UNIVERSITÀ
DEGLI STUDI
FIRENZE

Esercitazione

Class exercise

Obiettivo

- Il progetto CLion fornito contiene classi e scheletri di classi che implementano un gioco in stile Rogue (https://it.wikipedia.org/wiki/Roguelike)

- Scopo dell'esercitazione è:
  - Completare le classi GameCharacter e Weapon, scrivendo codice di diversi metodi
  - Correggere il comportamento del gioco nel file main.cpp per fare in modo che il combattimento tra l'eroe ed un mostro sia più realistico

Obiettivo


- Muro
Corridoio buio
. Zona illuminata
$ Oro
+ Porta
| Muro
! Pozione magica
@ Il protagonista
D Drago rosso
< Scale per livello sup.
? Pergamena magica

Schema del codice

- Il programma è composto da 3 classi:
- Dungeon crea mappe casuali con stanze, corridoi, scale, porte, etc.
- `Weapon` rappresenta un arma con forza e magia
- `GameCharacter` rappresenta un personaggio del gioco, ed è composto con `Weapon`, sfruttandone il metodo `use()` per calcolare l'effetto del combattimento contro un altro `GameCharacter`.

Schema del codice

- In main sono presenti diverse funzioni che implementano elementi fondamentali del gioco:
  - `getEvent()` interagisce col giocatore tramite la tastiera, ricevendo i comandi
  - `isLegalCell()` controlla che una cella della mappa del Dungeon possa ospitare un personaggio
  - `setupCharacterCell()` cerca delle coordinate nella mappa dove disporre un personaggio all'avvio del gioco ("spawn")
  - `isLegalMove()` controlla che il movimento di un personaggio sia consentito, ovvero si muova su una cella che lo può contenere, controllando che questa cella già non contenga un altro personaggio.
  - `updateGame()` aggiorna lo stato del gioco secondo i comandi del giocatore
  - `renderHUD()` disegna sullo schermo le caratteristiche del giocatore (head up display)
  - `renderGame()` disegna sullo schermo la mappa ed i personaggi (operazione di "compositing")

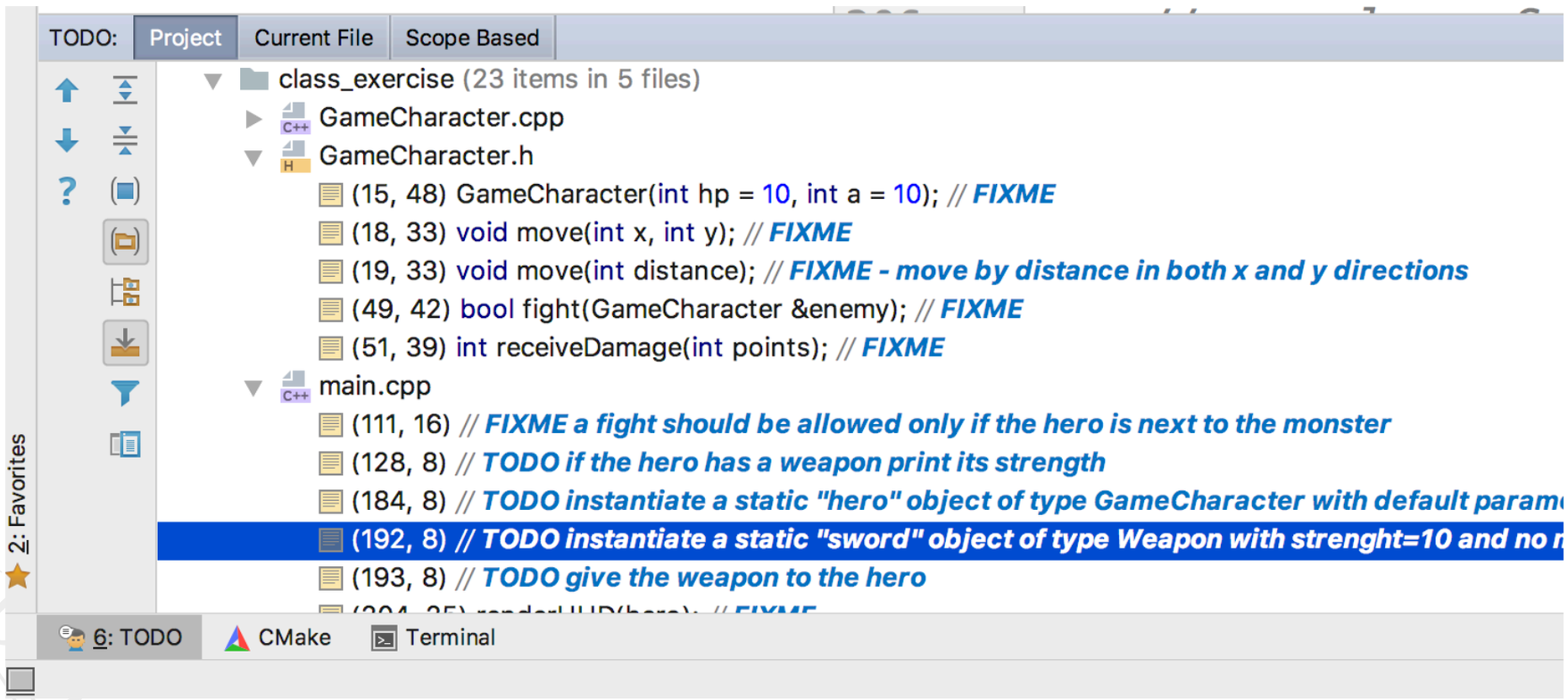
Schema del codice

- La funzione main() provvede ad istanziare personaggi, mappa ed armi. Il cuore è un classico “game loop”, ovvero un ciclo infinito (a meno di richiesta di fine gioco) in cui:
  - Si ricevono gli eventi del gioco (in questo caso i comandi del giocatore)
  - Si aggiorna lo stato del gioco sulla base degli eventi
  - Si disegna (rendering) il tutto:
    - Disegnando lo HUD
    - Disegnando la schermata del gioco

