UNIVERSITÀ
DEGLI STUDI
FIRENZE

Esercitazione

Adapter exercise

Obiettivo

- Il progetto CLion fornito contiene classi e scheletri di classi relative ad una componente di rendering grafico di un gioco.

- Scopo della presente esercitazione è:
  - Implementare un design pattern Adapter nelle versioni class ed object
  - Scrivere un’iterazione sul container di oggetti e sfruttare il polimorfismo
  - Riscrivere il codice per passare da raw pointer a smart pointer

Schema del codice

- Il programma è composto da 4 classi di partenza:

- `Shape` rappresenta una forma; è una classe astratta che fornisce l'interfaccia per disegnare e scalare oggetti grafici.

- `Sprite` e `Tile`, derivate da `Shape`, rappresentano elementi grafici del gioco come i personaggi da disegnare e le caselle della mappa.

- `Text` rappresenta del testo, non eredita da `Shape`, non ha il metodo `draw` di `Shape` e il `resize` ha argomenti diversi.

Schema del codice

- In questa esercitazione andremo a creare un `TextShapeAdapter` così da poter inserire scritte nel vettore di oggetti grafici in `main`

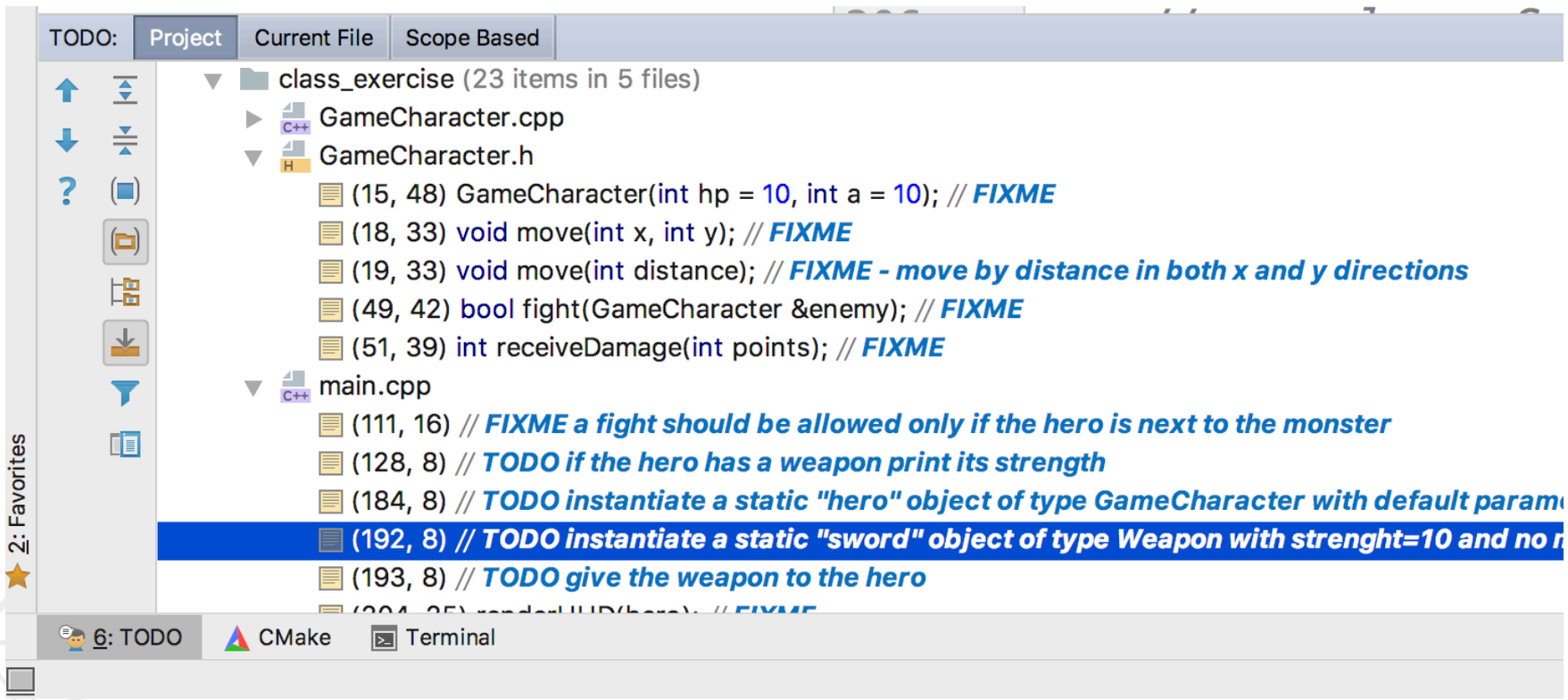
- Scriveremo un'iterazione sul container di oggetti e sfrutteremo il polimorfismo per ridimensionare e disegnare tutti gli oggetti

- Una volta completate le due parti precedenti riscriveremo il codice per usare smart pointer a scelta

