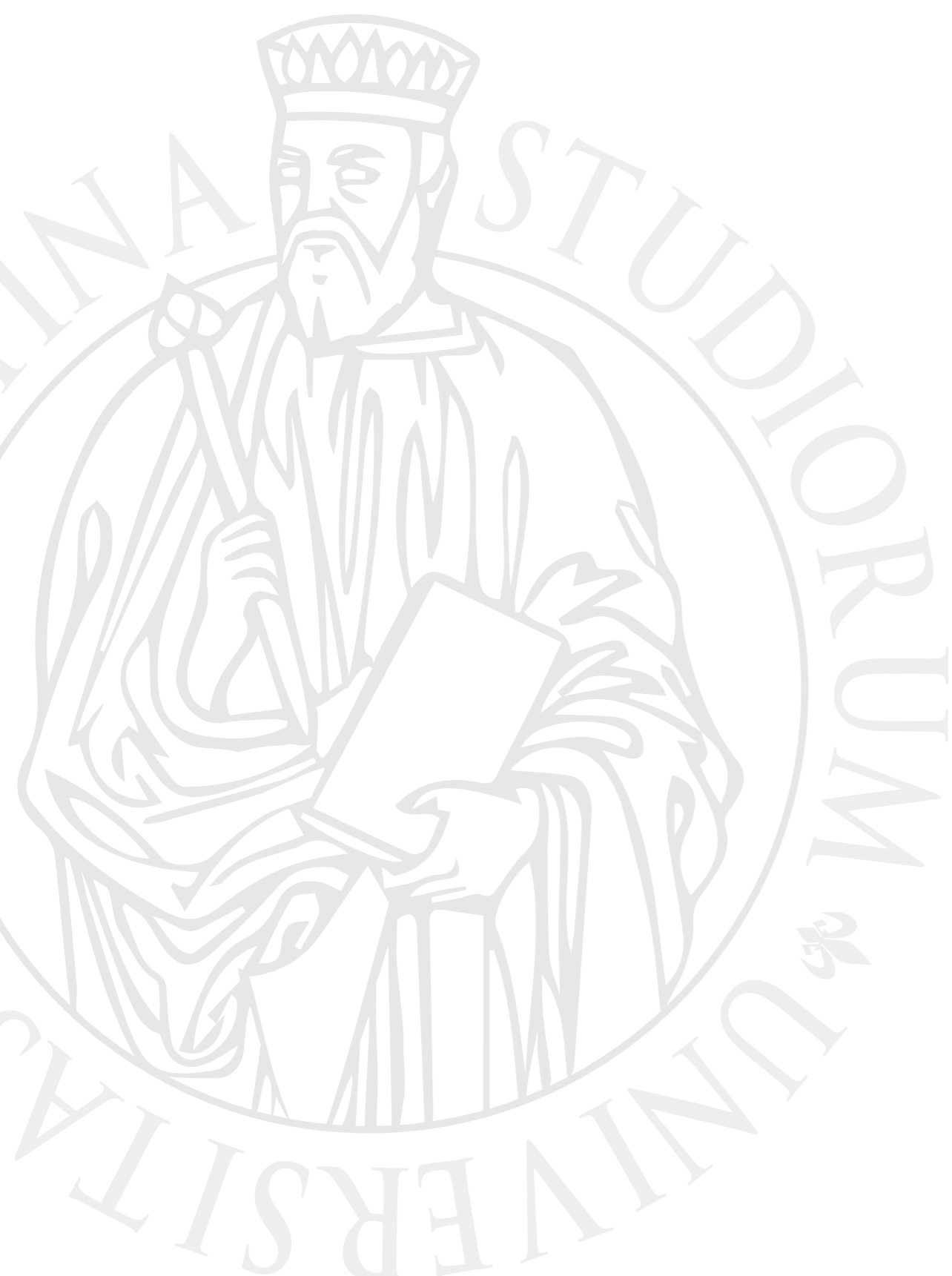




UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

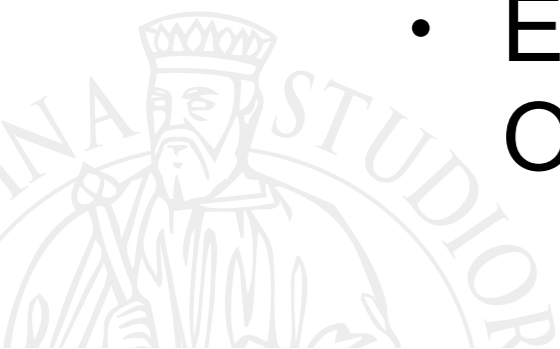


# Esercitazione

Observer exercise

# Obiettivo

- Il progetto CLion fornito contiene classi e scheletri di classi relative ad una componente di rendering grafico di un gioco (mappa e mini-mappa).
- Scopo della presente esercitazione è:
  - Implementare un design pattern Observer nella versione pull
  - Implementare le interfacce Subject e Observer
  - Estendere le classi appropriate per essere Observer e Subject



