**CASE STUDY PAYROLL SYSTEM**

**Nội dung đoạn code :**

* **Header file :**

#pragma once

#include <string>

#include<iostream>

using namespace std;

// lớp trừu tượng Employee

class Employee

{

private:

string firstName;

string lastName;

string socialSecurityNumber;

public:

Employee(const string&, const string&, const string&);//constructor

void setFirstName(const string&);//cập nhập FirstName;

string getFirstName() const;//xuất thông tin FirstName

void setLastName(const string&);

string getLastName() const;

void setSocialSecurityNumber(const string&);

string getSocialSecurityNumber() const;

virtual double earnings() const = 0;//hàm thuần ảo

virtual void print() const;

};

//tạo lớp SalariedEmployee kế thừa từ lớp Employee "public"

class SalariedEmployee : public Employee

{

private:

double weeklySalary;

public:

SalariedEmployee() : Employee("", "", ""), weeklySalary(0.0) {}//constructor mặc định

SalariedEmployee(const string&, const string&, const string&, double = 0.0);//constructor nếu nhập giá trị

void setWeeklySalary(double);

double getWeeklySalary() const;

virtual double earnings() const;

virtual void print() const;

void sinhSEmployee();//sinh ra ngẫu nhiên thông tin của SalariedEmployee

};

//tạo lớp CommissionEmployee kế thừa từ lớp Employee "public"

class CommissionEmployee : public Employee

{

private:

double grossSales;

double commissionRate;

public:

CommissionEmployee() : Employee("", "", ""), grossSales(0.0), commissionRate(0.0){}//constructor mặc định

CommissionEmployee(const string&, const string&, const string&, double = 0.0, double = 0.0);//constructor nếu nhập giá trị

void sinhCEmployee();//sinh thông tin ngẫu nhiên của CommissionEmployee

void setGrossSales(double);

double getGrossSales() const;

void setCommissionRate(double);

double getCommissionRate() const;

virtual double earnings() const;

virtual void print() const;

};

//tạo lớp BasePlusCommissionEmployee kế thừa từ lớp CommissionEmployee "public"

class BasePlusCommissionEmployee : public CommissionEmployee

{

private:

double baseSalary;

public:

BasePlusCommissionEmployee() : CommissionEmployee("","","",0.0,0.0) ,baseSalary(0.0){}

BasePlusCommissionEmployee(const string&, const string&, const string&, double = 0.0, double = 0.0, double = 0.0);

void setBaseSalary(double);

double getBaseSalary() const;

void sinhBPCEmployee();////sinh thông tin ngẫu nhiên của BasePlusCommissionEmployee

virtual double earnings() const;

virtual void print() const;

};

* **Resource file :**

#include"salary.h"

#include <cstdlib>// su dung srand

#include <ctime>// cho time

// Implementation of Employee class

// Hiện thực hóa lớp Employee

Employee::Employee(const string& first, const string& last, const string& ssn)

: firstName(first), lastName(last), socialSecurityNumber(ssn) {}

// Constructor của lớp Employee, khởi tạo họ, tên và số Bảo hiểm xã hội (SSN)

void Employee::setFirstName(const string& first) {

firstName = first;

}

// Thiết lập họ của nhân viên

string Employee::getFirstName() const {

return firstName;

}

// Lấy họ của nhân viên

void Employee::setLastName(const string& last) {

lastName = last;

}

// Thiết lập tên của nhân viên

string Employee::getLastName() const {

return lastName;

}

// Lấy tên của nhân viên

void Employee::setSocialSecurityNumber(const string& ssn) {

socialSecurityNumber = ssn;

}

// Thiết lập số Bảo hiểm xã hội (SSN) của nhân viên

string Employee::getSocialSecurityNumber() const {

return socialSecurityNumber;

}

// Lấy số Bảo hiểm xã hội (SSN) của nhân viên

void Employee::print() const {

cout << getFirstName() << ' ' << getLastName()

<< "\nsocial security number: " << getSocialSecurityNumber();

}

// In ra họ, tên và số Bảo hiểm xã hội (SSN) của nhân viên

// Implementation of SalariedEmployee class

// Hiện thực hóa lớp SalariedEmployee

SalariedEmployee::SalariedEmployee(const string& first, const string& last, const string& ssn, double salary)

: Employee(first, last, ssn), weeklySalary(salary) {

setWeeklySalary(salary);

