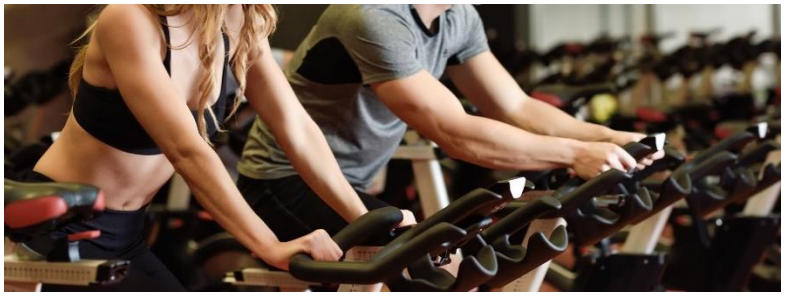
IECS350 軟體框架設計

08 物件導向程式設計練習

[Inheritance Polymorphism]



Programming Practice #5

Exploring the Similarities Between Class BasePlusCommissionEmployee and Class CommissionEmployee:
Most of the code for class BasePlusCommissionEmployee is similar, if not identical, to the code for class CommissionEmployee.

Please use an inheritance hierarchy containing types of employees in a company's payroll application to describe the relationship between a base class and a derived class.
Commission employees (who will be represented as objects of a base class) are paid a percentage of their sales, while base-salaried commission employees (who will be represented as objects of a derived class) receive a base salary plus a percentage of their sales.

Programming Practice #6

A company pays its employees weekly.

The employees are of three types:

- ✓ ***Salaried employees are paid a fixed weekly salary,***
- ✓ ***Commission employees are paid a percentage of their sales, and***
- ✓ ***Base-salary-plus-commission employees receive a base salary plus a percentage of their sales.***

For the current pay period, the company has decided to reward base-salary-plus-commission employees by adding 10 percent to their base salaries.

The company wants to implement a C++ program that performs its payroll calculations polymorphically.

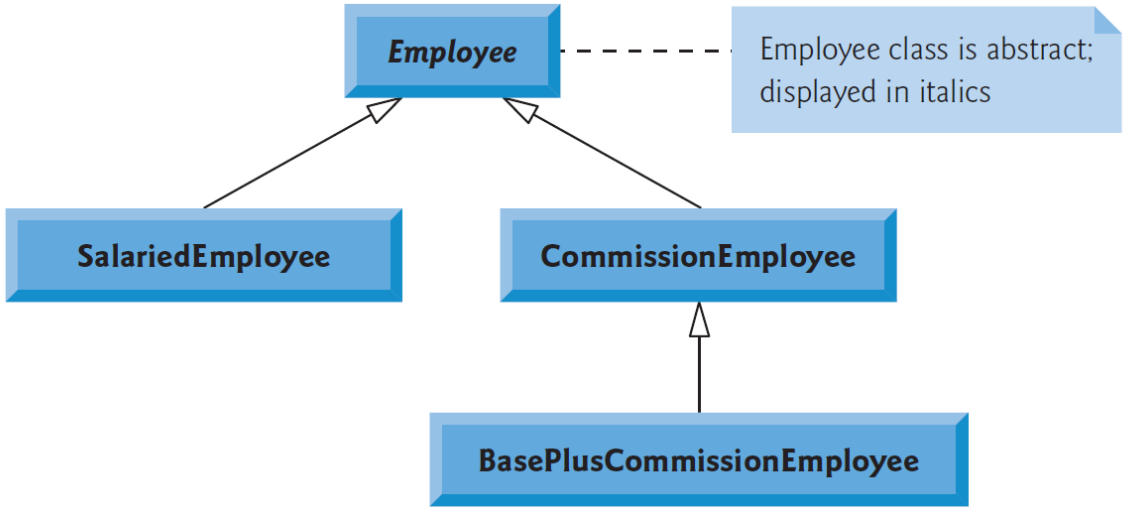
Programming Practice #6

Please use abstract class Employee to represent the general concept of an employee. The classes that derive directly from Employee are SalariedEmployee and CommissionEmployee.

Class BasePlusCommissionEmployee—derived from CommissionEmployee—represents the last employee type.

Abstract base class Employee declares the "interface" to the hierarchy—that is, the set of member functions that a program can invoke on all Employee objects.

Each employee, regardless of the way his or her earnings are calculated, has a first name, a last name and a social security number, so private data members firstName, lastName and socialSecurityNumber appear in abstract base class Employee.



Programming Practice #6

Polymorphic interface for the Employee hierarchy classes.

```
// pure virtual function makes Employee an abstract base class
virtual double earnings() const = 0; // pure virtual
virtual void print() const; // virtual
```

	earnings	print
Employee	= 0	*firstName lastName* social security number: *SSN*
Salaried-Employee	*weeklySalary*	salaried employee: *firstName lastName* social security number: *SSN* weekly salary: *weeklySalary*
Commission-Employee	*commissionRate* * *grossSales*	commission employee: *firstName lastName* social security number: *SSN* gross sales: *grossSales*; commission rate: *commissionRate*
BasePlus-Commission-Employee	(*commissionRate* * *grossSales*) + *baseSalary*	base-salaried commission employee: *firstName lastName* social security number: *SSN* gross sales: *grossSales*; commission rate: *commissionRate*; base salary: *baseSalary*