Deeper Look into Classes

LAB 6: Employee Count

Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

4/1/2021

Objectives

- **Learn how to use friend. (Chapter 10.4)**
- **Learn how to use static data member and static member functions. (Chapter 10.6)**

Friend Functions & Friend Classes

- A friend function of a class is defined outside that class's scope, yet has the right to access the non-public members of the class. (Chapter 10.4)
- A friend can be a function, an entire class, or a member function of another class.
- Friendship relation is neither symmetric nor transitive.

```
class Count {
  friend void setX( Count &, int ); // friend
declaration
public:
  Count(): x( 0 ) // initialize x to 0
  { /* empty body*/ } // end constructor
private:
  int x;
};
void Count::print const { cout << x << endl; }
void setX(Count &c, int val)
{ c.x = val;}  // setX is a friend function
```

```
int main()
{
  Count counter; // create Count object
  cout << "counter.x after instantiation: ";
  counter.print();
  setX( counter, 8 );
 // set x using a friend function
  cout << "counter.x after call to setX friend
function: ";
  counter.print();
} // end main
```

Static Data Member

- **(Chapter 10.6)**
  - Used to handle the case that only one copy of the member's value for all the objects belonging to the same class. For example, a static data member called objectCount can be used to record the number of objects of the same type.
  - Belonging to no object. Thus, It should be initialized in the global namespace scope by **className::staticDataMember= initialValue.**
  - Being accessed using className::staticDataMember if it is a public one.
  - Being accessed using a public static member function or friends if it is a private or protected one.

Example: Static Data Member

- // Used when all objects refer to the same value of the data member.

```
#include <string>
using namespace std;
class Employee
{
public:
   Employee( const string &, const string & ); // constructor
   ~Employee(); // destructor
   string getFirstName() const; // return first name
   string getLastName() const; // return last name
   // static member function
   static int getCount(); // return number of objects instantiated
private:
   string firstName;
   string lastName;
   // static data
   static int count; // number of objects instantiated
}; // end class Employee
```

Initializing Static Data Members

- **Usually initialized in the global scope, i.e., not in the class definition.**

```
// Fig. 10.21: Employee.cpp
// Employee class member-function definitions.
#include <iostream>
#include "Employee.h" // Employee class definition
using namespace std;

// define and initialize static data member at global namespace scope
int Employee::count = 0; // cannot include keyword static

// define static member function that returns number of
// Employee objects instantiated (declared static in Employee.h)
int Employee::getCount()
{
   return count;
} // end static function getCount
```

Lab 6: Employee Count

- **Modify the code in Fig. 10.20, Fig. 10.21 such that the given main() function will generate the desired output. The following are the major tasks needed to be completed.**
  - **Add a private data member gender of type char. If gender is 'F', the employee is a female. If gender is 'M', the employee is a male. Otherwise, "Employee's gender specification is incorrect." followed by the gender's value should be printed when an employee object is created.**
  - **Add two private static data members fCount and mCount which counts the number of female employees and male employees, respectively.**
  - **Add a member function int getfCount() to return the value of fCount.**
  - **Add a member function int getmCount() to return the value of mCount.**
  - **Add a member function void printCount() to print the values of all the static data members of class Employee.**
  - **Add two member functions Employee& printFirstName() and Employee* printLastName() to print the first name and last name of an employee, respectively.**
  - **Add a function void print(const Employee &) to print the values of all the non-static data members of an employee in the global namespace scope and make it as a friend of the class Employee.**
  - **Replace void print(const Employee &) with void print(const Employee), then compile and run the program again. What are the differences on the output?**
  - **Add a copy constructor Employee(const Employee &), then compile and run the program again. What are the differences on the output?**
- **The given main() function should not be changed.**

