (111, 16) // FIXME a fight should be allowed only if the hero is next to the monster
- (128, 8) // TODO if the hero has a weapon print its strength
- (184, 8) // TODO instantiate a static "hero" object of type GameCharacter with default parameters
- (192, 8) // TODO instantiate a static "sword" object of type Weapon with strenght=10 and no magic
- (193, 8) // TODO give the weapon to the hero
- (204, 25) `renderHUD(hero);` // FIXME

The status bar at the bottom indicates the current context is `class_exercise [D]`.

Classe Inventory

- Implementata in `Inventory.h` rappresenta un contenitore di oggetti con un numero massimo di oggetti.
- Gli oggetti sono contenuti in un array, per ogni posizione abbiamo un valore Booleano corrispondente per indicare se lo slot è libero (oggetto ancora non messo o prelevato) oppure no.
- Un metodo `setElement` riceve una posizione dell'array e l'oggetto da inserire nell'inventario. Se lo slot è disponibile inserisce l'oggetto e rende `true`, altrimenti rende `false`.
- Un metodo `getElement` riceve una posizione dell'array e una variabile dove mettere l'oggetto dell'inventario da prendere. Se lo slot indicato contiene un oggetto lo mette nella variabile, imposta lo slot a libero e rende `vero`, altrimenti rende `falso`.
- Il metodo `printContent` stampa il contenuto dell'inventario stampando cosa c'è in ogni slot pieno.

Classe Vault

- Estende `Inventory`, aggiungendo le coordinate 2D della posizione dello scrigno ed un attributo `closed` Booleano che indica se aperto.
- Il metodo `printContent` stampa l'inventario solo se lo scrigno è aperto.
- Il metodo `open` imposta a `false` l'attributo `closed`



Funzione L1 Distance

- Funzione generica che calcola la distanza L1 tra due oggetti di qualsiasi tipo del gioco



Main

- Istanziare un Vault che contenga armi (di qualsiasi tipo) ed un Inventory che contenga pozioni.
- É già presente il codice che consente di aprire il vault se è vicino (usando la funzione `template I1Distance`)