



UNIVERSITÀ
DEGLI STUDI
FIRENZE



Esercitazione

Smart pointer 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:
 - Ridurre dei memory leak usando smart pointer (unique_ptr e shared_ptr)
 - In particolare si deve trasformare il raw pointer del personaggio del giocatore e la bitmap dello splash screen





The screenshot shows a C++ IDE with a project explorer on the left, a code editor in the center, and a console/Valgrind window at the bottom. The code in the editor is as follows:

```
323 PlayerType playerType = PlayerType::MAGE_KNIGHT;
324 switch (playerType) {
325     case PlayerType::KNIGHT:
326         hero = new Knight("Isildur", 34);
327         break;
328     case PlayerType::WIZARD:
329         hero = new Wizard("Gandalf", 35);
330         break;
331     case PlayerType::MAGE_KNIGHT:
332         hero = new MageKnight("Boromir", 35, 20, 12, 5);
333         break;
334 }
335 // find a legal start position
336 int startX = 0;
337 int startY = 0;
338 findFreeMapTile(startX, startY, map, nullptr);
339 hero->setPosX(startX);
340 hero->setPosY(startY);
341 Vault<Weapon*> armory(startX+1, startY);
342 Sword* aSword1 = new Sword(12, false, true);
343 Bow* aBow1 = new Bow(18, 40, true);
344 Sword* aSword2 = new Sword(15, true, true);
345 armory.setElement(0, aSword1);
346 armory.setElement(1, aBow1);
347 armory.setElement(2, aSword2);
348 if (l1Distance(*hero, armory)<2) {
349     armory.open();
350 }
```

The console/Valgrind window shows the following output:

```
Leak_DefinitelyLost 7 warnings
vg_replace_malloc.c 7 warnings
136 (120 direct, 16 indirect) bytes in 1 blocks are definitely lost in loss record 33 of 50
0x1000CC5F6 malloc vg_replace_malloc.c:302
0x1001B8E0D operator new(unsigned long)
0x100005284 main main.cpp:332
16 bytes in 1 blocks are definitely lost in loss record 5 of 50
16 bytes in 1 blocks are definitely lost in loss record 6 of 50
```

The IDE interface includes a file explorer on the left with files like GameCharacter.h, Knight.cpp, MageKnight.h, and main.cpp. A structure view at the bottom left shows GameEvent and PlayerType enums. The bottom right shows a preview editor with a zoomed-in view of the code line: `hero = new MageKnight("Boromir", 35, 20, 12, 5);`



```
GameCharacter.h
GameFileException.cpp
GameFileException.h

Knight.h
MageKnight.cpp
MageKnight.h
main.cpp
Orc.cpp
Orc.h
Potion.cpp
Potion.h
Skeleton.cpp
Skeleton.h
splash_screen.cpp
splash_screen.h

Structure
loadBitmap(string) : const FakeJPEG *
displaySplash() : void

323 PlayerType playerType = PlayerType::MAGE_KNIGHT;
324 switch (playerType) {
325     case PlayerType::KNIGHT:
5 const FakeJPEG* loadBitmap(std::string fileName) {
6     FakeJPEG* result = new FakeJPEG();
7     result->load(fileName);
8     return result;
9 }
10
11 void displaySplash() {
12     // ... start graphics
13
14     try {
15         const FakeJPEG* splashScreen = loadBitmap(SPLASH_SCREEN_FILENAME); // XXX should work
16         //const FakeJPEG* splashScreen = loadBitmap("non_existent_file.fake_jpg"); // XXX tes
17         // const FakeJPEG* splashScreen = loadBitmap("./res/splash_screen_corrupted.fake_jpg"
18         // ... display bitmap
19         auto bitmap = splashScreen->getBitmap();
20         for (auto bitmapLine : bitmap)
21             std::cout << bitmapLine << std::endl;
22     } catch (GameFileException &e) {
23         std::cerr << e.what() << std::endl;
24         e.printStackTrace();
25         if(e.isFatal())
26             abort();
27     }
}
```

Run: Build All x

Console Valgrind

- ▶ 24 bytes in 1 blocks are definitely lost in loss record 11 of 50
- ▶ 48 bytes in 1 blocks are definitely lost in loss record 22 of 50
- ▼ 632 (32 direct, 600 indirect) bytes in 1 blocks are definitely lost in loss record 44 of 50
 - 0x1000CC5F6 malloc vg_replace_malloc.c:302
 - 0x1001B8E0D operator new(unsigned long)
 - 0x100016C0A loadBitmap(std::__1::basic_string<char, std::__1::char_traits, std::__1::allocator>) splash_screen.c
 - 0x100016D7E displaySplash() splash_screen.cpp:15
 - 0x1000048A9 main main.cpp:294

Frame Information Preview Editor →

```
const FakeJPEG* loadBitmap(std::string fileName) {
    FakeJPEG* result = new FakeJPEG();
    result->load(fileName);
    return result;
}
```

