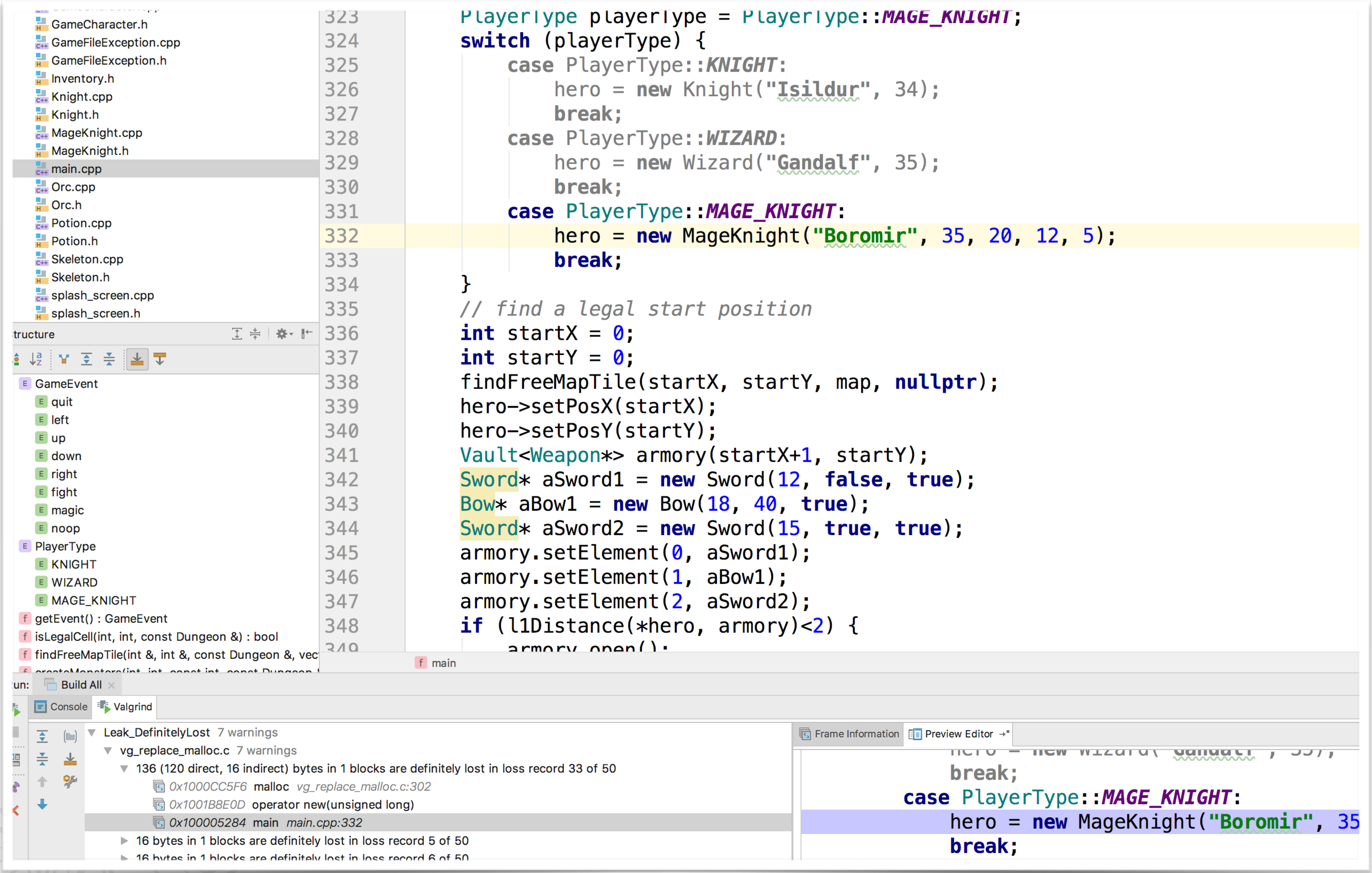
Esercitazione

Smart pointer exercise

Obiettivo

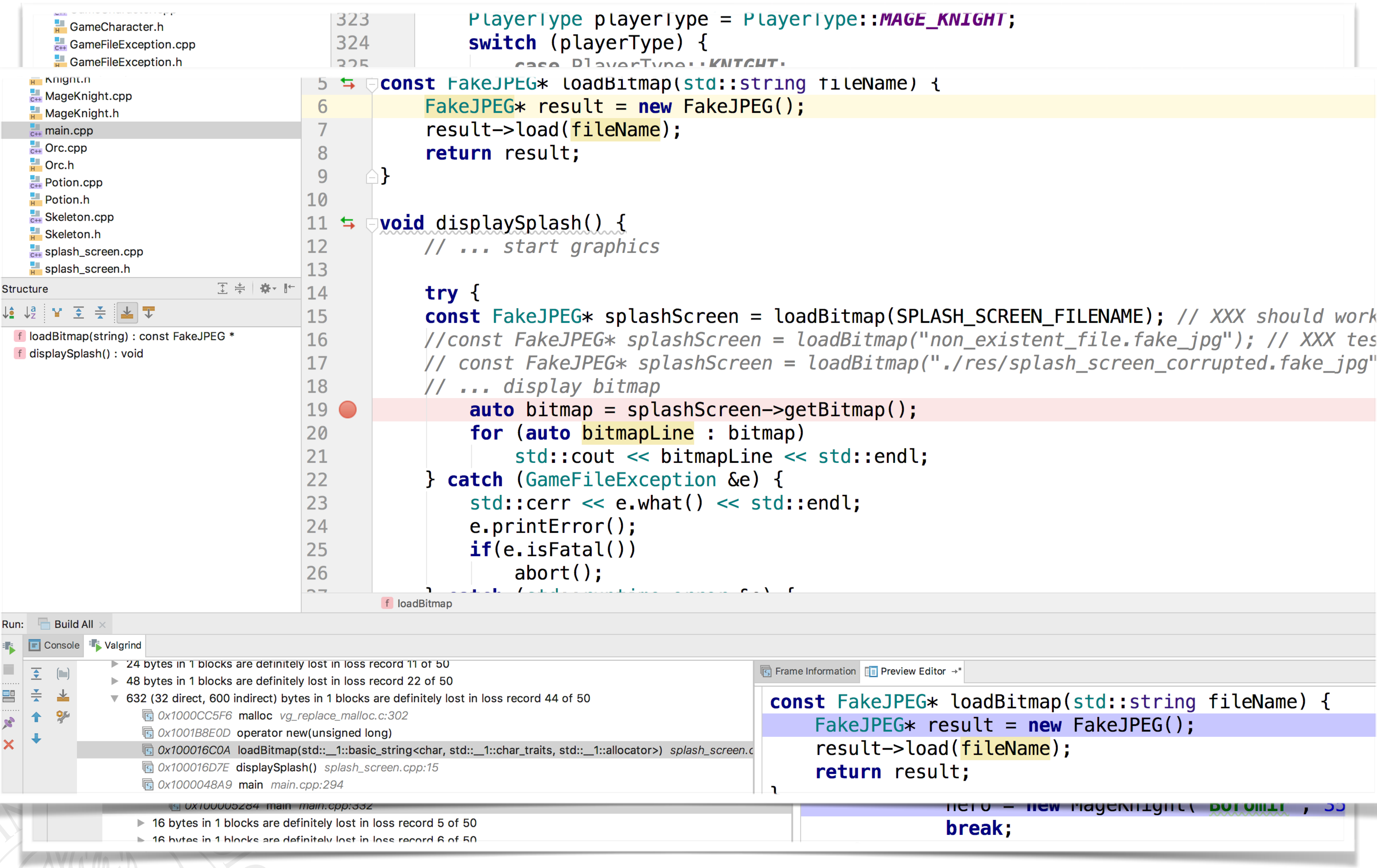
- Il progetto CLion fornito contiene classi e scheletri di classi relative al gioco in stile Rogue (https://it.wikipedia.org/wiki/Roguelike) della scorsa esercitazione.
- Scopo della presente esercitazione è:
  - Ridurre dei memory leak usando smart pointer (unique_ptr e shared_ptr)
    - In particolare si deve trasformare il raw pointer del personaggio del giocatore e la bitmap dello splash screen



```
    PlayerType playerType = PlayerType::MAGE_KNIGHT;
    switch (playerType) {