Dove modificare il codice

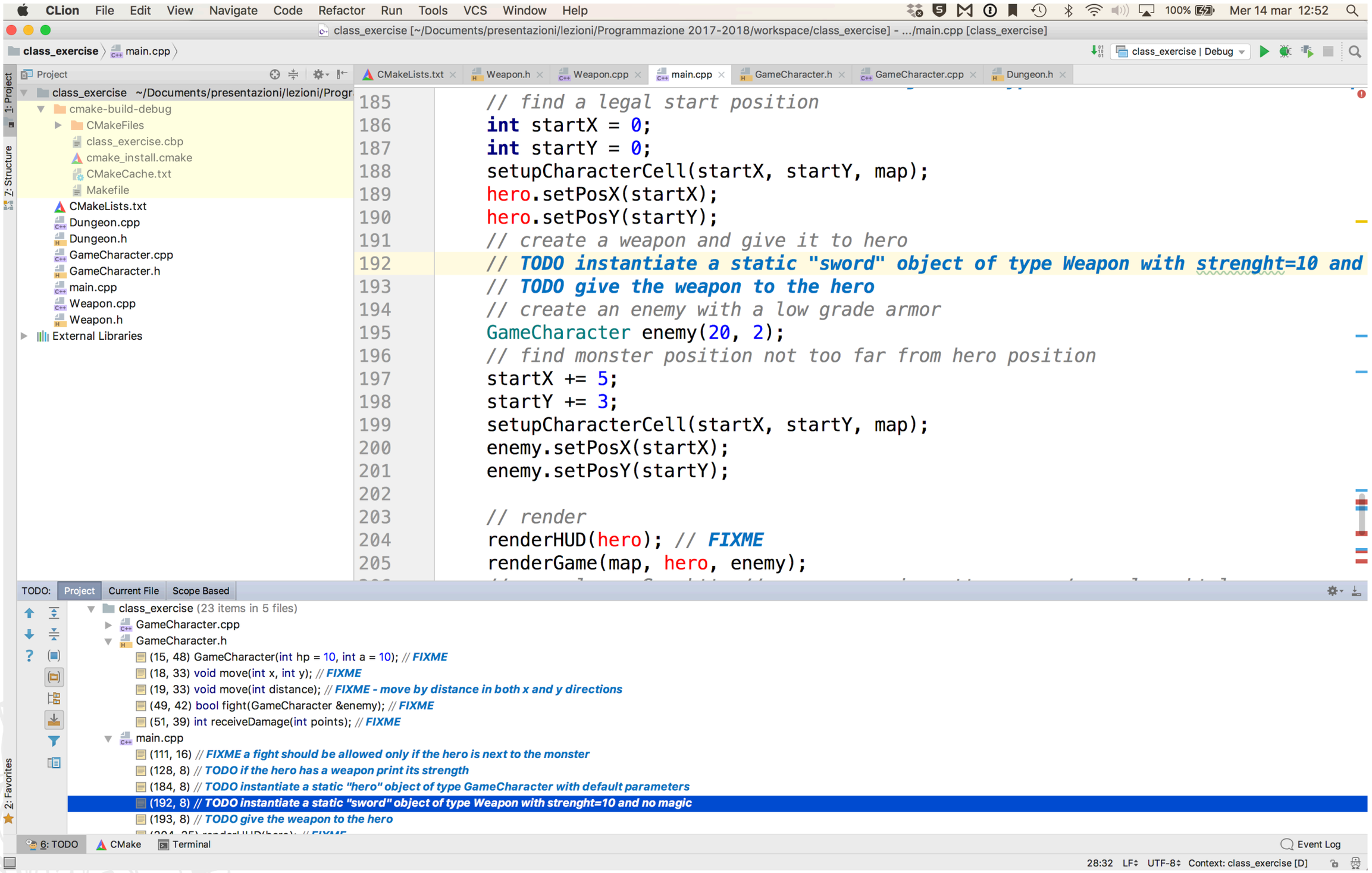
- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con `TODO` e `FIXME`

- Per vedere tutti questi commenti selezionare la finestra TODO di CLion



UNIVERSITÀ DEGLI STUDI FIRENZE

Dove modificare il codice



Classe Weapon

- Implementare il costruttore
- Implementare getter/setter per I due attributi
- Implementare il metodo `use()` secondo le specifiche nei commenti

Classe GameCharacter

- Implementare il costruttore (la signature è fornita)
- Implementare il metodo `move()` con due overload (signature fornite)
- Completare l'implementazione dei metodi `fight()` e `receiveDamage()`
- Notare come la classe `GameCharacter` sia composta con `Weapon`. Attenzione al fatto che `Weapon` è un puntatore e deve essere inizializzato e controllato prima del suo uso.

Main

- Istanziare oggetti: `hero` di tipo `GameCharacter`, `sword` di tipo `Weapon`, e `enemy` di tipo `GameCharacter`

- Completare la funzione `renderHUD()` per fare in modo che se `hero` ha un'arma a disposizione ne venga stampata la forza

- EXTRA: modificare la funzione `updateGame()`, eventualmente aggiungendo nuove funzioni di aiuto per fare in modo che il giocatore possa attaccare il mostro solo quando è in una posizione adiacente.

Classe Dungeon

- Questa classe non deve essere toccata