【C++】Day58

Class	C++
Date	@February 17, 2022
Material	
Series Number	
Summary	

【Ch13】Copy Control

private Copy Control

Prior to the new standard, classes prevented copies by declaring their copy constructor and copy-assignment operator as `private`:

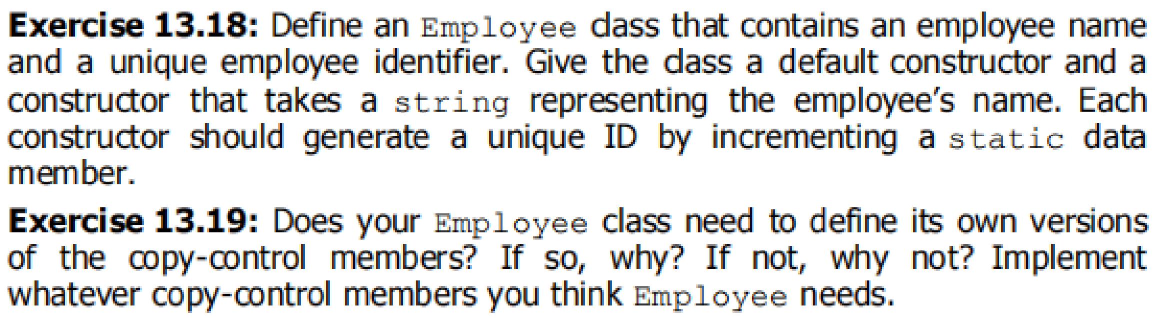
```
class PrivateCopy {
  // no access specifier; following members are private by default.
  // copy control is private and so is inaccessible to ordinary user code.
  PrivateCopy(cosnt PrivateCopy&);
  PrivateCopy& operator=(const PrivateCopy&);

public:
  PrivateCopy() = default;
  ~PrivateCopy(); // users can define objects of this type but not copy them.
};
```

Because the copy constructor and copy-assignment operator are `private`, user code will not be able to copy such objects.

However, friends and members of the class can still make copies. To prevent copies by friends and members, we declare these members as private but do not define them.

Best Practice: Classes that want to prevent copying should define their copy constructor and copy-assignment operators using `= delete` rather than making those members `private`.

Exercise



Exercise 13.18: Define an `Employee` class that contains an employee name and a unique employee identifier. Give the class a default constructor and a constructor that takes a `string` representing the employee's name. Each constructor should generate a unique ID by incrementing a `static` data member.

Exercise 13.19: Does your `Employee` class need to define its own versions of the copy-control members? If so, why? If not, why not? Implement whatever copy-control members you think `Employee` needs.

See 13_18.cpp for code