

Module M2

Partha Pratin Das

Objectives Outlines

Inheritance i C++

private Inheritanc Uncopyable HAS-A

protecte Inheritano

Visibilit

Examples

Module Summar

Programming in Modern C++

Module M25: Inheritance: Part 5: private & protected Inheritance

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagour

ppd@cse.iitkgp.ac.in

All url's in this module have been accessed in September, 2021 and found to be functional



Module Recap

Module M2!

Partha Pratin Das

Objectives & Outlines

Inheritance

C++

Inheritane Uncopyabl

protect

Inheritai

Visibility

Examples

Module Summar

• Using the Phone Hierarchy as an example analyzed the design process with inheritance





Module Objectives

Module M2

Partha Pratin Das

Objectives & Outlines

Inheritance C++

C++ private

Inheritane Uncopyabl

protect

Inheritar

_ .

Examples

• Explore restricted forms of inheritance (private and protected) in C++ and their semantic implications



Module Outline

Module M2

Partha Pratir Das

Objectives & Outlines

Inheritance C++

private

Inheritan Uncopyabi HAS-A

protecte Inheritan

Visibility

Liamples

Objectives & Outlines

- 2 Inheritance in C++
- private Inheritance
 - Uncopyable
 - HAS-A
- 4 protected Inheritance
- Visibility
- **6** Examples
- Module Summary



Inheritance in C++

Module M2

Partha Pratim

Objectives Outlines

Inheritance in C++

private

Înheritan Uncopyabi

HAS-A

Inheritar

Visibilit

Examples

Module Summai

Inheritance in C++



Inheritance in C++: Semantics

Inheritance in

```
    Derived ISA Base

     Base
                           Norivol
```

```
class Base:
                            // Base Class = Base
class Derived: public Base; // Derived Class = Derived
```

- Use keyword public after class name to denote inheritance
- Name of the Base class follow the keyword



Inheritance Exercise: What is the output?

```
Module M2
```

Partha Pratim Das

Objectives Outlines

Inheritance in C++

private Inheritano Uncopyable HAS-A

Inheritano

Visibility

Madula Commun

```
class B {
public:
   B()
           cout << "B ": }
    "B() { cout << ""B "; }
};
class C {
public:
   C()
           cout << "C ": }
    ~C() { cout << "~C": }
};
class D : public B {
   C data : // Embedded Object
public:
   D()
         { cout << "D " << endl; } // Intrinsic Base Class Object
    ~D() { cout << "~D ": }
};
int main() {
   D d:
```



Inheritance Exercise: What is the output?

```
Module M25
Partha Pratim
Das
bjectives &
```

Inheritance in

C++

Inheritano Uncopyable HAS-A

protected Inheritano

Visibility

Examples

Module Summary

```
class B {
public:
   B()
           cout << "B "; }
    "B() { cout << ""B ": }
};
class C {
public:
   C()
         { cout << "C "; }
    ~C() { cout << "~C"; }
class D : public B {
   C data_: // Embedded Object
public:
   D()
         { cout << "D " << endl; } // Intrinsic Base Class Object
    ~D() { cout << "~D "; }
int main() {
   D d:
Output:
            // First base class object, then embedded object, finally self
BCD
~D ~C ~B
```



Module M2

Partha Pratir Das

Objectives Outlines

Inheritance C++

private Inheritance

Inheritan

HAS-/

protecte Inheritano

Visibilit

Examples

Aodule Summar

private Inheritance



private Inheritance

• private Inheritance

Definition

```
class Base:
class Derived: private Base;
```

- Use keyword private after class name
- Name of the Base class follow the keyword
- o private inheritance does not mean generalization / specialization



private Inheritance

```
public Inheritance
```

```
class Student:
   public Person { ... };
void eat(const Person& p); // anyone can eat
void study(const Student& s); // only students study
```

Person p: // p is a Person Student s: // s is a Student

class Person { ... };

eat(p): // fine, p is a Person eat(s): // fine, s is a Student. // and a Student is-a Person

study(s): // fine study(p); // error! p isn't a Student

• Compilers *converts* a derived class object (Student) into a base class object (Person) if the inheritance relationship is public

private Inheritance

class Person { ... };

```
class Student: // inheritance is now private
   private Person { ... };
void eat(const Person& p); // anyone can eat
void study(const Student& s): // only students study
Person p: // p is a Person
Student s: // s is a Student
eat(p): // fine, p is a Person
eat(s): // error! a Student isn't a Person
```

