C++ course – Exercises Set 8

Wouter Verkerke (Jan 2023)

Exercise 8.1 – Inheritance

The goal of this exercise is to write a class representing a `Manager` *through inheritance from an existing class* `Employee`

Step 1 – Write class Manager

- Copy the file `ex8.1/Employee.hh`, look at the class and understand all the features.

- Create a small main program, instantiate an `Employee` and print it information by calling its `businessCard()` function. Also print out the salary information accessed through the `salary()` member function in the main program.

- Write a new class `Manager` that inherits from class `Employee`. The class `Manager` should have a `set<Employee*> subordinates` as an additional data member, which represents of subordinates that the manager manages.
    - Why is a `set` a better choice than e.g. a `list` or `vector` to keep track of a collection of subordinates?

- Add the following accessor/modifier functions for the employee set of class `Manager`
    - void `addSubordinate(Employee& empl)` – A method that adds a subordinate employee to the subordinates list ;
    - `const set<Employee*>& listOfSubordinates() const` – A method that returns a const reference to the employee managed by this manager

- Extend your main program such that it also creates a `Manager` object. Print the manager information using the `businessCard()` function and also print its salary information retrieved through `salary()`.

- Does in your opinion the manager object behave *exactly* like the employee object as long as you only refer to the employee-defined properties of both (such as the business card)?

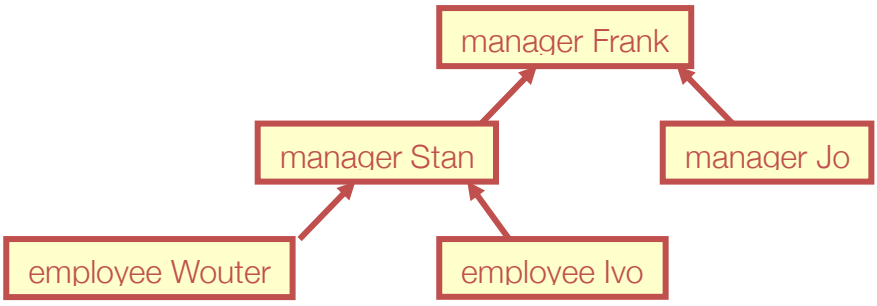
Step 2 - Write a better business card method for class Manager

- Implement the function void `businessCard(ostream& os = cout) const` in class `Manager`. The idea is that this business card function writes a better version that supersedes the employee-style business card.
    - The idea is that the manager-style card will be like the employee-style card *plus* some extra information.
    - So, first call the employee business card function inside the manager business card function.
      The name of the employee-style function is the same as that of the manager-style function, i.e. `businessCard()`, but use the scope operator we can be specific: call `Employee::businessCard()` within the `businessCard()` implementation of `Manager`.
    - After that call, add code to the `businessCard()` implementation of Manager that prints the set of managed employees. Use an iterator to go through the set of employees.

- Rerun the main program and see if the managers business card is printed out in the new style

Step 3 – Creating a hierarchy of manager and employees

- Enter the following personnel hierarchy in the main program, using classes `Employee`, `Manager`, and function `Manager::addSubordinate()`



- Do you have problems entering a `Manager` as `Employee` in function `Manager::addSubordinate()`? *Why is(n't) that?*

- Print out everybody's business card to verify that you entered the personnel hierarchy correctly