16 bytes in 1 blocks are definitely lost in loss record 5 of 50

16 bytes in 1 blocks are definitely lost in loss record 6 of 50

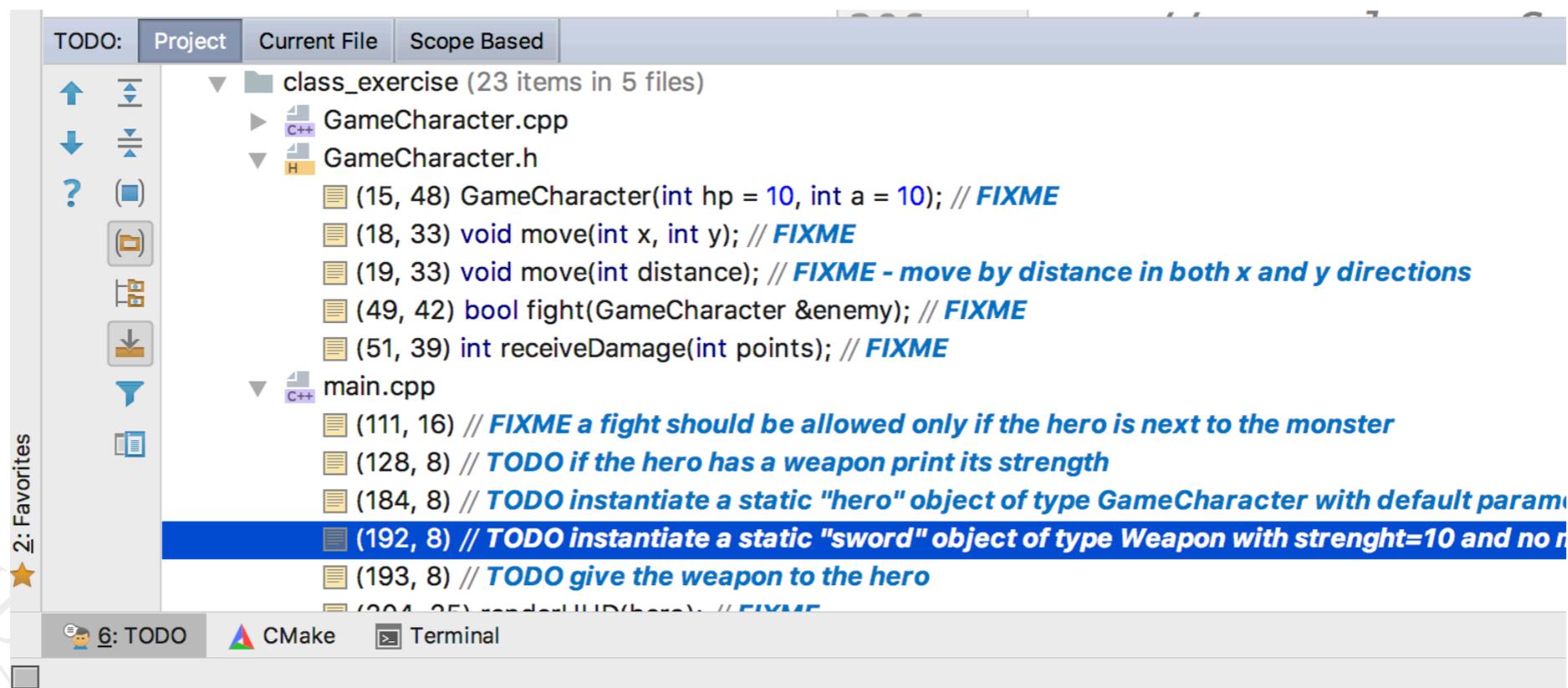
```
new MageKnight( ... , 50
break;
```

Schema del codice

- Lo schema del codice delle classi di base è lo stesso di quello della volta scorsa. In particolare opereremo su:
- `Main` nel file andremo a modificare la creazione dell'oggetto da classi derivate da `GameCharacter` (`hero`)
- `splash_screen` modificando la funzione che legge la bitmap.

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 C++ 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 IDE interface includes a Project Structure view on the left, a TODO list at the bottom, and a status bar at the very bottom.

TODO List:

- (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

Status Bar: 28:32 LF+ UTF-8+ Context: class_exercise [D]



main/splash_screen

- Decidere quali smart pointer usare nei due casi e sostituirli agli attuali raw pointers



unique_ptr e shared_ptr

- Uno `unique_ptr` o `shared_ptr` dichiarato ma non inizializzato equivale ad un `nullptr`
- `unique_ptr` non può essere copiato, per cui non si può creare uno `unique_ptr` non inizializzato per poi assegnargli il valore di un altro `unique_ptr`

- ... a meno che si usi `std::move()`:

```
std::unique_ptr<MyClass> test;  
std::unique_ptr<MyClass> test2(new MyClass);  
test = std::move(test2);
```

- ... o si assegni uno `unique_ptr` senza nome

```
std::unique_ptr<MyClass> test;  
test = std::unique_ptr<MyClass>(new MyClass);
```

- Tutto questo si capirà meglio nell'ultima lezione in cui si tratteranno temi avanzati del C++11 (move semantics)