OOP HW - Inheritance & Access Control Notes

Notes

2026 年 5 月 6 日

1 Inheritance, Access Specifiers and Object Lifecycle

Question 6

If class B inherits class A privately. And class B has a friend function. Will the friend function be able to access the private member of class A?

- A. Yes, because friend function can access all the members
- B. Yes, because friend function is of class B
- **C. No, because friend function can only access private members of friend class**
- D. No, because friend function can access private member of class A also

解析：正确选项是 **C**。在 C++ 中，友元特权是**不继承、不传递**的。B 的友元函数只拥有访问 B 自身所有成员的特权。由于 A 的 `private` 成员对 B 本身都是不可见、被锁死的，B 的友元函数自然也绝对无法“跨越两层”去访问 A 的私有成员。

Question 10

Which among the following is true for the given code below?

```
class A {
protected: int marks;
public:
    A() { marks=100; }
    void disp() { cout<<"marks="<<marks; }
};
class B: protected A { };
B b;
b.disp();
```

- **A. Object b can’ t access disp() function**
- B. Object b can access disp() function inside its body

- C. Object b can’t access members of class A
- D. Program runs fine

解析：正确选项是 **A**。本题考查继承权限的降级规则。由于使用了 `protected` 保护继承，基类 A 中原有的 `public` 接口 `disp()` 在派生类 B 中被降级成了 `protected`。受保护成员只对 B 的内部和 B 的子类开放，绝对不允许在外部通过对象（如 `b.disp()`）直接调用，因此引发编译错误。

Question 14

If a class have default constructor defined in private access, and one parameter constructor in protected mode, how will it be possible to create instance of object?

- A. Define a constructor in public access with different signature
- B. Directly create the object in the subclass
- C. Directly create the object in main() function
- **D. Not possible**

解析：正确选项是 **D**。构造函数是对象实例化的“唯一大门”。如果所有的构造函数（大门）全部被 `private` 或 `protected` 锁死，外部代码（比如 `main` 函数）就没有任何访问权限，也就绝对不可能（Not possible）直接在外部创建该类的实例（著名的单例模式正是利用这一特性封死外部实例化的）。

Question 19

Which specifier allows to secure the public members of base class in inherited classes?

- A. Private
- B. Protected
- C. Public
- **D. Private and Protected**

解析：正确选项是 **D**。**【重点笔记】：**在 C++ 面向对象设计的语境中，**“secure（保护/确保安全）”通常代表“隐藏外部的访问权限”**。如果你希望基类原本对外敞开的 `public` 接口，在派生类中变得“安全且不可见”，你应该使用 `private` 继承或 `protected` 继承。这两种方式都会将继承来的公有接口降级，从而切断外部对象直接访问的可能。

Question 25

Which type of inheritance is illustrated by the following code?

```
class student { public: int marks; };
class topper: public student { public: char grade; };

class average { public: int makrs_needed; };
class section: public average { public: char name[10]; };
class overall: public average { public: int students; };
```

- A. Single level
- B. Multilevel and single level
- C. Hierarchical
- **D. Hierarchical and single level**

解析：正确选项是 **D**。`student` 派生出 `topper` 是经典的一对一 **“单继承（Single level）”**；而 `average` 像树根一样，同时横向派生出了 `section` 和 `overall` 两个平行的分支，这种 “一个基类、多个派生类” 的结构被称为 **“层次继承 / 树状继承（Hierarchical inheritance）”**。

Question 27

Which type of inheritance results in the diamond problem?

- A. Single level
- **B. Hybrid (部分题库为 Hierarchical & Multiple 的组合)**
- C. Hierarchical
- D. Multilevel

解析：【新概念笔记】：大名鼎鼎的菱形问题（Diamond Problem）是由 **Hybrid（混合继承）**引发的。什么是混合继承？它其实就是 **Hierarchical（层次继承）+ Multiple（多重继承）**。例如：爷爷类派生出两个大伯类（这是 Hierarchical）；然后孙子类又同时继承这两个大伯类（这是 Multiple）。孙子体内会保留两份爷爷的数据，导致调用时的严重歧义。在题库中通常选 Hybrid，它代表了导致菱形危机的复杂混合结构。

Question 31

If class A and class B are derived from class C and class D, then _____

- **A. Those are 2 pairs of single inheritance**
- B. That is multilevel inheritance

- C. Those is enclosing class
- D. Those are all independent classes

解析：正确选项是 **A**。A 继承 C，B 继承 D。这两个继承结构各自为战，互不相干。每一对都完美符合“一个子类对应一个父类”的单继承定义。因此这仅仅是简简单单的 2 对单继承关系。

Question 32

Single level inheritance supports _____ inheritance.

- A. Runtime
- **B. Compile time**
- C. Multiple inheritance
- D. Language independency

解析：正确选项是 **B**。在 C++ 中，类的继承关系、内存布局大小、以及普通成员函数的绑定，全都是在**编译期（Compile time）**就已经严格静态确定的（即使是虚函数表 vtable，其结构的构建也是在编译期完成的，只不过查找动作发生在运行期）。

Question 34

Which among the following is false for single level inheritance?

