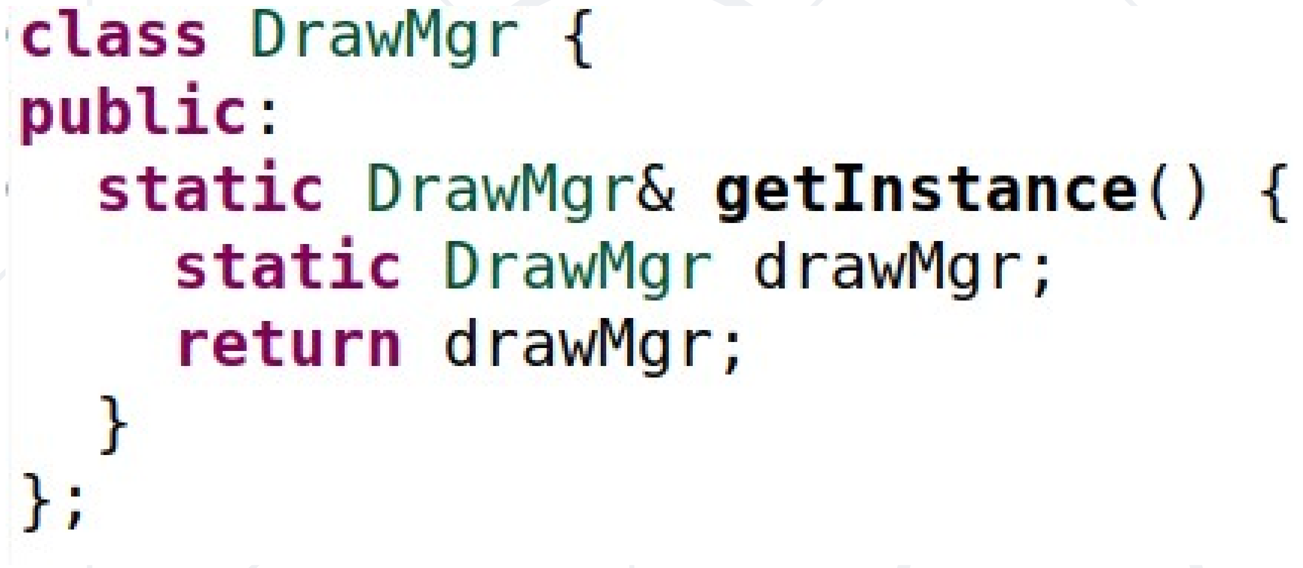
Singleton



```
class DrawMgr {
public:
  static DrawMgr& getInstance() {
    static DrawMgr drawMgr;
    return drawMgr;
  }
};
```

Zhivko Petrov

A guy that knows C++

Software University

SoftUni

Software University

https://about.softuni.bg

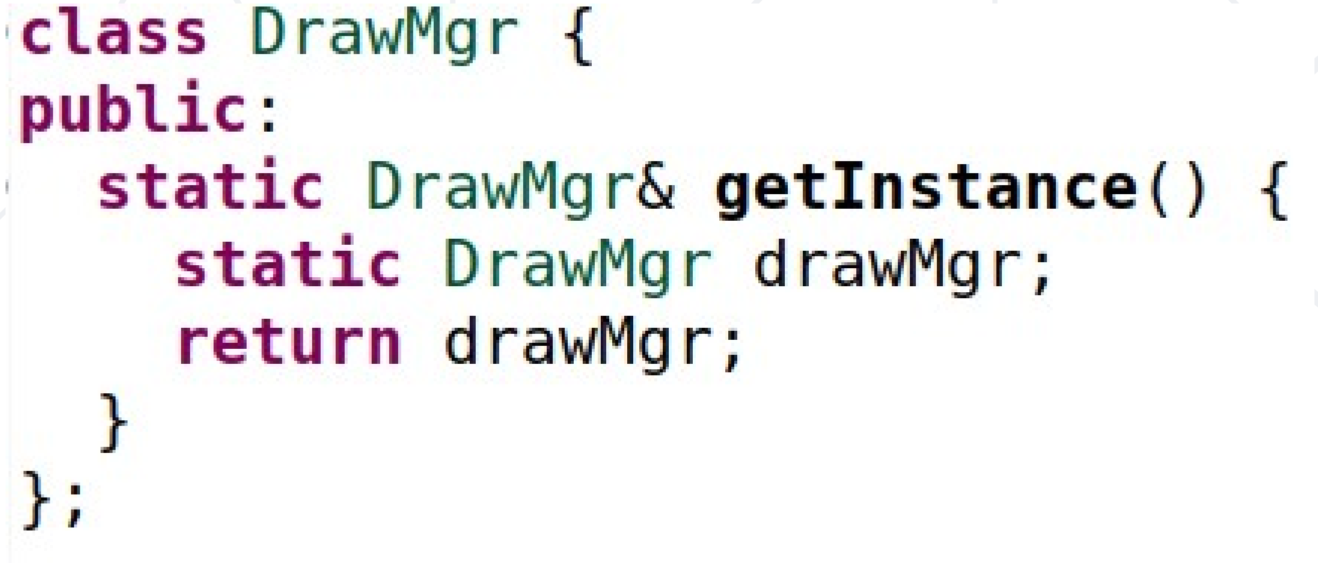
Have a Question?