# Schema del codice

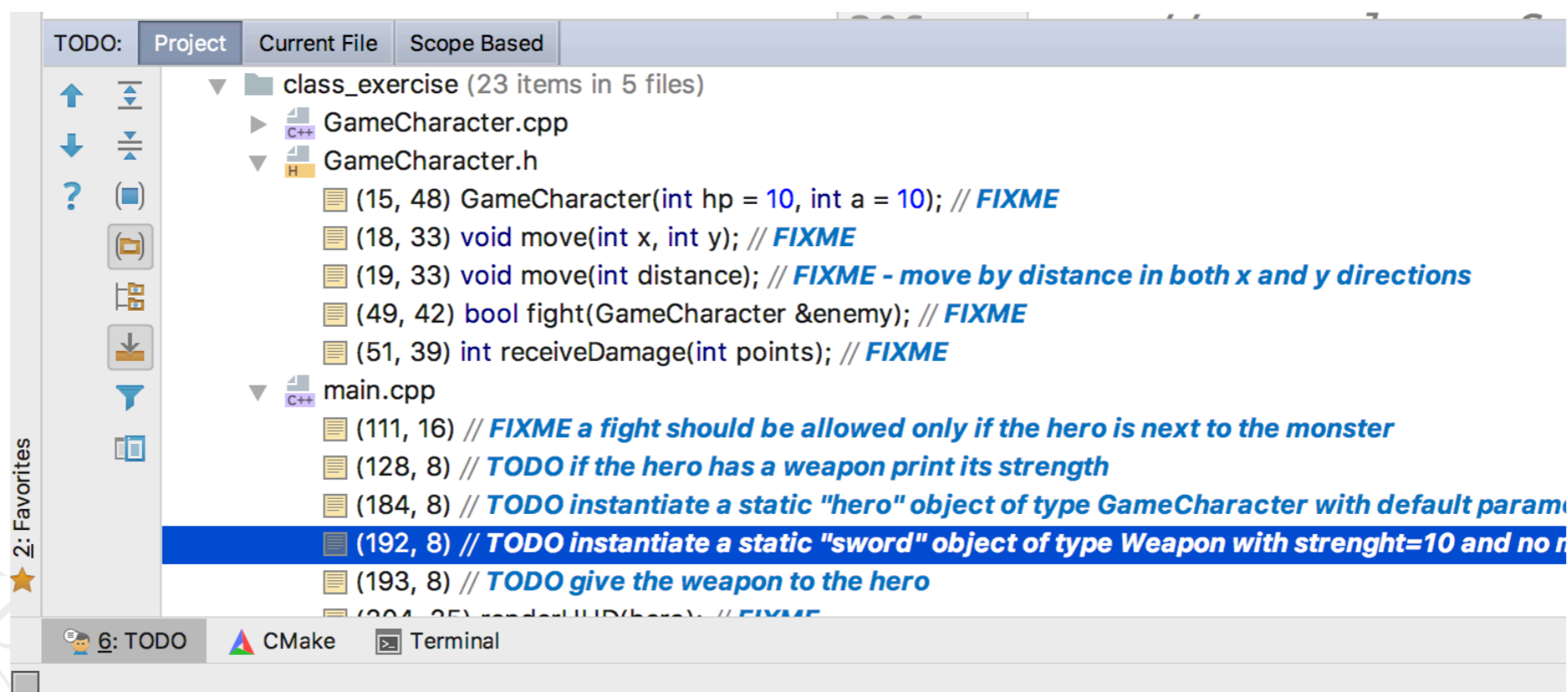
- Il programma è disponibile in due versioni:
  - Terminale - non richiede l'uso di librerie grafiche e tutto l'output è su terminale
  - wxWidgets - richiede l'installazione della libreria wxWidgets, crea una GUI
- Il programma è composto da tre classi principali:
  - GameCharacter rappresenta un personaggio del gioco, in particolare considerando la sua posizione sulla mappa
  - VideogameMapView rappresenta una finestra grafica dove si disegna il personaggio su una vista in alta risoluzione della mappa
  - MiniMapView rappresenta una finestra grafica dove si disegna il personaggio sulla mappa complessiva del gioco

# Schema del codice

- C'è una quarta classe di aiuto - `GameApp` - che rappresenta il gioco con suo game loop e fornisce funzioni per il setup del sistema e l'interazione con l'utente (riceve i comandi da tastiera)
- `Main` con game loop è presente solo nella versione a terminale, dato che in `wxWidgets` il main loop degli eventi è fornito automaticamente dal framework.
- La versione `wxWidgets` quando viene eseguita (in modo normale o debug) richiede che la directory di lavoro sia quella del progetto, per poter leggere i file `.JPG` delle mappe.
- Menu a tendina dell'esecuzione -> `Edit configurations` -> impostazione di working directory

# 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 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 at the bottom of the IDE 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 indicates the current context is `class_exercise [D]`.

# Classe Subject e Observer

- Implementare le interfacce delle due classi.
- Observer deve essere in modalità Pull
- Subject deve fornire metodi per l'abbonamento/disiscrizione
- Observer deve fornire due metodi per interrompere e riprendere la ricezione di notifiche.





# GameCharacter

- Implementare tutte le modifiche necessarie per fare in modo che un cambiamento di stato porti ad un ridisegno della sua posizione sulle mappe





# VideogameMapView e MiniMapView

- Fare tutte le modifiche necessarie perché aggiornino in modo automatico la posizione del giocatore quando si muove
- E' possibile spengere/distruggere la MiniMapView in GameMap, premendo il tasto m
  - Si deve gestire quindi una variazione a runtime degli oggetti da aggiornare
  - Quando si mostra nuovamente una MiniMap precedentemente distrutta/nascosta è bene riaggiornare la vista (o la MiniMap non può disegnare fino al prossimo cambio di stato)