- A. There can be more than 2 classes in program to implement single inheritance
- **B. There can be exactly 2 classes to implement single inheritance in a program**
- C. There can be more than 2 independent classes involved in single inheritance
- D. The derived class must implement all the abstract method if single inheritance is used

解析：正确选项是 **B**。“单继承”仅仅是规范了“认爹的规则（一个派生类只能有一个直接基类）”，但绝不限制程序总体规模。错在“恰好 2 个类（exactly 2 classes）”，一个程序里定义成百上千个类并组成无数对单继承是完全合法的。

Question 35

Which among the following best defines multilevel inheritance?

- **A. A class derived from another derived class**
- B. Classes being derived from other derived classes
- C. Continuing single level inheritance

- D. Class which have more than one parent

解析：正确选项是 **A**。多级继承（Multilevel Inheritance）是指一条垂直向下的纵向继承链：基类 → 派生类 → 再派生类。即一个派生类，它是从另一个本来也是派生类的类继承而来的。

Question 37

In multilevel inheritance one class inherits _____

- **A. Only one class**
- B. More than one class
- C. At least one class
- D. As many classes as required

解析：正确选项是 **A**。即使是多级继承（A → B → C），你在审查任何一个单独的层级时，它依然只继承了直接上级的一个类（例如 C 的直接基类只有 B）。这和拥有多个直接父类的多重继承（Multiple Inheritance）有本质区别。

Question 40

Why does diamond problem arise due to multiple inheritance?

- **A. Methods with same name creates ambiguity and conflict**
- B. Methods inherited from the superclass may conflict
- C. Derived class gets overloaded with more than two class methods
- D. Derived class can’t distinguish the owner class of any derived method

解析：正确选项是 **A**。菱形问题的核心痛点在于**二义性（Ambiguity）**。当子类通过多条不同的继承路径，继承了最顶层基类同名的方法或变量时，当你试图调用它，编译器会懵逼：因为存在两条等价的查找路径，编译器无法决断该走哪一条，从而抛出二义性编译错误。

Question 44

Pointer to a base class can be initialized with the address of derived class, because of _____

- **A. derived-to-base implicit conversion for pointers**
- B. base-to-derived implicit conversion for pointers
- C. base-to-base implicit conversion for pointers
- D. derived-to-derived implicit conversion for pointers

解析：正确选项是 **A**。这是 C++ 实现多态的绝对基石，被称为 **“向上转型（Upcasting）”**。因为一个派生类对象在内存布局中必定包含一个完整的基类子对象，所以 C++ 编译器原生且安全地支持将派生类的指针**隐式转换（implicit conversion）**为基类的指针。

Question 47

Can constructors be overloaded in derived class?

- **A. Yes, always**
- B. Yes, if derived class has no constructor
- C. No, programmer can’t do it
- D. No, never

解析：正确选项是 **A**。派生类（子类）在行为上也是一个完完全全的标准类。就像普通类一样，只要它的参数列表（类型、数量、顺序）不同，你就可以随心所欲地为派生类重载无数个构造函数。

Question 55

Is it compulsory for all the classes in multilevel inheritance to have constructors defined explicitly if only last derived class object is created?

- A. Yes, always
- B. Yes, to initialize the members
- **C. No, it not necessary**
- D. No, Constructor must not be define

解析：正确选项是 **C**。C++ 编译器极其聪明，如果你在继承链的某一层没有手动（显式）定义任何构造函数，编译器会自动为你生成一个“隐式的无参默认构造函数（Implicit default constructor）”。因此，即便继承链很长，也不强制要求（Not compulsory）你给每一个基类都写代码去定义构造函数。

附录：重点知识梳理

C++ 经典继承模式全景对比

在 C++ 面向对象编程与考试中，理清类的派生拓扑结构是解决复杂结构题和“菱形危机”的关键。请牢记以下四种核心继承模式的本质区别：

继承模式 (英文)	拓扑结构	核心描述与易错考点
Single level (单继承)	**1 对 1**	一个派生类只拥有**唯一**一个直接基类。（A → B） **易错点：**单继承约束的只是“认爹的规则”。哪怕程序里写了 1000 个独立的类，只要它们全都是两两一对一继承，整体就依然是单继承。
Multilevel (多级继承)	**单向垂直链**	派生类作为新的基类，继续派生下一个类，形成家族代代相传的垂直链条。（A → B → C） **易错点：**构造顺序永远从“最顶层的老祖宗”开始往下，析构顺序严格反转（拆楼逻辑）。每层只能看见紧挨着自己的上一层非私有成员。
Hierarchical (层次继承)	**1 对多**	一个基类同时派生出多个不同的子类，像大树发叉一样散开。（A → B 且 A → C） **易错点：**兄弟子类之间是平行的，互不干扰。多用于同一基类在不同业务方向上的分支实现。
Hybrid (混合继承)	**网状交织**	两种或两种以上继承方式的结合体（通常是 Hierarchical 加上 Multiple 多重继承）。 **易错点：**这是导致大名鼎鼎的 **菱形问题 (Diamond Problem)** 的罪魁祸首！子类通过不同的路径继承了同一个顶层基类的两份数据拷贝，调用时会产生严重的“二义性（Ambiguity）”。

【特别补充：Multiple Inheritance (多重继承)】

很多时候题目会单独考察它。它指的是一个子类同时拥有**多个直接基类**（多对 1，例如 B 和 C 共同派生出 D）。它正是引发上述 Hybrid 混合继承“菱形危机”的核心机制。在 C++ 中，为了解决菱形二义性，必须在多重继承的中间层使用 `virtual` 关键字进行**虚继承 (Virtual Inheritance)**。