sli.do

#app-dev-cpp

Singleton concept

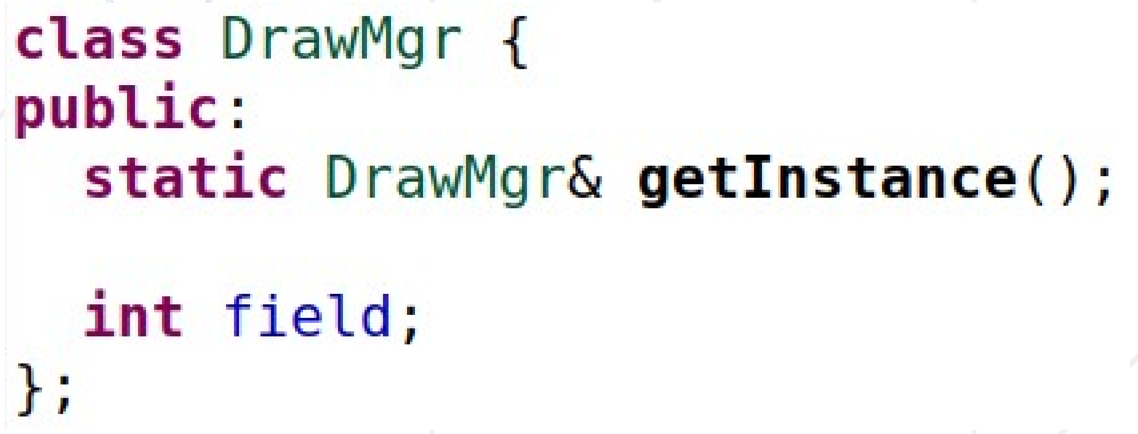
- The Singleton is one of the most famous design pattern
- Singletons control a **unique resource** or have **unique control** over some piece of code
- They **enforce** the creation of only a **single object** for a specific class



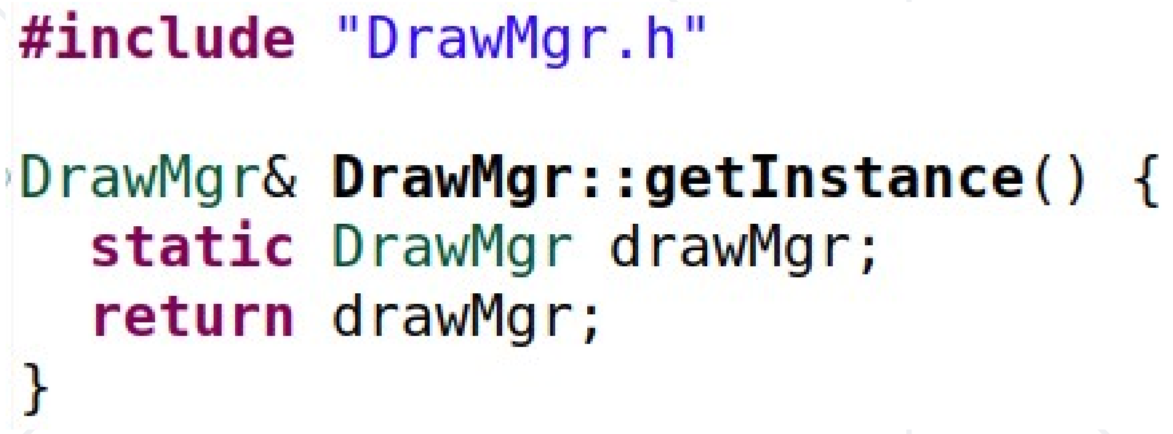
```
class DrawMgr {
public:
  static DrawMgr& getInstance() {
    static DrawMgr drawMgr;
    return drawMgr;
  }
};
```

