Remaining Assignments

• HW #5 due tomorrow (Thurs)
• Project #2 due Sunday night
• Final exam on Monday
More Inheritance Details...

• C++ has **protected** visibility
• Friendship is **not** inherited
• There is no “root” class (like Java’s Object)
• Unlike Java, you may **reduce** visibility of inherited member (violating “Is-A”)

© 2011 Fawzi Emad, Computer Science Department, UMCP
Example: ReducedVisibility

- Visibility of inherited member *can* be reduced
- No problem for static binding
- What happens for virtual method???
Private Inheritance

class Foo : private Bar

Foo inherits all members from Bar, but they become private.

• Used to restrict API
• Violates “Is-A” relationship

Foo f;
Bar &b = f;       // No longer works
Bar *p = &f;     // No longer works
Example: PrivateInheritance

• Used to “restrict” API for base class
• Internally, private members from base class are still available.
Multiple Inheritance

Java:
• Java has “interfaces”
• A class may implement multiple interfaces
• No multiple inheritance

C++:
• No Java-style interfaces
• Abstract classes are used instead
• Multiple Inheritance allowed and used frequently to mimic Java interfaces
Why is Multiple Inheritance Scary?

“The Diamond of Death!”

(Next slide please...)
Baseball-Playing Chef

It’s the same guy!!
The Diamond of Death!

Chef has:
• name
• hat

Baseball Player has:
• name
• hat

By default, Baseball-Chef will inherit:
• Two names (He should only have one)
• Two hats (He needs them both)

There is a messy mechanism that allows you to change the default setup so that a Baseball-Chef has only one name (but still gets both hats).
• Constructors become complicated
• Issues remain with conflicting contracts
• Better idea: Avoid the Diamond of Death
Template Overview

**Basic idea:** Code that works the same way for many different data types. (The type is a parameter.)

Templates in C++ are similar to Java generics, but are quite different:

- In C++, the compiler will generate many different versions of the same code – one for each type that is encountered.
- Works with a class or a non-class function
- Parameterized type can be object or primitive
Function Templates

Syntax:

```cpp
template<typename T>
void foo(T)
{
    ...
}
```

What will compiler do with this?

```cpp
string s;
foo(s);
```

1. If this is the first time type `string` has been used to call `foo`, it creates a version of `foo` using `string` in place of “`T`”, and calls it here.
2. If `string` has been used previously, it calls the function previously generated.
Example: FunctionTemplates

findMin revisited:

• Template used so we can find the minimum value of any vector.
• Works for objects or primitives
• How many versions of findMin will the compiler generate?
• What kinds of objects will cause a compilation error?
Class Templates

• A bit tricky to separate interface from implementation
• Like function templates, compiler generates as many different copies of the class as it needs.
Example: ClassTemplate

• More than one parameterized type can be used
• “friend” can be used