UNIVERSITÀ
DEGLI STUDI
FIRENZE

Esercitazione

Exam exercises

Obiettivo

- Scopo della presente esercitazione è:
  - Risolvere esercizi di programmazione presentati in compiti di esame
  - Implementare design pattern
  - Implementare eccezioni, RTTI, polimorfismo

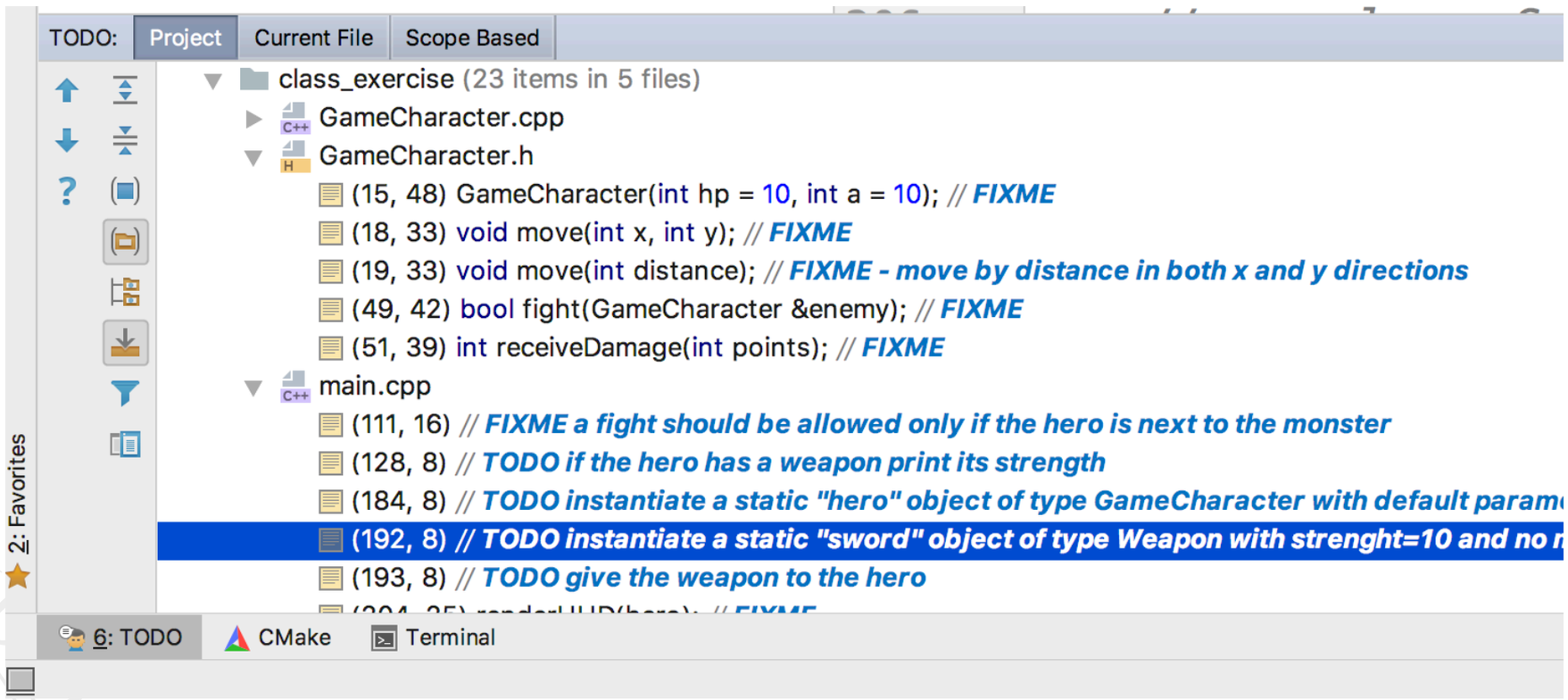
Schema del codice

- Per ogni esercizio sono presenti frammenti di codice da modificare e la descrizione dell'esercizio
- Per risolvere l'esercizio può essere necessario modificare il codice fornito, es. per fare in modo che una classe estenda una qualche classe base, o aggiungendo metodi e attributi