Meyer’s Singleton concept

- The lifetime of function static variables begins the **first time** the program flow encounters the declaration and ends at program termination
- If control enters the **declaration concurrently** while the variable is being initialized, the concurrent **execution shall wait** for completion of the initialization



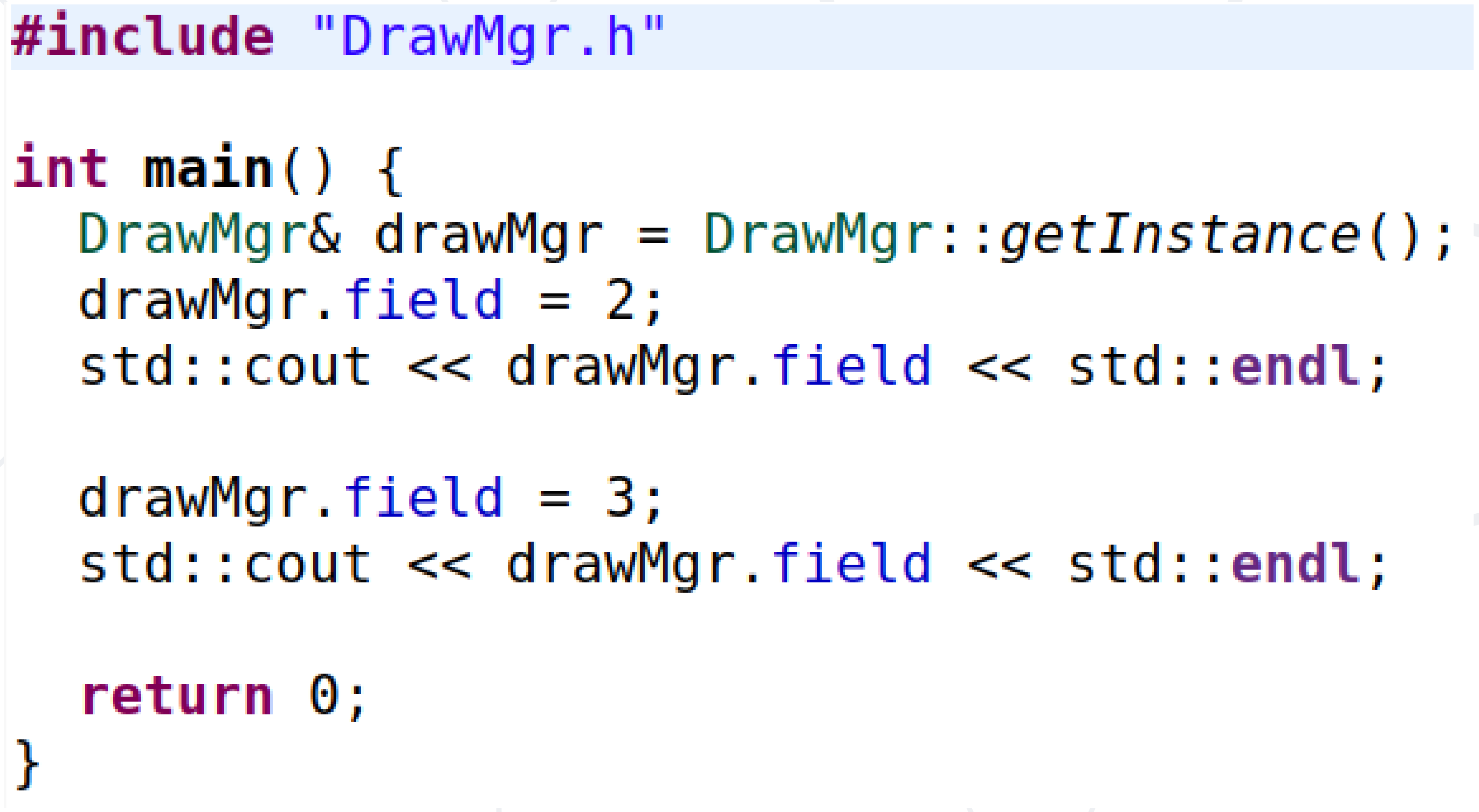
```cpp
class DrawMgr {
public:
  static DrawMgr& getInstance();

  int field;
};
```



```cpp
#include "DrawMgr.h"

DrawMgr& DrawMgr::getInstance() {
  static DrawMgr drawMgr;
  return drawMgr;
}
```

