

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | Fu Ruoxuan |
| **Student ID:** | 222019321062060 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Review of C++ | | | |
| Date | | Sep. 18，2020 | Type | | √ Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Review the concepts of class, object, inheritance, overriding and overloading in the C++ programming language    2. Learn the Data Abstraction Principle, the Open-Closed Principle, and the Subclass Substitution Rule    3. Practice C++ programming skills | | | | | |
| 1. **Experimental environment (**platform and software**)**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results) 2. Task 1    1. Declare and define the Employee and Company classes. The requirements are as stated in the slides;    2. Declare and define the Employee and Company classes for the case of hourly paid employee by using inheritage. The requirements are as stated in the slides and specified in the given source files:       1. Implement the input() method for the derived Employee2 class       2. Implement the findBestPaid() method for the derived Company2 class    3. Test your classes implementation to get and output the best paid employee 3. Source Code   1. employee1.h  #ifndef EMPLOYEE  #define EMPLOYEE  #include <string> *// declares string class*  using namespace std;  class Employee  {      protected:  *//private:*          string name;          double grossPay;      public:  *// Postcondition: this employee's name has been set to ""*  *//                and gross pay to 0.00.*          Employee();  *// Postcondition: The name and gross pay of this Employee have*  *//                been read in.*          virtual void input();  *// Postcondition: this Employee's name and gross pay have been*  *//                written out.*          void output() const;  *// Postcondition: this Employee contains a copy of otherEmployee.*          void copy(const Employee& otherEmployee);  *// Postcondition: true has been returned if this Employee's gross*  *//                pay is greater than that of otherEmployee.*  *//                Otherwise, false has been returned.*          bool compare(const Employee& otherEmployee) const;          string getName() const;  };*// Employee*  #endif  2 .employee2.h  #ifndef EMPLOYEE2  #define EMPLOYEE2  #include "employee1.h"  class Employee2:public Employee  {  private:      int month;      double monthlySalary;  public:      Employee2();      void input();  };  #endif  3.company1.h  #ifndef COMPANY  #define COMPANY  #include "employee1.h" *// declares Employee class*  class Company  {        protected:  *//private:*        Employee bestPaid;          public:        Company();        void findBestPaid();        void printBestPaid() const;  };*// Company*  #endif  4.company2.h  #ifndef COMPANY2  #define COMPANY2  #include "company1.h"  class Company2:public Company  {  public:      void findBestPaid();  };  #endif  5.employee1.cpp  #include <iostream>  #include "employee1.h"  Employee::Employee()  {     name = "";     grossPay = 0;  }*// default constructor*  void Employee::input()  {     cout << "Please enter a name and gross pay; to quit, enter \* followed by any number: ";     cin >> name >> grossPay;  }*// input*  void Employee::output() const  {     cout << "is: " << name << ",  $" << grossPay << endl;  }*// output*  void Employee::copy(const Employee &otherEmployee)  {     name = otherEmployee.name;     grossPay = otherEmployee.grossPay;  }*// copy*  bool Employee::compare(const Employee &otherEmployee) const  {     return grossPay > otherEmployee.grossPay;  }*// compare*  string Employee::getName() const  {     return name;  }    6.employee2.cpp  #include "employee2.h"  #include <iostream>  Employee2::Employee2()  {      month = 0;      monthlySalary = 0;  }  void Employee2::input()  {      cout << "Please enter a name and monthlySalary and month; to quit, enter \* followed by any number: ";      cin >> name >> monthlySalary >> month;      grossPay = monthlySalary \* month;  *//please implement this*  }  7.company1.cpp    #include <iostream>  #include "company1.h"  #include "employee1.h"  *//comment: ctrl+k ctrl+c*  *//uncomment: ctrl+k ctrl+u*  Company::Company()  {     bestPaid = Employee();  }*//default constructor*  void Company::findBestPaid()  {     Employee employee;     employee.input();     while (employee.getName() != "\*")     {        if (employee.compare(bestPaid))           bestPaid.copy(employee);        employee.input();     }*// while*  }*// findBestPaid*  void Company::printBestPaid() const  {     cout << "The best-paid employee (and gross pay) ";     bestPaid.output();  }*// printBestPaid*  8.company2.cpp  #include "company2.h"  #include "employee2.h"  void Company2::findBestPaid()  {      Employee2 employee\_1;      employee\_1.input();      while (employee\_1.getName() != "\*")      {          if (employee\_1.compare(bestPaid))              bestPaid.copy(employee\_1);          employee\_1.input();      }  *//please implement this*  }  9.BestPaidEmployee.cpp  *// BestPaidEmployee.cpp : This file contains the 'main' function. Program execution begins and ends there.*  *//*  #include <iostream>  #include "company1.h"  #include "employee1.h"  #include "company2.h"  #include "employee2.h"  *//#include "company2.h"*  *//#include "employee2.h"*  int main()  {      Company cmp;      cmp.findBestPaid();      cmp.printBestPaid();      Company2 cmp2;      cmp2.findBestPaid();      cmp2.printBestPaid();  }   1. Result | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Analysis:  According to the requirement, I should write definition of “Company2::findBestPaid()”and “void Employee2::input()”.  On the one hand, I create a object of employee2 and reuse the method of the base class. On the other hand, I conduct month and  monthlySalary to get the grossPay.  Harvest:  I have a deeper understanding in object-oriented programming and Polymorphism. In this experiment, I used to have a problem with the “input” method. I can’t tell that this is because of hide, overload or Polymorphism. After I look up some reference books, I can tell that  Hide:  When calling a member function of a class, the compiler will look up the definition of the function step by step along the inheritance chain of the class. If it finds it, it will stop the search, so if a derived class and a base class have the same The function of the same name (regardless of whether the parameters are the same), and the compiler finally chooses the function in the derived class, then we say that the member function of this derived class "hides" the member function of the base class.  Overload:  It must be in a domain, with the same function name but different function parameters. The role of overloading is that the same function has different behaviors. Therefore, functions that are not in a domain cannot constitute an overload. This is an important feature of overloading.  Polymorphism:  When we use the reference or pointer of the base class to call a method defined in the base class, because of the existence of dynamic type and static type, we do not know which object it points to, depending on the object bound by the reference and pointer The true type. If the function is a virtual function, it will not decide which method to execute until runtime.  Problems:  At first, I was stuck by the input. I wonder how to let the program determine that which method I refer to in one prompt . | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |