SINGLETON PATTERN

ENSURE A CLASS HAS **ONE INSTANCE**

AND PROVIDE A **GLOBAL POINT TO ACCESS IT**
Singleton

- static Instance()
- SingletonOperation()
- GetSingletonData()

```
static uniqueInstance
singletonData
```

return uniqueInstance
THE GOOD

Easily accessible

Easy to implement

Lazy

Can be subclassed
THE GOOD

Can make use of config files

Can depend on other singletons
GLOBAL VARIABLE PROBLEMS

- harder to understand the code
- encourages coupling
- not concurrency-friendly
THE BAD

BREAKS THE

SINGLE RESPONSIBILITY PRINCIPLE
THE BAD

DETRIMENTAL TO UNIT TESTING

- Holds state for as long as the program lasts
- Unable to replace singletons with mock implementations.
- Methods that access singleton cannot be effectively unit tested
WHERE TO USE

**GENERALLY:**
Systems that bind a single available resource.

**IN GAMES:**
Manager objects that control a widely accessed system.

*Although a service locator is usually better option. If possible, it is better not to have such systems at all.*
HOMEWORK: 29.03 – 12.04

- **Read Singleton chapter** from Game Programming Patterns
  http://www.gameprogrammingpatterns.com/singleton.html

- **Read Singleton chapter** from CGLearn
  https://cglearn.eu/student/materials

- **Do the Singleton task** from CGLearn