Singleton usage

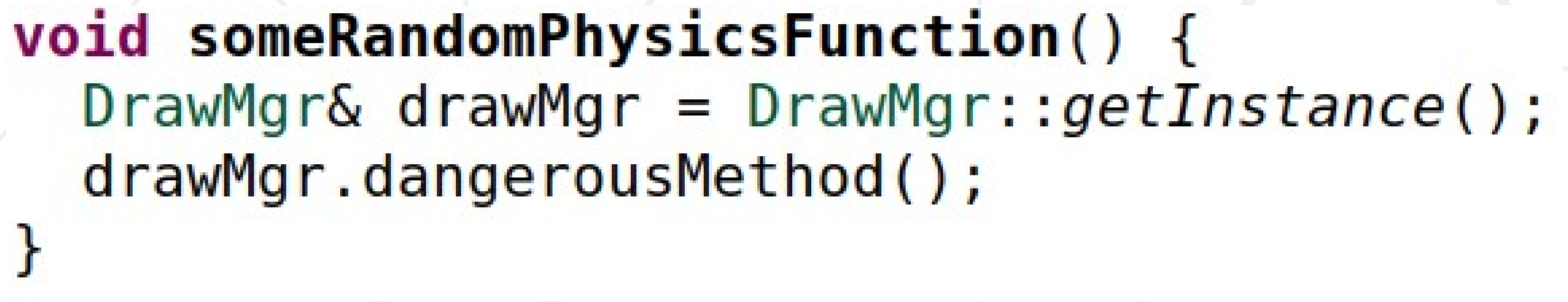
- Singletons are only used through the special **getter fuction**
- Used as normal global variable
- Can be accessed form anywhere



```cpp
#include "DrawMgr.h"

int main() {
  DrawMgr& drawMgr = DrawMgr::getInstance();
  drawMgr.field = 2;
  std::cout << drawMgr.field << std::endl;

  drawMgr.field = 3;
  std::cout << drawMgr.field << std::endl;

  return 0;
}
```

Singleton pitfalls

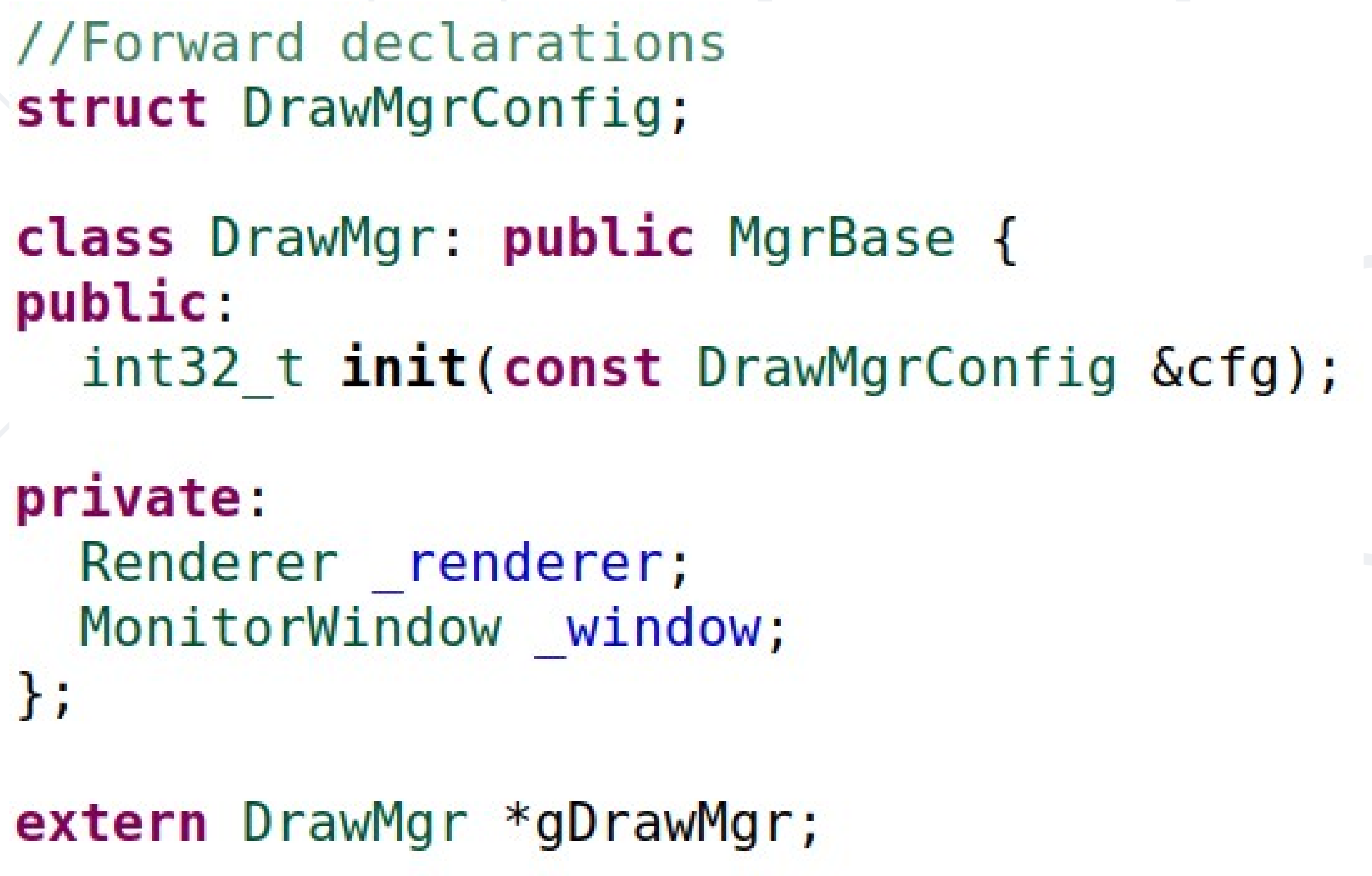
- Although **convenient** to use – Singletons has **one big pitfall**
- They can easily lead to “spaghetti code” **breaking encapsulation**
- Imagine this scenario
- What do Physics has to be with Rendering?
- Why should Physics **access** Rendering sub-system?



```
void someRandomPhysicsFunction() {
  DrawMgr& drawMgr = DrawMgr::getInstance();
  drawMgr.dangerousMethod();
}
```

