



UNIVERSITÀ
DEGLI STUDI
FIRENZE



Esercitazione

Abstract class + RTTI 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:
 - Far diventare GameCharacter e Weapon classi base astratte
 - Sarà necessario implementare una tecnica chiamata “virtual constructor” per consentire la copia di Weapon in GameCharacter
 - Usare RTTI per consentire a giocatori che usano dei Wizard di fare magie, e fare in modo che l’interfaccia mostri il tipo di personaggio del giocatore
 - Usare ereditarietà multipla per creare una nuova classe MageKnight che combatte come un cavaliere (senza perdere mana) e fa magie come un Wizard (ma solo se vicino al nemico)

Schema del codice

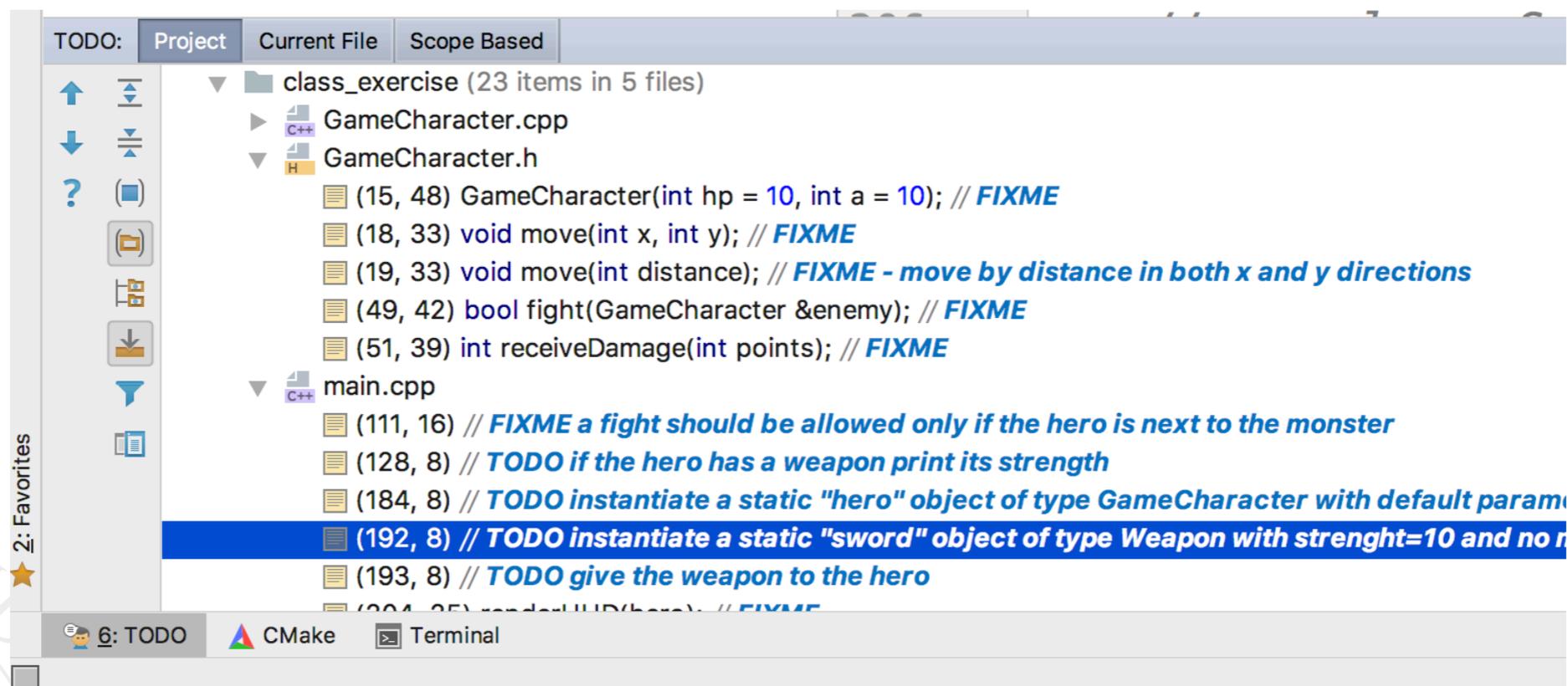
- Il programma è composto da 10 classi di partenza. Lo schema del codice delle classi di base è lo stesso di quello della volta scorsa:
- Dungeon crea mappe casuali con stanze, corridoi, scale, porte, etc.
- Weapon rappresenta un arma con forza e magia
 - Bow e Sword estendono Weapon
- GameCharacter rappresenta un personaggio del gioco, ed è composto con Weapon.
 - Orc, Skeleton, Knight e Wizard estendono GameCharacter
- Dice rappresenta un dado

Schema del codice

- In questa esercitazione andremo ad aggiungere una classe con ereditarietà multipla (MageKnight) e modificheremo le classi Weapon e GameCharacter, oltre che le classi derivate Bow, Sword, Knight e Wizard
- Modificheremo il main per implementare RTTI.

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 C++ 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 panel at the bottom left 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 right shows: 28:32 LF+ UTF-8+ Context: class_exercise [D]

Classe Weapon

- La classe deve diventare astratta, rendendo puramente virtuale il metodo `int use()`
- La classe mette a disposizione delle classi derivate il metodo `basicUse()` fornisce comunque un'implementazione di base dell'uso dell'arma.
- La classe deve implementare uno schema di design del software chiamato costruttore virtuale per consentire il funzionamento della copia/assegnazione di `GameCharacter`

Classe Sword

- Estende Weapon, implementando use().
 - Se la spada è in acciaio di Valiria fa più danni... si sfrutti basicUse() nell'implementazione
- Si deve aggiungere un metodo per l'implementazione del costruttore virtuale

Classe Bow

- Estende Weapon, implementando use().
 - si sfrutti basicUse() nell'implementazione, tenendo conto della disponibilità di frecce e aggiornandone il numero
- Si deve aggiungere un metodo per l'implementazione del costruttore virtuale

Classe GameCharacter

- Questa è una classe base per Orc, Skeleton, Knight e Wizard. Renderla astratta col metodo puramente virtuale:
`int fight(GameCharacter &enemy)`
- Modificare costruttore di copia e operatore di assegnazione per sfruttare il costruttore virtuale di Weapon



Classe Wizard e Knight

- Non vanno modificate: i loro override già presenti sono sufficienti per renderle classi concrete.
- Andranno modificate solo per risolvere il problema del diamante quando si implementa MageKnight...



Main

- Usare RTTI per stampare nelle istruzioni la disponibilità del comando “m” nel caso si abbia un mago o uno MageKnight
- Usare RTTI per stampare il tipo di personaggio del giocatore
- Usare RTTI per controllare che il personaggio del giocatore sia Knight o mageKnight e nel caso eseguire doMagic()
- Consentire la creazione di personaggi del giocatore di tipo MageKnight
 - Prima scrivere il codice riferendosi a Wizard e poi una volta fatta la classe MageKnight estenderlo.

Classe MageKnight

- Modificare la classe per estendere in modo multiple sia Knight che Wizard
- Fare override sia di `move()` che di `fight()`
- Implementare il costruttore per fare in modo che la componente Knight NON sia mai un paladino. Usare gli argomenti del costruttore per invocare entrambi i costruttori delle classi base



Classe Dungeon

- Questa classe non deve essere toccata

