

Inheritance Workshop

- What is meant by
 - Overloading a member function?
 - To overload a member function, we add another function in the same class that has the same name but takes different arguments
 - Overriding a member function?
 - To override a member function, we add a function in a child class that has the same name and takes the same arguments

- Are there any potential issues in the program shown on the next two pages?
 - The base class does not have a virtual destructor. If we destroy a derived instance via a pointer to base, the derived class's destructor will not be called
 - Derived class members will not be destroyed
 - The wrong object size will be passed to delete
 - The derived class member function `f(complex)` hides the base class member function `f(double)`

- What would you expect the main function to print out?
- Compile and run the program. Is the output what the programmer probably intended?
 - The programmer probably expected `base::f(double)` to be printed when `d.f()` is called
 - However, this member function is not visible in derived
 - Instead, the compiler performs an implicit conversion from `double` to `complex<double>` and calls `derived::f`
- Fix any issues and run the program again

Base and derived class

```
class base {  
public:  
    virtual void f(double) { cout << "base::f(double)" << endl; }  
};
```

```
class derived: public base {  
public:  
    void f(complex<double>) { cout << "derived::f(complex)" << endl; }  
};
```

Main function

```
int main() {  
    base b;  
    derived d;  
    unique_ptr<base> pb = make_unique<derived>();  
  
    b.f(1.0);  
    d.f(2.0);  
    pb->f(3.0);  
}
```