Singleton correct usage

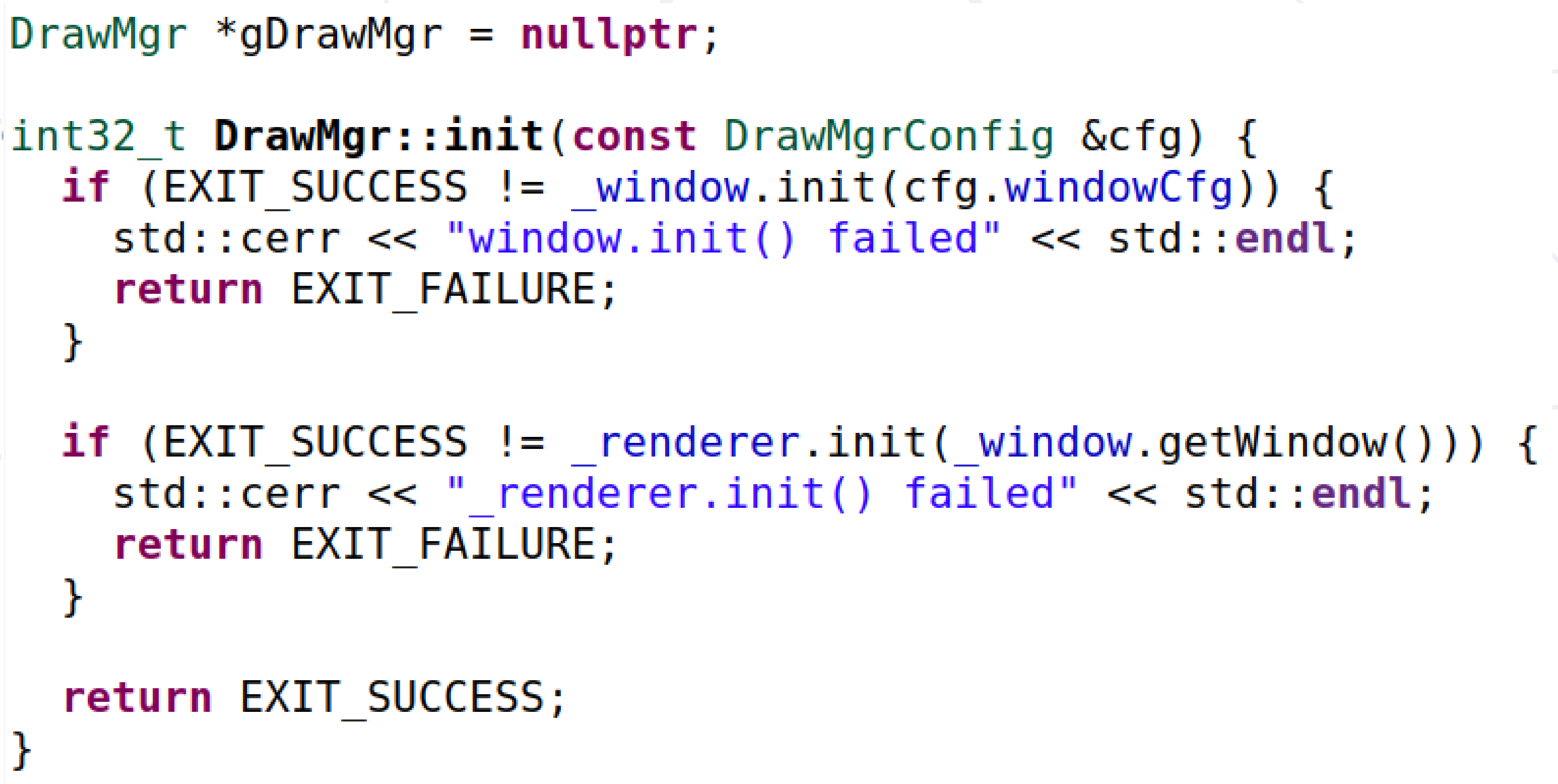
- To avoid the Singletons pitfalls some certain rules must be followed
- **Remove** the static function – it is both verbose and expensive
- Use **Abstraction** and **Encapsulation** as much as possible



```
//Forward declarations
struct DrawMgrConfig;

class DrawMgr: public MgrBase {
public:
  int32_t init(const DrawMgrConfig &cfg);

private:
  Renderer _renderer;
  MonitorWindow _window;
};

extern DrawMgr *gDrawMgr;
```