        case PlayerType::KNIGHT:
            hero = new Knight("Isildur", 34);
            break;
        case PlayerType::WIZARD:
            hero = new Wizard("Gandalf", 35);
            break;
        case PlayerType::MAGE_KNIGHT:
            hero = new MageKnight("Boromir", 35, 20, 12, 5);
            break;
    }
    // find a legal start position
    int startX = 0;
    int startY = 0;
    findFreeMapTile(startX, startY, map, nullptr);
    hero->setPosX(startX);
    hero->setPosY(startY);
    Vault<Weapon*> armory(startX+1, startY);
    Sword* aSword1 = new Sword(12, false, true);
    Bow* aBow1 = new Bow(18, 40, true);
    Sword* aSword2 = new Sword(15, true, true);
    armory.setElement(0, aSword1);
    armory.setElement(1, aBow1);
    armory.setElement(2, aSword2);
    if (l1Distance(*hero, armory)<2) {
        armory.open();
```

```
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
```

```
            hero = new Wizard("Gandalf", 35);
            break;
        case PlayerType::MAGE_KNIGHT:
            hero = new MageKnight("Boromir", 35
            break;
```



```
    PlayerType playerType = PlayerType::MAGE_KNIGHT;
    switch (playerType) {
        case PlayerType::KNIGHT:
```