Dove modificare il codice

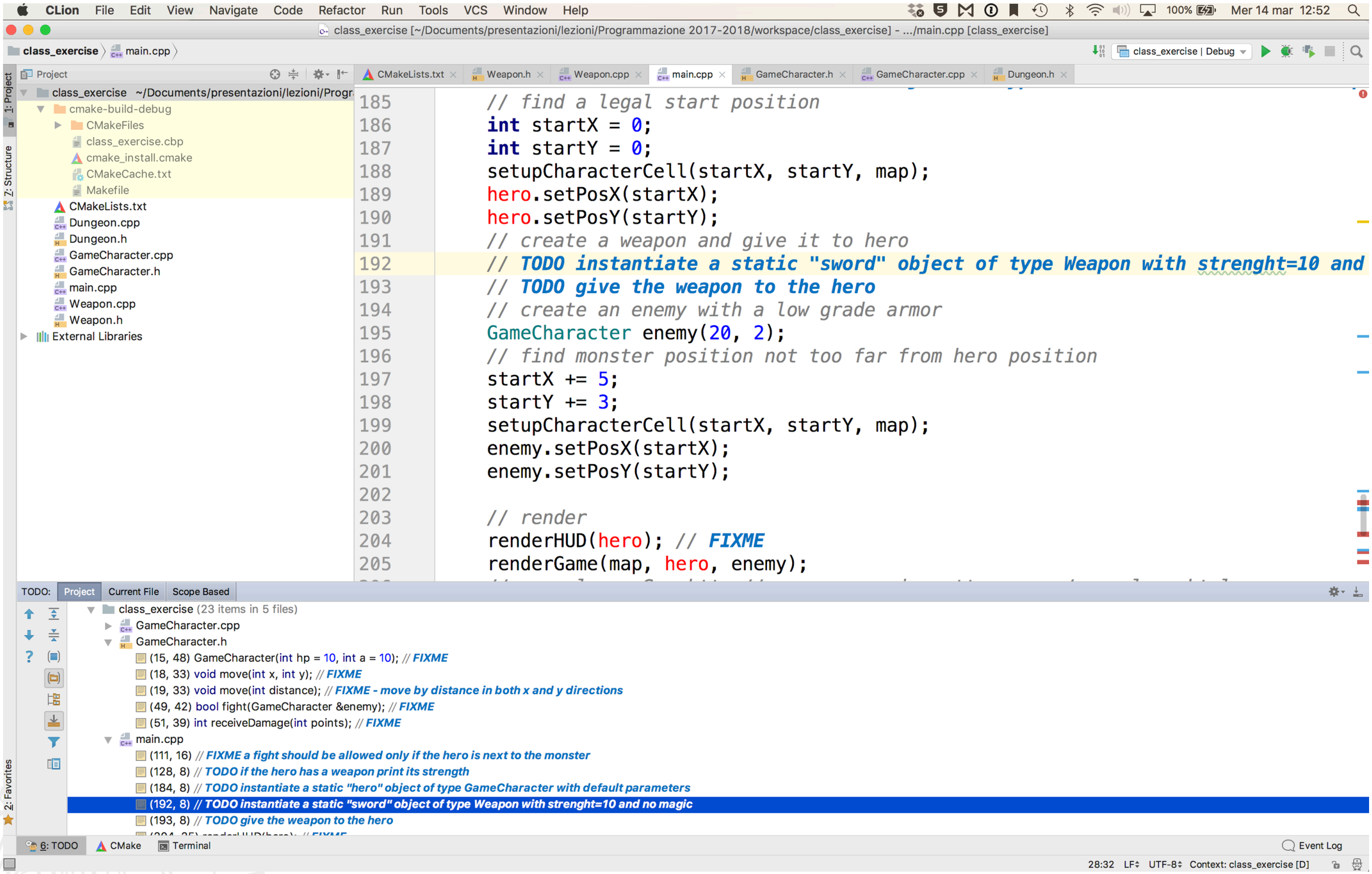
- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con `TODO` e `FIXME`

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



UNIVERSITÀ DEGLI STUDI FIRENZE

Dove modificare il codice



Classe TextShapeAdapter

- Implementare sia un Class che un Object Adapter che consenta di inserire oggetti che racchiudano testo nel vettore di puntatori a shape

Main

- Iterare su tutti gli elementi grafici, ridimensionarli di 1.2 volte e disegnarli.

- Sostituire i raw pointer con smart pointer.
  Trasformare il vettore in vettore di smart pointer

unique_ptr e shared_ptr

- Uno `unique_ptr` non può essere copiato. Per inserirlo in un container STL si può:
  - Usare emplace_back di un'oggetto allocato dinamicamente (verrà gestito da uno unique_ptr creato automaticamente)
  - Usare `push_back` di uno smart pointer creato al momento della chiamata del push_back (ovvero senza nome)
  - Usare il push_back passando uno unique_ptr con nome tramite std::move:

```
std::vector<std::unique_ptr<Obj>> myVector;
std::unique_ptr<Obj> fooPtr(new Obj);
myVector.push_back(std::move(fooPtr));
```

- Uno `shared_ptr` può essere copiato per cui l'inserimento in un container STL non richiede nessun particolare accorgimento.