Singleton correct usage

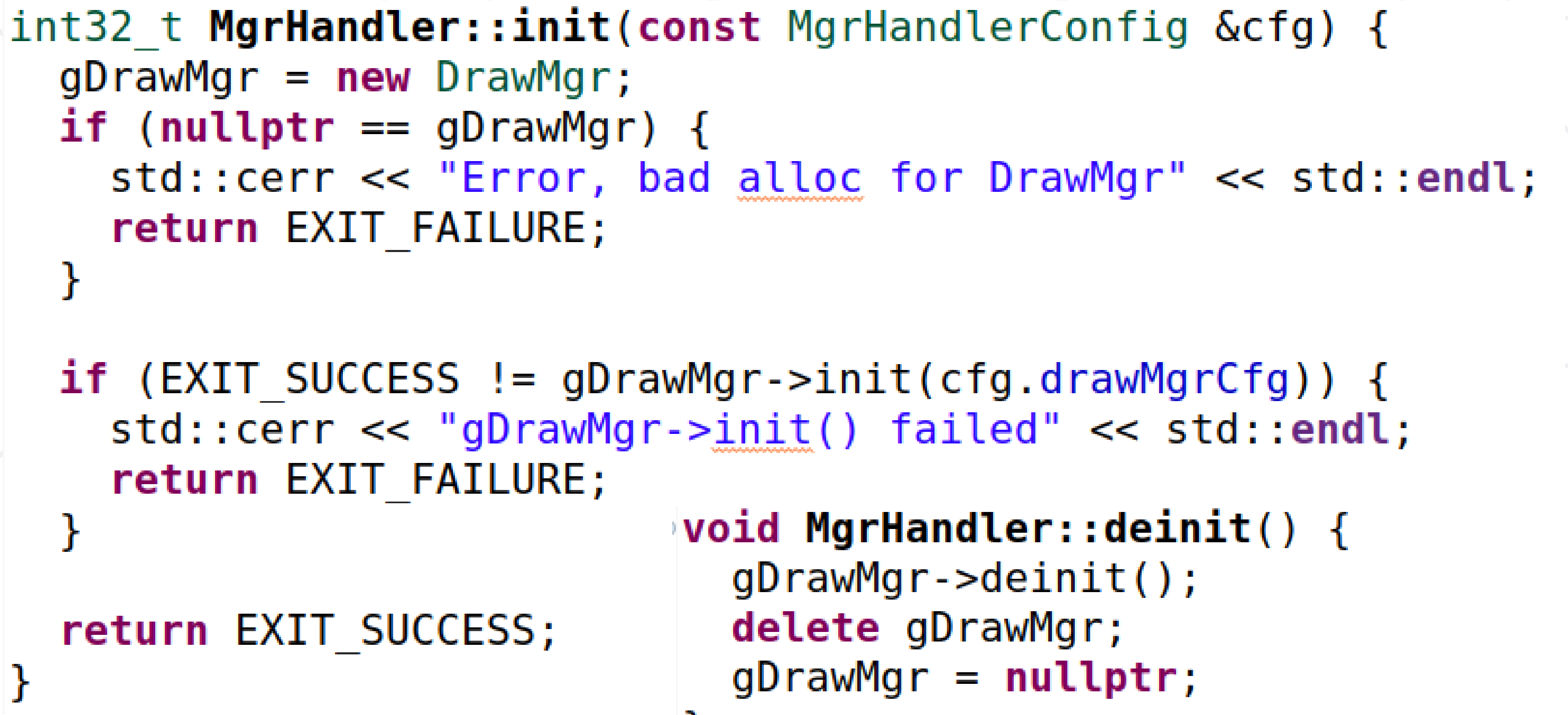
- Implement your class functionalities as normal
- Have a **single** global object