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

void displaySplash() {
    // ... start graphics

    try {
    const FakeJPEG* splashScreen = loadBitmap(SPLASH_SCREEN_FILENAME); // XXX should work
    //const FakeJPEG* splashScreen = loadBitmap("non_existent_file.fake_jpg"); // XXX tes
    // const FakeJPEG* splashScreen = loadBitmap("./res/splash_screen_corrupted.fake_jpg"
    // ... display bitmap
        auto bitmap = splashScreen->getBitmap();
        for (auto bitmapLine : bitmap)
            std::cout << bitmapLine << std::endl;
    } catch (GameFileException &e) {
        std::cerr << e.what() << std::endl;
        e.printError();
        if(e.isFatal())
            abort();
    } catch (std::runtime_error &e) {
```

```
▶ 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
```

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

```
      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
```

```
            hero = new MageKnight("Bot0mir", 35
            break;
```

Schema del codice

- Lo schema del codice delle classi di base è lo stesso di quello della volta scorsa. In particolare opereremo su:

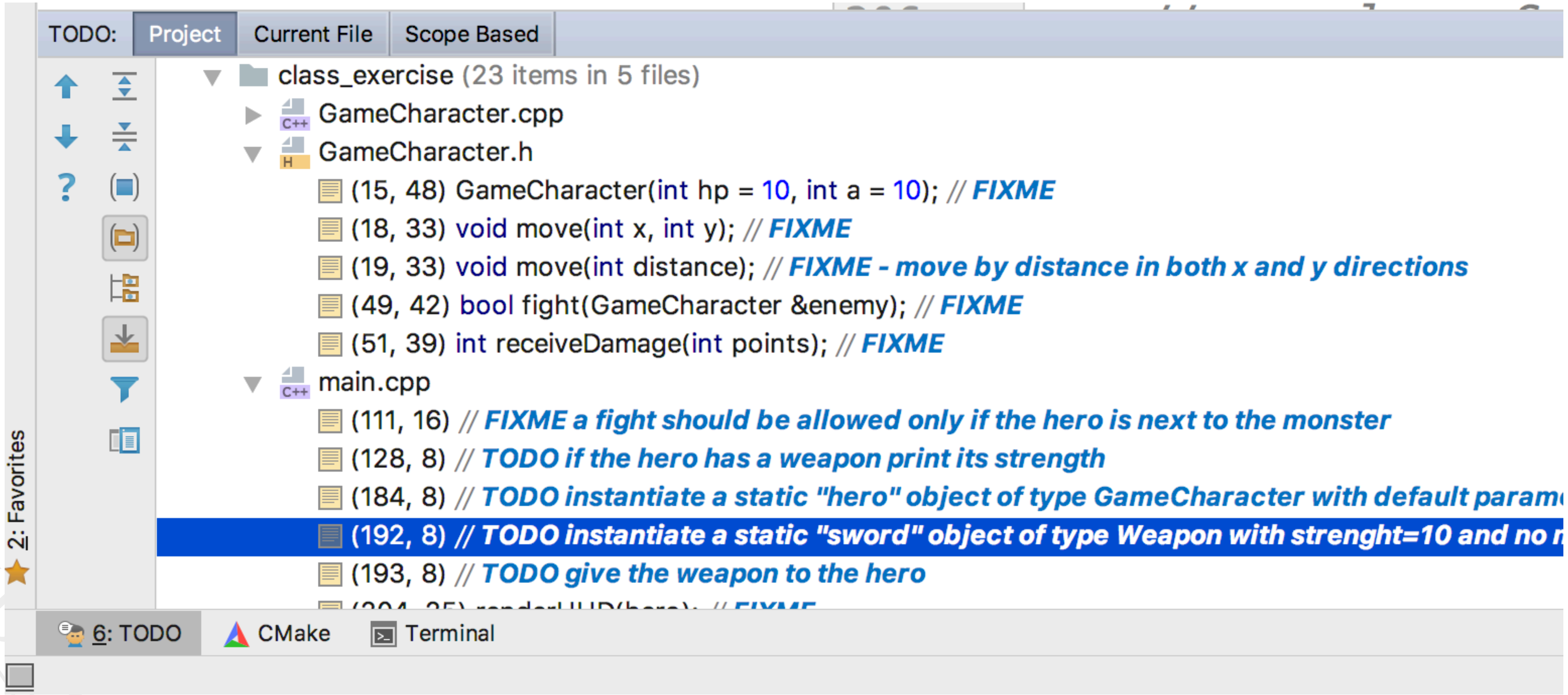
- `Main` nel file andremo a modificare la creazione dell’oggetto da classi derivate da `GameCharacter` `(hero)`

- `splash_screen` modificando la funzione che legge la bitmap.