 Compilers will not convert a derived class object (Student) into a base class object (Person) if the inheritance relationship is private



Uncopyable Class

Module M2

Partha Pratii

Objectives Outlines

 $\begin{array}{c} {\sf Inheritance} \\ {\sf C++} \end{array}$

Uncopyable

protecte Inheritano

mneritano

Example

Module Summar

Suppose we want to design a MyClass every object must be unique. That is instance objects must not be copied (the
class needs to be Uncopyable)

```
MyClass m1;
MyClass m2;

MyClass m3(m1); // attempt to copy m1 - should not compile!
m1 = m2; // attempt to copy m2 - should not compile!
```

 Naturally, we do not want to provide copy constructor or copy assignments operator. But that does not work as the compiler will provide free versions of these functions. How to stop that?

```
class MyClass {
public:
...
private:
...
    MyClass(const MyClass&); // declarations only
    MyClass& operator=(const MyClass&);
};
```

The last trick is not to provide the implementations (bodies) of these copy functions.

- With the above any global function or other class would not be able to copy there will be compilation error; and any
 member function of MyClass will get linker error on missing implementation
- This is, of course, not an elegant solution and has to depend on the programmer to do things right for every such Uncopyable class. private inheritance helps out



Uncopyable

Uncopyable

```
    class Uncopyable is designed as a root class that can make copy functions of any child class private

  class Uncopyable { protected: // allow construction and destruction of derived objects ...
           Uncopyable() { }
           ~Uncopyable() { }
       private:
           Uncopyable(const Uncopyable&); // ... but prevent copying
           Uncopyable& operator=(const Uncopyable&):
   };
  Any class that inherits from class Uncopyable, will not have copy functionality:
```

class MyClass: private Uncopyable { // class no longer declares copy ctor or copy assign. operator // ...

```
void ProhibitiveCopy() { MyClass test1, test2; // Member functions cannot perform copy
        MyClass test3(test1): // Error 1: 'Uncopyable::Uncopyable' : cannot access private member
        test2 = test1:
                             // Error 2: 'Uncopyable::operator =' : cannot access private member
int main() { MyClass test1, test2; // Global functions cannot perform copy
   MyClass test3(test1); // Error 1: 'Uncopyable::Uncopyable': cannot access private member
   test2 = test1:
                         // Error 2: 'Uncopyable::operator =' : cannot access private member
```

- The inheritance from Uncopyable need not be public (though it will work), hence using private we express that it is purely for implementation - not for modeling ISA that actually does not exist You may also use the boost::noncopyable
- C++11 provides an explicit support by delete to stop compilers from providing free functions Programming in Modern C++ Partha Pratim Das



Car HAS-A Engine: Composition OR private Inheritance?

dule M29

Partha Pratio

Objectives Outlines

Inheritance C++

private Inheritano Uncopyable HAS-A

protecte Inheritano

Visibility

Module Summ

```
Simple Composition
```

mposition private Inheritance

```
#include <iostream>
                                                     #include <iostream>
 using namespace std:
                                                     using namespace std:
 class Engine {
                                                     class Engine {
 public:
                                                     public:
     Engine(int numCylinders) { }
                                                         Engine(int numCvlinders) { }
     void start() { } // Starts this Engine
                                                         void start() { } // Starts this Engine
 class Car {
                                                     class Car : private Engine { // Car has-a Engine
 public:
                                                     public:
      // Initializes this Car with 8 cylinders
                                                          // Initializes this Car with 8 cylinders
     Car() : e_{(8)} \{ \}
                                                         Car() : Engine(8) { }
     // Start this Car by starting its Engine
                                                         // Start this Car by starting its Engine
     void start() { e .start(): }
                                                         using Engine::start:
 private:
     Engine e : // Car has-a Engine
 int main() {
                                                     int main() {
     Car c:
                                                         Car c:
     c.start():
                                                         c.start():
Programming in Modern C++
                                                       Partha Pratim Das
                                                                                                       M25 14
```



