UNIVERSITÀ
DEGLI STUDI
FIRENZE

Esercitazione

Inheritance 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:
  - Estendere le classi GameCharacter e Weapon, partendo da scheletri di classi derivate
    - Costruttori, metodi, attributi
  - Vedere come usare puntatori e riferimenti a classi base per usare oggetti di tipo derivato.

Schema del codice

- Il programma è composto da 4 classi di partenza:
- 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`.
- `Dice` rappresenta un dado

Schema del codice

- Lo schema del codice delle classi di base è lo stesso di quello della volta scorsa

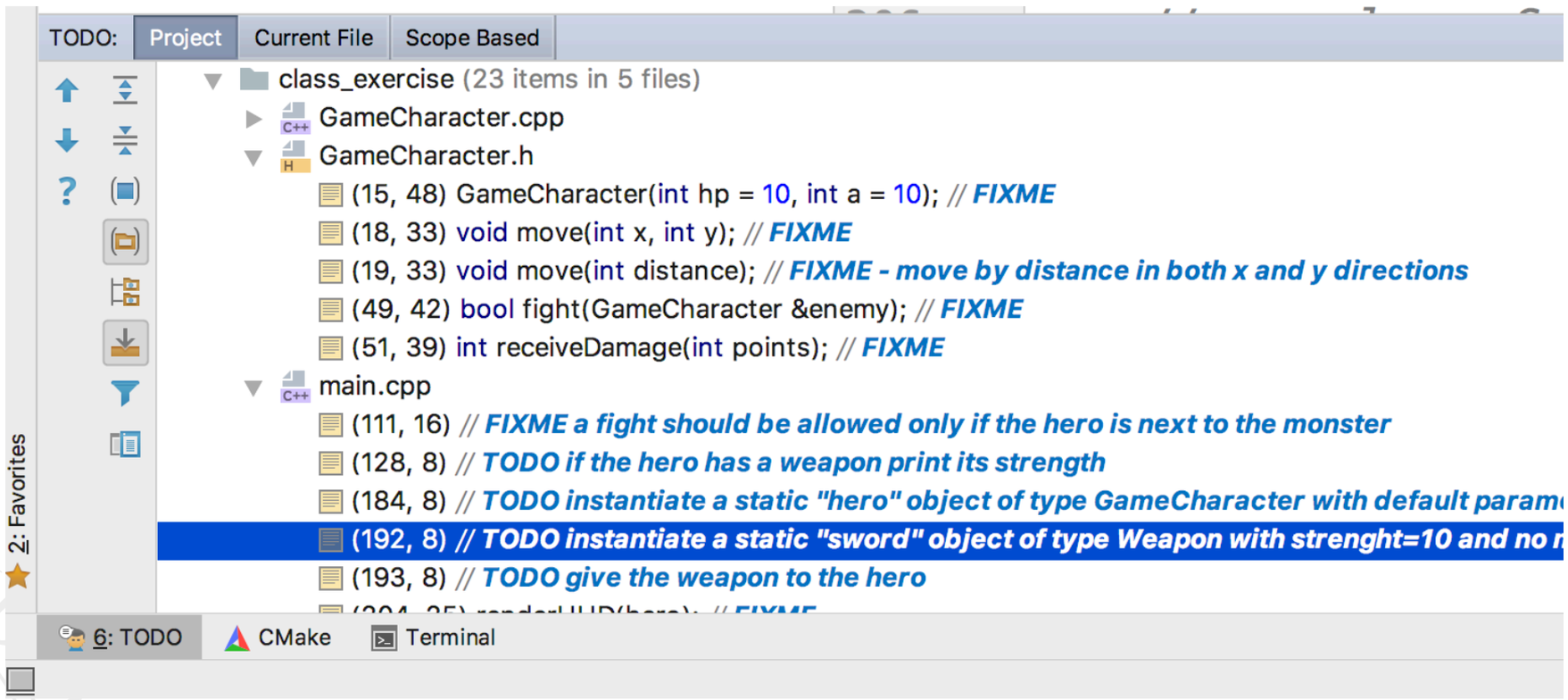
- In questa esercitazione andremo ad estendere le classi GameCharacter e Weapon per aggiungere nuove funzionalità come due tipi di armi diverse (Sword e Bow), due tipi di personaggi per il giocatore (Knit e Wizard) e due mostri (Orc e Skeleton)

- Queste classi derivate hanno funzionalità diverse, modificando il comportamento di metodi di classi base, aggiungendo nuovi metodi ed attributi.