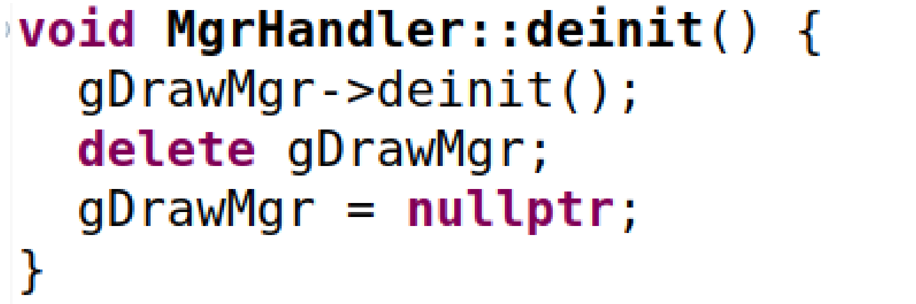
```
DrawMgr *gDrawMgr = nullptr;

int32_t DrawMgr::init(const DrawMgrConfig &cfg) {
  if (EXIT_SUCCESS != _window.init(cfg.windowCfg)) {
    std::cerr << "window.init() failed" << std::endl;
    return EXIT_FAILURE;
  }

  if (EXIT_SUCCESS != _renderer.init(_window.getWindow())) {
    std::cerr << "_renderer.init() failed" << std::endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

Singleton creation/destruction

- Ensure your objects gets created and initialized/deinitilized only on a **single place** in the code
- Usually this is done in the **core** system init/deinit methods



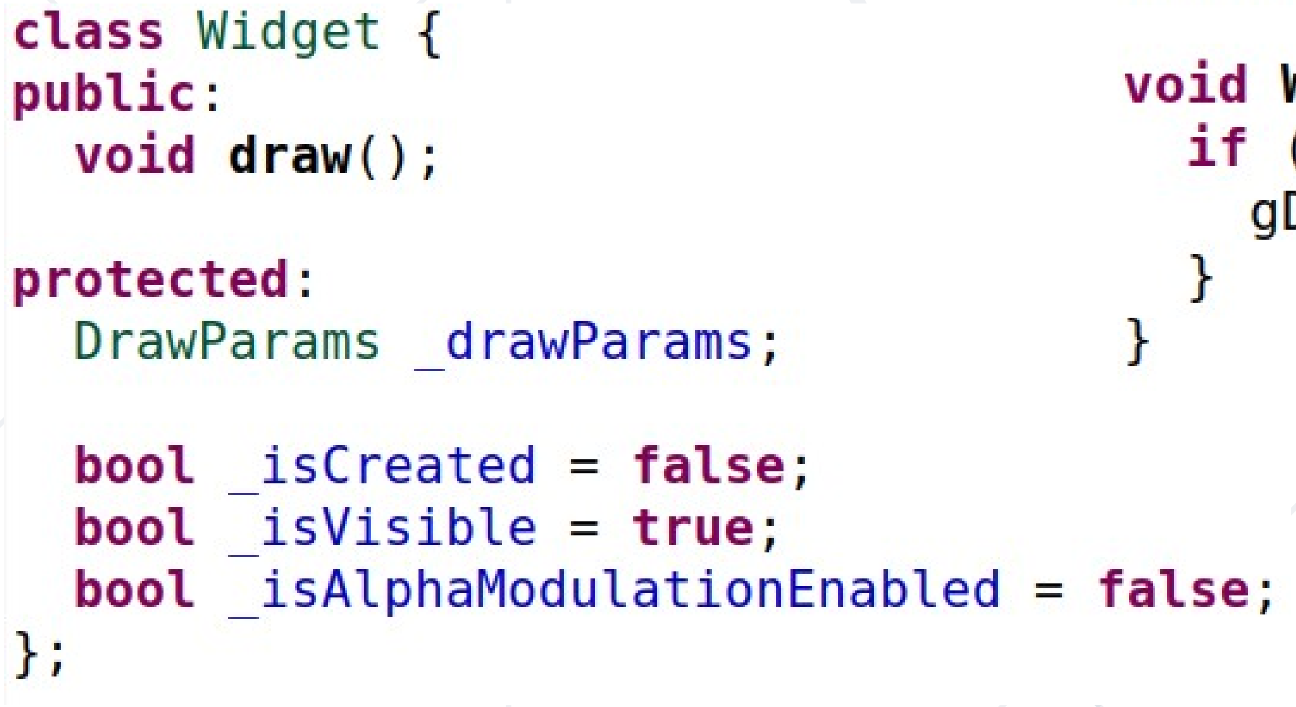
```cpp
int32_t MgrHandler::init(const MgrHandlerConfig &cfg) {
  gDrawMgr = new DrawMgr;
  if (nullptr == gDrawMgr) {
    std::cerr << "Error, bad alloc for DrawMgr" << std::endl;
    return EXIT_FAILURE;
  }

