



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

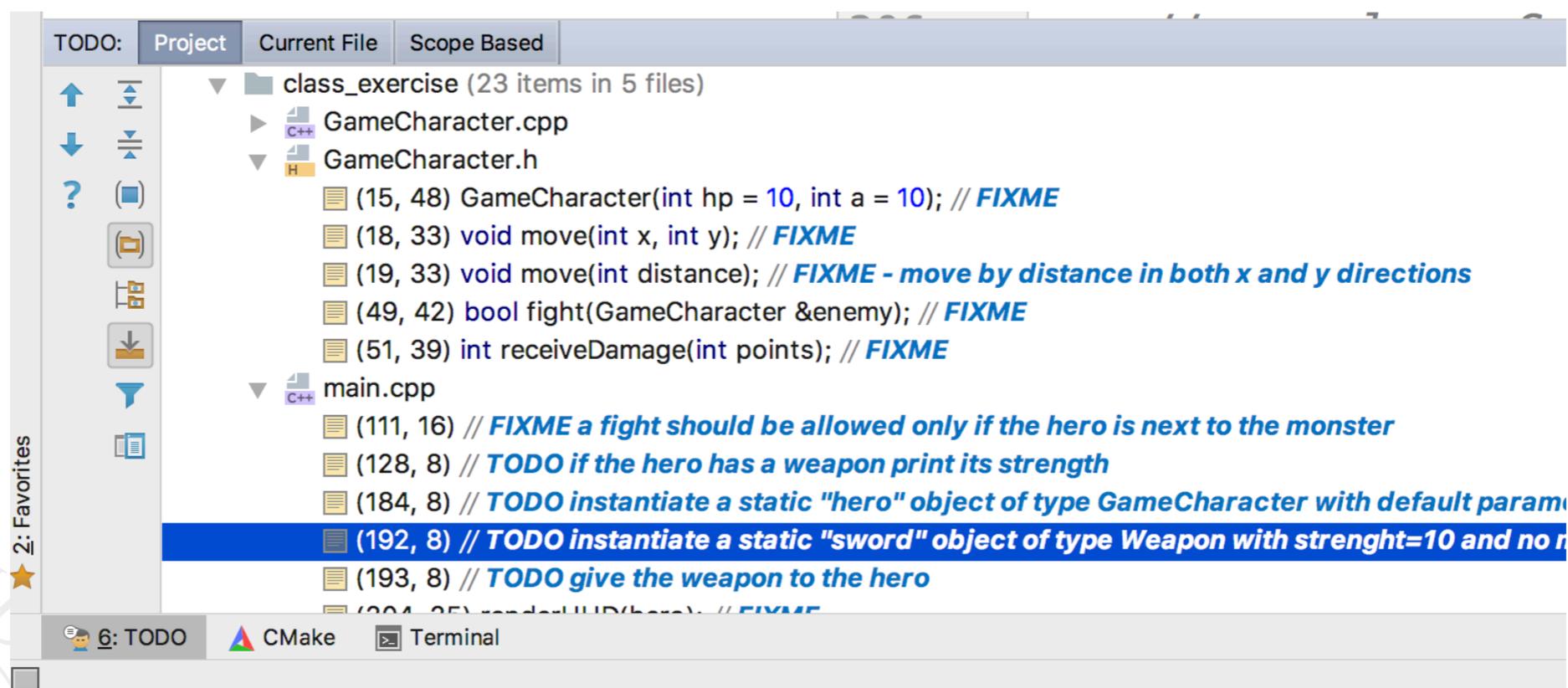


# Schema del codice

- 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



# Dove modificare il codice

The screenshot shows the CLion IDE interface. The main editor displays the code in `main.cpp` with line numbers 185 to 205. The code includes comments and function calls for setting up a hero and an enemy. A yellow highlight is placed on line 192, which contains a TODO comment: `// TODO instantiate a static "sword" object of type Weapon with strenght=10 and`. The bottom panel shows a TODO list with several items, including the one highlighted in the code editor.

```
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);
```

TODO: Project Current File Scope Based

- class\_exercise (23 items in 5 files)
  - GameCharacter.cpp
  - GameCharacter.h
    - (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
  - main.cpp
    - (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

6: TODO CMake Terminal

28:32 LF+ UTF-8+ Context: class\_exercise [D]

# 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)