Dove modificare il codice

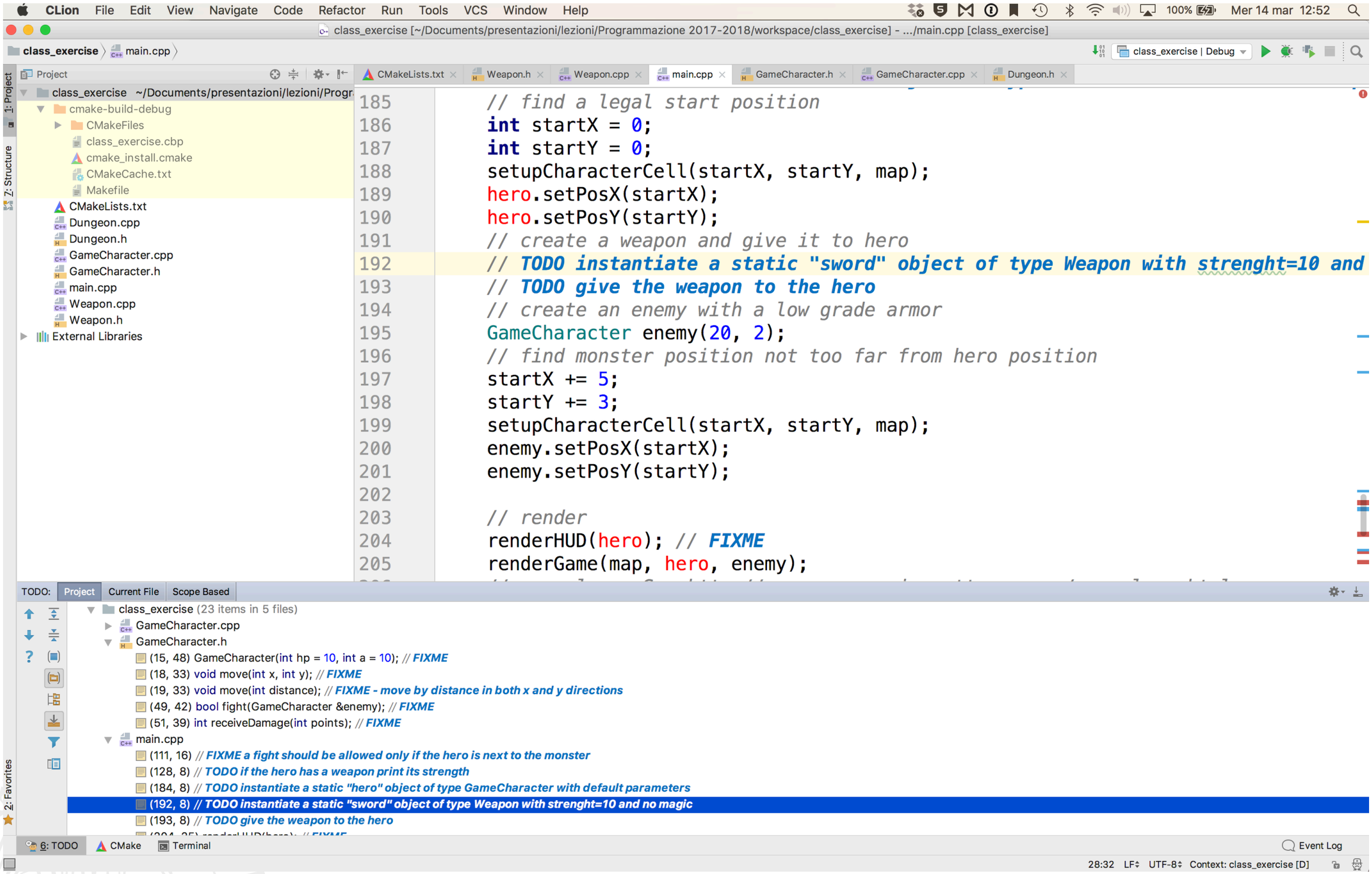
- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con `TODO` e `FIXME`

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



UNIVERSITÀ DEGLI STUDI FIRENZE

Dove modificare il codice



Classe Weapon

- Questa è una classe base per `Sword` e `Bow`
- Il metodo più interessante è `use()` e ci si aspetta che debba essere implementato in modo diverso nelle classi derivate, per cui è stato marcato come `virtual`
- E' l'unico cambiamento. Si può usare come esempio per rendere classe base GameCharacter

Classe Sword

- Estende Weapon, mostrando come fare override e come aggiungere attributi.
- Se la spada è in acciaio di Valiria fa più danni…
- Usarla come ispirazione per il lavoro sulla classe Bow

Classe Bow

- Estende `Weapon`, rappresenta un arco, ovvero un'arma che ha un numero limitato di colpi (le frecce).
  - Dobbiamo inizializzare l'attributo del numero delle frecce e fare l'override del metodo use per fare in modo che si possa attaccare solo se si hanno frecce e si decrementi il numero di frecce ad ogni uso.
- Si può riusare il metodo della classe base all'interno del metodo derivato. La sintassi per invocarlo è `Weapon::use()`

Classe GameCharacter

- Questa è una classe base per `Orc`, `Skeleton`, `Knight` e `Wizard`

- Bisogna decidere quali metodi rendere metodi di cui fare override nelle classi derivate, per fare in modo che queste possano fornire varianti e usarli in modo polimorfico

  - Guardare i metodi selezionati come candidati.

  - Un metodo nuovo è `getCharacterSymbol()` che è usato nel rendering, usando una lettera diversa per ogni tipo di personaggio.

Classe Orc e Knight

- Il lavoro da fare è molto simile.

- Quando si fa l'override di un metodo usare `override`, per essere sicuri di avere la signature corretta (es. non dimenticarsi un const di un metodo…)

- Per Orc l'implementazione di due metodi come move e fight è già fornita… basta fare la dichiarazione

  - Ispirarsi a questo codice per l'implementazione dei metodi di Knight

Classe Skeleton

- Il metodo receiveDamage è già implementato, basta dichiararlo

- Implementare solo il costruttore, gli altri metodi vengono semplicemente ereditati dalla classe base

Classe Wizard

- Aggiunge un nuovo metodo (già implementato)
- Per uno dei metodi d cui fare override è già fornita l'implementazione, basta fare la dichiarazione

Main

- Istanziare oggetti: armi, personaggi del giocatore e mostro.

- Notare come si usino puntatori e riferimenti a classe base per puntare/riferirsi a oggetti di tipo derivato

- Indipendentemente dal tipo di oggetto derivato a cui ci si riferisce/punta si invoca il metodo specifico della classe derivata.

  - E’ un esempio di **polimorfismo** !

Classe Dice

- Questa classe non deve essere toccata

Classe Dungeon

- Questa classe non deve essere toccata