main() Function (1)

```
int main()
{
  // no objects exist; use class name and binary scope resolution
  // operator to access static member function getCount
  cout << "Number of employees before instantiation of any objects is "
    << Employee::getCount() << endl; // use class name

  // the following scope creates and destroys
  // Employee objects before main terminates
  {
    Employee e1( "Susan", "Baker", 'M' );
    Employee e2( "Robert", "Jones", 'F' );
    Employee e3( "Emily", "Willow", 'F' );
    Employee e4( "Jhon", "Reid", 'K' );
    Employee e5( "Maria", "Vinci", 'M' );
    Employee e6( "Vincent", "Url", 'F');
    Employee e7( "RB", "Lin", 'M' );
    print(e5);
    print(e6);
    // two objects exist; call static member function getCount again
    // using the class name and the binary scope resolution operator
    cout << "Number of employees after objects are instantiated is "
      << Employee::getCount() << endl;

    cout << "\n\nEmployee 1: "
      << e1.getFirstName() << " " << e1.getLastName()
      << "\nEmployee 2: "
      << e2.getFirstName() << " " << e2.getLastName() << "\n\n";
```

main() Function (2)

```
   Employee e8("Tomas", "Hwang", 'F');
   Employee e9("James", "Wang", 'F');
   cout << "Number of employees after objects are instantiated is: \n";
     Employee::printCount();

   cout << "\n\nEmployee 3: ";
     e3.printFirstName().printLastName();
   cout << "\nEmployee 3: ";
     e3.printLastName()->printFirstName();

   cout << "\nEmployee 4: ";
     e4.printFirstName().printLastName();
   cout << "\nEmployee 4: ";
     e4.printLastName()->printFirstName();
   cout << endl << endl;
} // end nested scope in main

// no objects exist, so call static member function getCount again
// using the class name and the binary scope resolution operator
cout << "\nNumber of employees after objects are deleted is "
   << "Number of employees= " <<Employee::getCount() << "   Female employees= "
   << Employee::getfCount()  << "   Male employees= " << Employee::getmCount() << endl;
} // end main
```

Key Points for Grading

- Check the printout highlighted by red circles marked in the output.
- Function **void print(const Employee &)** should be used to generate Output-1
- Functions void **print(const Employee)** should be used to generate Output-2
- Functions void **print(const Employee)** and **copy constructor Employee(const Employee&)** should be used to generate Output-3
- Other printout should also be inspected.

Output-1

```
Number of employees before instantiation of any objects is 0
Employee constructor for Susan Baker called.
Employee constructor for Robert Jones called.
Employee constructor for Emily Willow called.
Employee's gender specification is incorrect. K
Employee constructor for Jhon Reid called.
Employee constructor for Maria Vinci called.
Employee constructor for Vincent Url called.
Employee constructor for RB Lin called.
Maria Vinci M
Vincent Url F
Number of employees after objects are instantiated is 7


Employee 1: Susan Baker
Employee 2: Robert Jones

Employee constructor for Tomas Hwang called.
Employee constructor for James Wang called.
Number of employees after objects are instantiated is:
Number of Employees= 9   Male employees= 3   Female employees= 5


Employee 3: Emily Willow
Employee 3: Willow Emily
Employee 4: Jhon Reid
Employee 4: Reid Jhon

~Employee() called for James Wang
~Employee() called for Tomas Hwang
~Employee() called for RB Lin
~Employee() called for Vincent Url
~Employee() called for Maria Vinci
~Employee() called for Jhon Reid
~Employee() called for Emily Willow
~Employee() called for Robert Jones
~Employee() called for Susan Baker

Number of employees after objects are deleted is Number of employees= 0   Female employees= 0   Male employees= 0
```