Module M2

Partha Pratir Das

Objectives of Outlines

Inheritance i C++

private Inheritanc Uncopyable HAS-A

protecte

Visibility

Module Summa

• For HAS-A: Use composition when you can, private inheritance when you have to

- Private inheritance means nothing during software design, only during software implementation
- Private inheritance means is-implemented-in-terms of. It is usually inferior to composition, but it makes sense when a derived class needs access to protected base class members or needs to redefine inherited virtual functions
- Scott Meyers in Item 32, Effective C++ (3rd. Edition)



protected Inheritance

Module M2

Partha Pratir Das

Objectives Outlines

Inheritance C++

private Inheritanc

Inheritanc Uncopyable

protected Inheritance

Inheritan

Examples

Module Summar

protected Inheritance



protected Inheritance

protected Inheritance

protected Inheritance

```
    Definition
```

```
class Base:
class Derived: protected Base;
```

- Use keyword protected after class name
- Name of the Base class follow the keyword
- o protected inheritance does not mean generalization / specialization
- Private inheritance means something entirely different (from public inheritance), and protected inheritance is something whose meaning eludes me to this day
- Scott Meyers in Item 32, Effective C++ (3rd. Edition)



Visibility

Module M2

Partha Pratir Das

Objectives Outlines

Inheritance | C++

private Inheritanc

Uncopyabl

protect Inheritan

Visibility

Examples

Module Summa



Visibility



Visibility across Access and Inheritance

• Visibility Matrix

odule M2!

Partha Pratii

Objectives Outlines

Inheritance C++

private Inheritanc Uncopyable HAS-A

protect

Visibility

Examples

. Module Summar

Inheritance

| | | public | protected | private |
|------------|-----------|-----------|-----------|---------|
| Visibility | public | public | protected | private |
| | protected | protected | protected | private |
| | private | private | private | private |





Use and Examples

Module M2

Partha Pratir Das

Objectives Outlines

Inheritance C++

private Inheritano

Uncopyable Uncopyable

Inheritan

Visibilit

Examples

Module Summar

Use and Examples



Inheritance Exercise: What is the output?

```
Module M2
```

Partha Pratir Das

Objectives Outlines

 $\begin{array}{c} \text{Inheritance} \\ \text{C}++ \end{array}$

private Inheritanc Uncopyable HAS-A

protected Inheritance

Visibility

Examples

10dule Summar

```
class B {
protected:
   B()
           cout << "B "; }
    "B() { cout << ""B "; }
class C : public B {
protected:
   C()
           cout << "C "; }
    ~C() { cout << "~C ": }
class D : private C {
   C data :
public:
         { cout << "D " << endl; }
   D()
    ~D() { cout << "~D "; }
};
int main() {
   D d:
```



Inheritance Exercise: What is the output?

```
Module M29
Partha Pratin
Das
bjectives & utlines
theritance in
```

private Inheritano Uncopyable HAS-A

protected Inheritance

Visibility

Examples

lodule Summar

```
class B {
protected:
    B()
           cout << "B "; }
    "B() { cout << ""B ": }
class C : public B {
protected:
    C()
           cout << "C "; }
    ~C() { cout << "~C": }
};
class D : private C {
    C data_:
public:
    D()
           cout << "D " << endl: }
    ~D() { cout << "~D ": }
};
int main() {
    D d:
Output:
BCBCD
```



Inheritance Exercise: Access Rights

Module M2

Partha Pratir Das

Objectives Outlines

Inheritance C++

private Inheritanc Uncopyable HAS-A

protected Inheritano

Visibility

Examples

Module Summar

Inaccessible Members

Accessible Members

```
class A { private: int x:
                                                             void f(A& a.
   protected: int v:
                                                                    B& b, C& c, D& d,
   public: int z;
                                                                    E& e, F& f, G& g) {
};
                                                                 a.z:
class B : public A { private: int u;
   protected: int v;
                                                                 b.z;
   public: int w: void f() { x: }
                                                                 b.w:
class C: protected A { private: int u:
                                                                 c.w;
   protected: int v;
   public: int w: void f() { x: }
                                                                 d.w:
class D: private A { private: int u;
                                                                 e.z:
   protected: int v:
                                                                 e.w:
   public: int w: void f() { x: }
                                                                 f.w:
class E : public B { public: void f() { x; u; }
                                                                 g.w;
class F : public C { public: void f() { x: u: }
class G : public D { public: void f() { x; y; z; u; }
};
```



Module Summary

Module Summary

• Introduced restricted forms of inheritance and protected specifier

• Discussed how private inheritance is used for Implemented-As Semantics

Partha Pratim Das M25 24