Dove modificare il codice

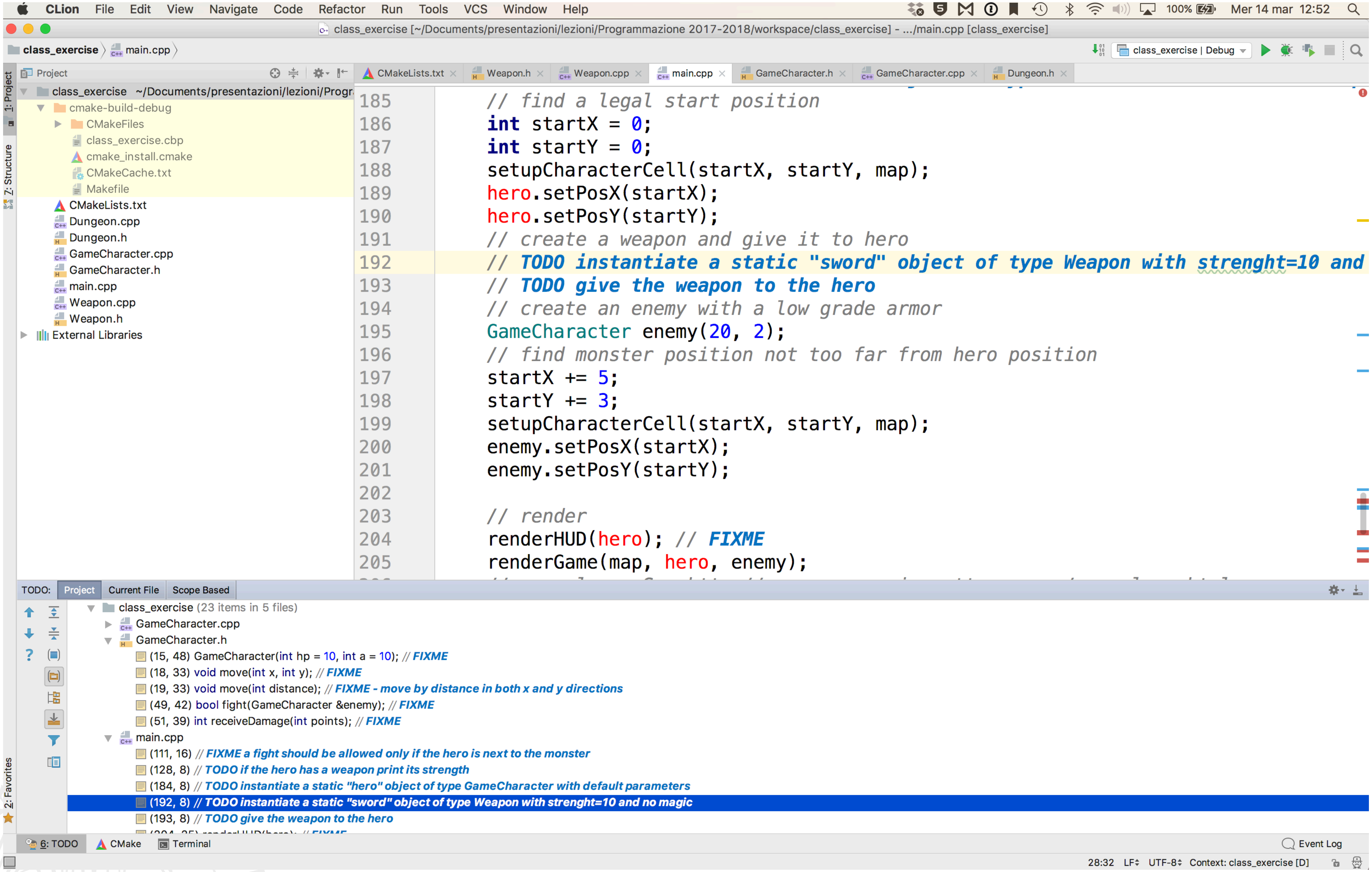
- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con `TODO` e `FIXME`

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



```
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 param
    (192, 8) // TODO instantiate a static "sword" object of type Weapon with strenght=10 and no r
    (193, 8) // TODO give the weapon to the hero
    (204, 25) renderHUD(hero); // FIXME
```

Dove modificare il codice



```
    // find a legal start position
    int startX = 0;
    int startY = 0;
    setupCharacterCell(startX, startY, map);
    hero.setPosX(startX);
    hero.setPosY(startY);
    // create a weapon and give it to hero
    // TODO instantiate a static "sword" object of type Weapon with strenght=10 and
    // TODO give the weapon to the hero
    // create an enemy with a low grade armor
    GameCharacter enemy(20, 2);
    // find monster position not too far from hero position
    startX += 5;
    startY += 3;
    setupCharacterCell(startX, startY, map);
    enemy.setPosX(startX);
    enemy.setPosY(startY);

    // render
    renderHUD(hero); // FIXME
    renderGame(map, hero, enemy);
```

```
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
```

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)