3/11/2016

Forgers, Cheats, Pickpockets, and Lawyers

S. Kislinskiy

dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer

Welcome to the DKFZ!

dkfz.
GERMAN CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer

Herb Sutter, Exceptional C++ Style, Item #15

- What code can access the following parts of a class?
  - public
    - Any code
  - protected
    - Own member functions and friends
    - Member functions and friends of derived classes
  - private
    - Own member functions and friends

Your mission: Get write access to m_Private


```
class X {
public:
  X() : m_Private(1) { /* … */ }

  template<class T>
  void f(const T& t) { /* … */ }

  int GetValue() { return m_Private; }

  // …

private:
  int m_Private;
};
```

The Forger


```
class X {
public:
  // Instead of including X.h, manually (and
  // illegally) duplicates X‘s definition, and
  // adds a line such as:
  friend ::Hijack(X&);
};

void Hijack(X& x) {
  x.m_Private = 2; // Evil laughter here
}
```

The Forger

- Violates One Definition Rule
  „If a type is defined more than once, the definitions must be identical.“
- Uses another kind of X as all the other code
- Works on most compilers as long as data layout is identical

The Pickpocket


```
#define private public // Illegal
#include "X.h"

void Hijack(X& x) {
  x.m_Private = 2; // Evil laughter here
}
```

The Pickpocket

- Illegal to #define a reserved word
- Violates One Definition Rule

The Cheat


```
class BaitAndSwitch { // Same data layout as X
public:
  int m_NotSoPrivate;
};

void Hijack(X& x) {
  (reinterpret_cast<BaitAndSwitch&>(x))
    .m_NotSoPrivate = 2; // Evil laughter here
}
```

The Cheat

- Data layouts of X and BaitAndSwitch must be the same
- Result of reinterpret_cast is undefined

The Lawyer


```
namespace {
  struct Y {};
}

template<>
void X::f(const Y&) {
  m_Private = 2; // Evil laughter here
}

void DoDailyBusiness() {
  X x;
  x.f(Y());
}
```

The Lawyer

- Legal to specialize a member template on any type
- Doesn‘t specialize on the same type twice (One Definition Rule)
  - Y is guaranteed to be unique (own unnamed namespace)

The Question

- Do member templates break encapsulation?
  - Yes, don‘t do that!

Thank you for your attention!

Further information on www.dkfz.de

dkfz. GERMAN CANCER RESEARCH CENTER IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for A Life Without Cancer