Dove modificare il codice

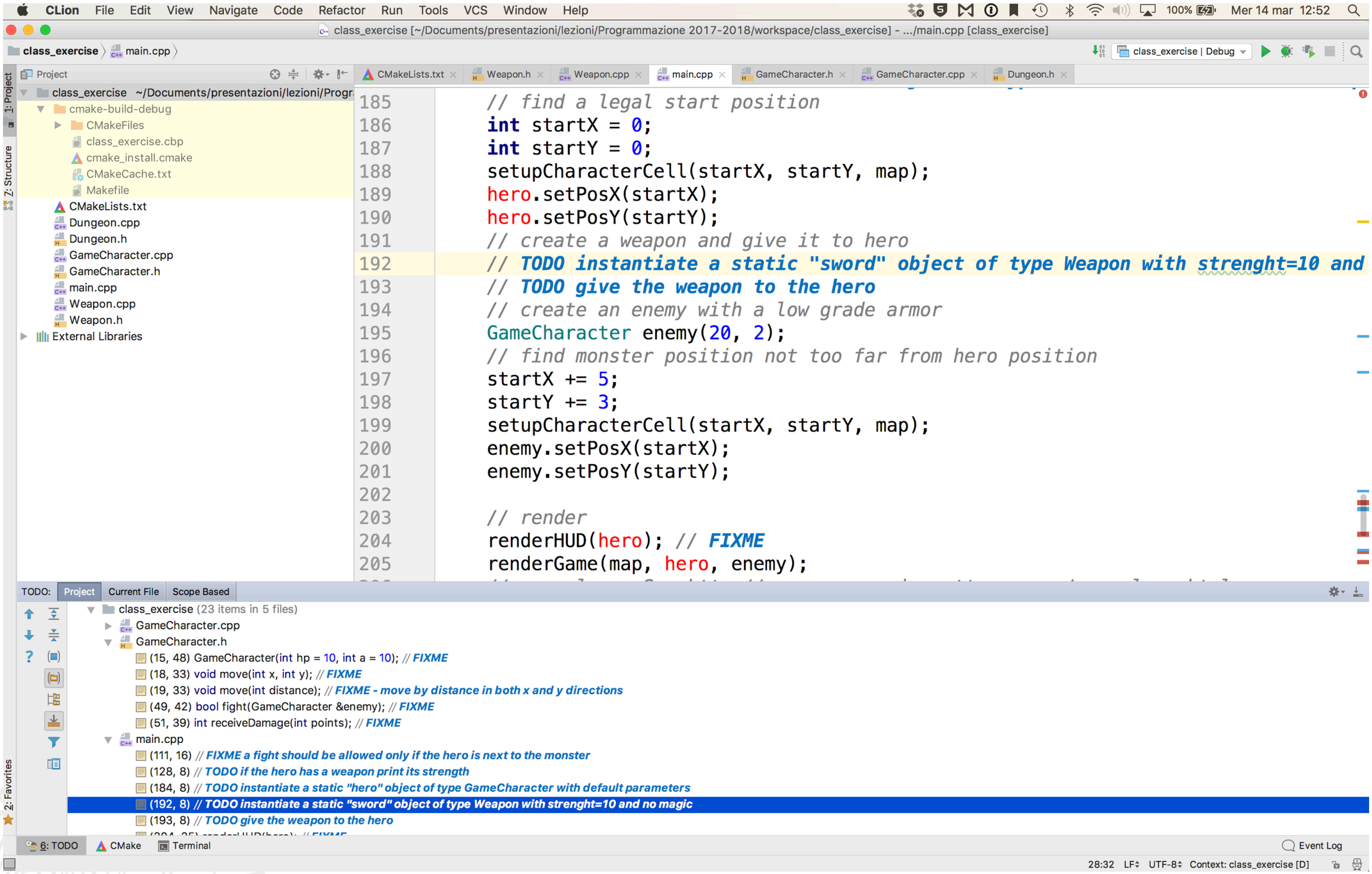
- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con `TODO` e `FIXME`

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



UNIVERSITÀ DEGLI STUDI FIRENZE

Dove modificare il codice



Esercizio ereditarietà, deep copy e RTTI

- Si considerino le due classi `MyFile` e `Directory`. Modificare il codice del `main()` per far stampare “Directory list” se possibile, altrimenti far stampare “Error”. (3 punti)

- Indicare se una, nessuna o entrambe le classi ha bisogno dell’operatore di assegnazione e di copia profonda (deep copy). Motivare la risposta (2 punti)

Esercizio RAII, exception

- Siano date le classi `HardwareDevice` (che rappresenta un dispositivo hardware) e `DeviceManager` (che gestisce l'hardware consentendo di effettuarci sopra operazioni).

- Si consideri il frammento della funzione test e si riscriva in modo tale da evitare che un'eccezione lanciata prima del delete provochi un leak. Scrivere una soluzione usando la gestione di eccezioni (3 punti) ed una usando un `unique_ptr<>` (3 punti).

- Si modifichi la classe DeviceManager per implementare l'idioma RAII (3 punti). Dare un'implementazione anche con l'uso di `unique_ptr<>` (3 punti).

- Si descriva l'idioma RAII (3 punti)

Esercizio Observer

- Sia data la classe che rappresenta un file system il cui contenuto è composto da file rappresentati dalla struct `DiskFile`. Implementare i metodi di `Filesystem` (4 punti) e quindi scrivere due classi:
  - i) classe `LowDiskSpaceWarning` - scrive la quantità di spazio libero rimasto la prima volta che questa cambia ed è sotto i 10 MB (primo avviso) o quando questo diminuisce rispetto all'ultimo avviso (ed è comunque sotto i 10 MB);
  - ii) classe `FilesystemInfo` – scrive numero di file, spazio occupato e spazio libero, quando il numero di file cambia.
- Implementare il tutto usando il design pattern Observer secondo lo schema pull (12 punti). Disegnare lo schema UML (2 punti).

Esercizio classe template

- Implementare una classe generica che rappresenta coordinate cartesiane bidimensionali. La classe deve avere l'operatore di uguaglianza (4 punti). Implementare anche gli operatori per somma e sottrazione (2 punti).

- La classe ha bisogno dell'operatore di assegnazione e di copia profonda (deep copy)? Motivare la risposta. (1 punto)