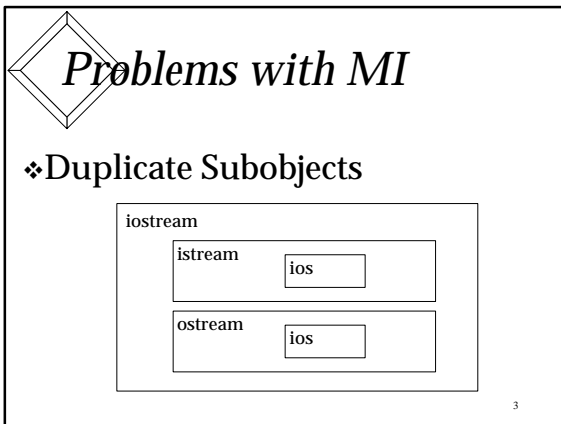
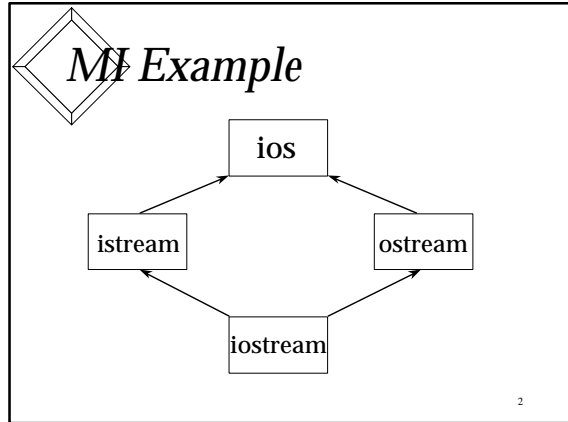


CS481 - Object-Oriented Programming

Multiple Inheritance

- ❖ Its useful to Inherit from more than one class:
 - Greater functionality
 - More code re-use.
- ❖ MI comes with complications
 - No single *root* of the Inheritance Tree



Problems with MI

- ❖ Ambiguous Upcasting
 - Upcasting is crucial to supporting the "is-a" relationship of inheritance
 - Which way does one cast an *iostream* instance as an *ios* object??!
 - *ios* has multiple *this* pointers!

Ambiguous Upcasting

```

//: MULTIPL1.CPP -- MI & ambiguity
#include <iostream.h>
#include "..\14\tstash.h"

class base {
public:
    virtual char* vf() const = 0;
};

class d1 : public base {
public:
    char* vf() const { return "d1"; }
};
    
```

Ambiguous Upcasting

```

class d2 : public base {
public:
    char* vf() const { return "d2"; }
};

// Causes error: ambiguous override of vf():
//! class mi : public d1, public d2 {};
    
```

CS481 - Object-Oriented Programming

Ambiguous Upcasting

```
main()
{
    tstash<base> b;
    b.add(new d1);
    b.add(new d2);
    // Cannot upcast: which subobject?:
    //! b.add(new mi);
    for(int i = 0; i < b.count(); i++)
        cout << b[i]->vf() << endl;
}
```

7

Two problems

- ❖ Can't create the class `mi` because the two def'ns of `vf()` clash!
- ❖ Creating vector `b[]` attempts to create a new `mi` and upcast the address to a `base*`.

8

Solutions

- ❖ Redefing the function `vf()` in class `mi` solves problem 1.
- ❖ The solution to the latter problem is to use:
 - *Virtual Inheritance.*

9

Virtual Inheritance

- ❖ Addresses the problem of multiple subobjects
 - Allows only one subobject
- ❖ Disambiguates Upcasting by using only the first subobject.

10

Virtual Inheritance

```
//: MULTIPL2.CPP -- Virtual Base Classes
#include <iostream.h>
#include "..\14\tstash.h"

class base {
public:
    virtual char* vf() const = 0;
};

class d1 : virtual public base {
public:
    char* vf() const { return "d1"; }
};
```

11


Virtual Inheritance

```
class d2 : virtual public base {
public:
    char* vf() const { return "d2"; }
};

// MUST explicitly disambiguate vf():
class mi : public d1, public d2 {
public:
    char* vf() const { return d1::vf(); }
};
```

12

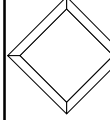
CS481 - Object-Oriented Programming



Virtual Inheritance

```
main()
{
    tstash<base> b;
    b.add(new d1);
    b.add(new d2);
    b.add(new mi); // OK
    for(int i = 0; i < b.count(); i++)
        cout << b[i]->vf() << endl;
}
```

13



14