  if (EXIT_SUCCESS != gDrawMgr->init(cfg.drawMgrCfg)) {
    std::cerr << "gDrawMgr->init() failed" << std::endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```



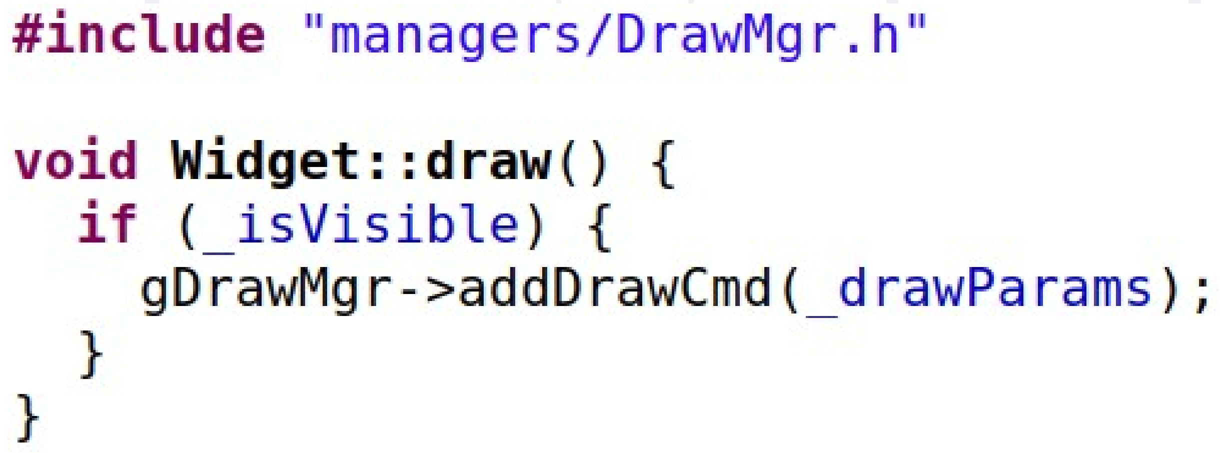
```cpp
void MgrHandler::deinit() {
  gDrawMgr->deinit();
  delete gDrawMgr;
  gDrawMgr = nullptr;
}
```

Singleton – restriction

- It is important to **restrict** the developer in “blindly” using the Singleton
- Achieved by **hiding** the usage of the Singleton in other API calls



```
class Widget {
public:
  void draw();

protected:
  DrawParams _drawParams;

  bool _isCreated = false;
  bool _isVisible = true;
  bool _isAlphaModulationEnabled = false;
};
```



```
#include "managers/DrawMgr.h"

void Widget::draw() {
  if (_isVisible) {
    gDrawMgr->addDrawCmd(_drawParams);
  }
}
```

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Diamond Partners

Software University

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

SmartIT

XS SOFTWARE

zühlke
empowering ideas

INFRAGISTICS®

Coca-Cola HBC
Bulgaria

DRAFT KINGS

Postbank
Решения за твоето утре

SOFTWARE GROUP

Educational Partners

eee.bg

VIRTUAL RACING SCHOOL

codexio

License

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg