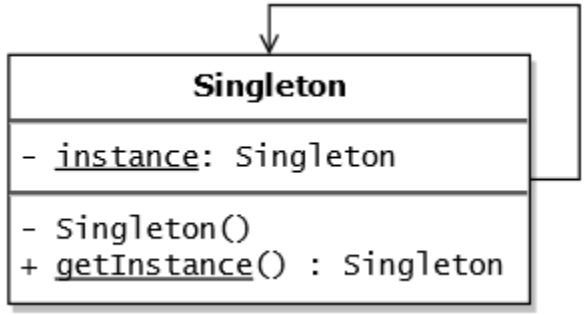
On using singletons in C++

Arno Lepisk

arno@lepisk.se / arno.lepisk@hiq.se

Version 1.1



Implementation methods

Pointer

```
class Singleton {
  Singleton() = default;
public:
  static Singleton * instance();
  void foo();
};
```

```
Singleton * Singleton::instance() {
    static Singleton * inst = nullptr;
    if(!inst) inst = new Singleton();
    return inst;
  }
```

Implementation methods

Pointer

Reference

```
class Singleton {
  Singleton() = default;
public:
  static Singleton & instance();
  void foo();
};
```

```
Singleton & Singleton::instance() {
    static Singleton inst;
    return inst;
  }
```

Implementation methods

Pointer

```
Singleton::instance()->foo();
```

Reference

```
Singleton::instance().foo();
```

Usage

Hiding implementation details

PIMPL

```
class Singleton {
  class Singleton_impl;
  std::unique_ptr<Singleton_impl> pimpl;
  Singleton();
public:
  static Singleton * instance();
  void foo();
};
```

```
class Singleton::Singleton_impl {
  void foo();
};
Singleton::Singleton() :
  pimpl(std::make_unique<Singleton_impl>()) {}
void Singleton::foo() { pimpl->foo(); }
```

Hiding implementation details

PIMPL

Abstract base

```
class ISingleton {
public:
  virtual void foo() = 0;
  static ISingleton * instance();
};
```

```
class Singleton : public ISingleton {
  //...
}
ISingleton * ISingleton::instance() {
    static Singleton * inst = nullptr;
    if(!inst) inst = new Singleton();
    return inst;
}
```

Ease of use

"ditch"
`instance()`

```
class Singleton {
  Singleton() = default;
  void foo_impl();
  static Singleton * instance();
public:
  static void foo();
};
```

```
void Singleton::foo_impl() {
  // ...
}
void Singleton::foo() {
  instance()->foo_impl();
}
```

Ease of use

"ditch" `instance()`

use

```
Singleton::foo();
```

instead of

```
Singleton::instance()->foo();
Singleton::instance().foo();
```

Ease of use

"ditch" `instance()`

Ditch the class!

```
namespace Singleton {
  void foo();
};
```

```
namespace Singleton {
namespace { // anon
  // singleton data here.
}
void foo() {

}
}
```

NB Take care if called from global object constructors.

Testing

Singleton

Testing

Singleton

```
void clearState(); // only for unit-tests!
```

Testing

Singleton

```
  void clearState(); // only for unit-tests!
```

```
#define private public
```

Please don't do this.

Testing

Singleton

Abstract base

```
class Singleton : public ISingleton {
  // ...
}
```

Testing

Singleton

Abstract base

class-less

```
namespace Singleton {
namespace detail {
  // data
}
}
```

In conclusion...

Next time you're implementing a singleton, consider putting your code in a namespace instead of a class.

Thanks for listening

arno@lepisk.se / arno.lepisk@hiq.se