Output-2 (after replacing print() Function)

```
Number of employees before instantiation of any objects is 0
Employee constructor for Susan Baker called.
Employee constructor for Robert Jones called.
Employee constructor for Emily Willow called.
Employee's gender specification is incorrect. K
Employee constructor for Jhon Reid called.
Employee constructor for Maria Vinci called.
Employee constructor for Vincent Url called.
Employee constructor for RB Lin called.
Maria Vinci M
~Employee() called for Maria Vinci
Vincent Url F
~Employee() called for Vincent Url
Number of employees after objects are instantiated is 5


Employee 1: Susan Baker
Employee 2: Robert Jones

Employee constructor for Tomas Hwang called.
Employee constructor for James Wang called.
Number of employees after objects are instantiated is:
Number of Employees= 7   Male employees= 2   Female employees= 4


Employee 3: Emily Willow
Employee 3: Willow Emily
Employee 4: Jhon Reid
Employee 4: Reid Jhon

~Employee() called for James Wang
~Employee() called for Tomas Hwang
~Employee() called for RB Lin
~Employee() called for Vincent Url
~Employee() called for Maria Vinci
~Employee() called for Jhon Reid
~Employee() called for Emily Willow
~Employee() called for Robert Jones
~Employee() called for Susan Baker

Number of employees after objects are deleted is Number of employees= -2   Female employees= -1   Male employees= -1
```

Output-3 (after adding copy constructor)

```
Number of employees before instantiation of any objects is 0
Employee constructor for Susan Baker called.
Employee constructor for Robert Jones called.
Employee constructor for Emily Willow called.
Employee's gender specification is incorrect. K
Employee constructor for Jhon Reid called.
Employee constructor for Maria Vinci called.
Employee constructor for Vincent Url called.
Employee constructor for RB Lin called.
Maria Vinci M
~Employee() called for Maria Vinci
Vincent Url F
~Employee() called for Vincent Url
Number of employees after objects are instantiated is 7


Employee 1: Susan Baker
Employee 2: Robert Jones

Employee constructor for Tomas Hwang called.
Employee constructor for James Wang called.
Number of employees after objects are instantiated is:
Number of Employees= 9   Male employees= 3   Female employees= 5


Employee 3: Emily Willow
Employee 3: Willow Emily
Employee 4: Jhon Reid
Employee 4: Reid Jhon

~Employee() called for James Wang
~Employee() called for Tomas Hwang
~Employee() called for RB Lin
~Employee() called for Vincent Url
~Employee() called for Maria Vinci
~Employee() called for Jhon Reid
~Employee() called for Emily Willow
~Employee() called for Robert Jones
~Employee() called for Susan Baker

Number of employees after objects are deleted is Number of employees= 0   Female employees= 0   Male employees= 0
```

```
// Fig. 10.20: Employee.h
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <string>
using namespace std;

class Employee
{
public:
   Employee( const string &, const string & ); // constructor
   ~Employee(); // destructor
   string getFirstName() const; // return first name
   string getLastName() const; // return last name
   // static member function
   static int getCount(); // return number of objects instantiated
private:
   string firstName;
   string lastName;
   // static data
   static int count; // number of objects instantiated
}; // end class Employee
#endif
```

```
// Fig. 10.21: Employee.cpp
// Employee class member-function definitions.
#include <iostream>
#include "Employee.h" // Employee class definition
using namespace std;

// define and initialize static data member at global namespace scope
int Employee::count = 0; // cannot include keyword static

// define static member function that returns number of
// Employee objects instantiated (declared static in Employee.h)
int Employee::getCount()
{
   return count;
} // end static function getCount
```

```
// constructor initializes non-static data members and
// increments static data member count
Employee::Employee( const string &first, const string &last )
   : firstName( first ), lastName( last )
{
   ++count; // increment static count of employees
   cout << "Employee constructor for " << firstName
      << ' ' << lastName << " called." << endl;
} // end Employee constructor

// destructor deallocates dynamically allocated memory
Employee::~Employee()
{
   cout << "~Employee() called for " << firstName
      << ' ' << lastName << endl;
   --count; // decrement static count of employees
} // end ~Employee destructor
```

```
// return first name of employee
string Employee::getFirstName() const
{
   return firstName; // return copy of first name
} // end function getFirstName

// return last name of employee
string Employee::getLastName() const
{
   return lastName; // return copy of last name
} // end function getLastName
```