}

// Constructor của lớp SalariedEmployee, khởi tạo mức lương hàng tuần

void SalariedEmployee::setWeeklySalary(double salary) {

if (salary >= 0.0)

weeklySalary = salary;

else

throw invalid\_argument("Weekly salary must be >= 0.0");

}

// Thiết lập mức lương hàng tuần của nhân viên

double SalariedEmployee::getWeeklySalary() const {

return weeklySalary;

}

// Lấy mức lương hàng tuần của nhân viên

double SalariedEmployee::earnings() const {

return getWeeklySalary();

}

// Tính toán thu nhập của nhân viên

void SalariedEmployee::print() const {

cout << "salaried employee: ";

Employee::print();

cout << "\nweekly salary: " << getWeeklySalary();

}

// In ra thông tin của nhân viên lương cố định, bao gồm mức lương hàng tuần

// Trong lớp SalariedEmployee

// Phương thức sinh nhân viên lương cố định ngẫu nhiên

void SalariedEmployee::sinhSEmployee() {

string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

string name = "";

string name1 = "";

string soSSN = "";

for (int i = 0; i < 2; i++) {

int random = rand() % letters.length();

name += letters[random];

}

for (int i = 0; i < 2; i++) {

int random1 = rand() % letters.length();

name1 += letters[random1];

}

for (int i = 0; i < 8; ++i) {

int random2 = rand() % 10;

soSSN += to\_string(random2);

}

setFirstName(name);

setLastName(name1);

setSocialSecurityNumber(soSSN);

int minRange = 400;

int maxRange = 10000;

int interval = 100;

int numIntervals = (maxRange - minRange) / interval;

int randomInterval = rand() % (numIntervals + 1);

int randomSalary = minRange + (randomInterval \* interval);

setWeeklySalary(randomSalary);

SalariedEmployee employee(name, name1, soSSN, randomSalary);

}

// Tạo ngẫu nhiên thông tin của nhân viên lương cố định

// Implementation of CommissionEmployee class

// Hiện thực hóa lớp CommissionEmployee

CommissionEmployee::CommissionEmployee(const string& first, const string& last, const string& ssn, double sales, double rate)

: Employee(first, last, ssn) {

setGrossSales(sales);

setCommissionRate(rate);

}

// Constructor của lớp CommissionEmployee, khởi tạo doanh số bán hàng và tỷ lệ hoa hồng

void CommissionEmployee::sinhCEmployee() {

string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

string name = "";

string name1 = "";

string soSSN = "";

for (int i = 0; i < 2; i++) {

int random = rand() % letters.length();

name += letters[random];

}

for (int i = 0; i < 2; i++) {

int random1 = rand() % letters.length();

name1 += letters[random1];

}

for (int i = 0; i < 8; ++i) {

int random2 = rand() % 10;

soSSN += to\_string(random2);

}

setFirstName(name);

setLastName(name1);

setSocialSecurityNumber(soSSN);

int minRange = 400;

int maxRange = 10000;

int interval = 100;

int numIntervals = (maxRange - minRange) / interval;

int randomInterval = rand() % (numIntervals + 1);

int randomGrossSales = minRange + (randomInterval \* interval);

float randomRate = ((rand() % 100) + 1) / 100.0; // Sinh ngẫu nhiên từ 0.01 đến 1.0

setGrossSales(randomGrossSales);

setCommissionRate(randomRate);

CommissionEmployee employee(name, name1, soSSN, randomGrossSales, randomRate);

}

// Tạo ngẫu nhiên thông tin của nhân viên hoa hồng

void CommissionEmployee::setGrossSales(double sales) {

if (sales >= 0.0)

grossSales = sales;

else

throw invalid\_argument("Gross sales must be >= 0.0");

}

// Thiết lập doanh số bán hàng của nhân viên

double CommissionEmployee::getGrossSales() const {

return grossSales;

}

// Lấy doanh số bán hàng của nhân viên

void CommissionEmployee::setCommissionRate(double rate) {

if (rate >= 0.0 && rate <= 1.0)

commissionRate = rate;

else

throw invalid\_argument("Commission rate must be >= 0.0 and < 1.0");

}

// Thiết lập tỷ lệ hoa hồng của nhân viên

double CommissionEmployee::getCommissionRate() const {

return commissionRate;

}

// Lấy tỷ lệ hoa hồng của nhân viên

double CommissionEmployee::earnings() const {

return getCommissionRate() \* getGrossSales();

}

// Tính toán thu nhập của nhân viên

void CommissionEmployee::print() const {

cout << "commission employee: ";

Employee::print();

cout << "\ngross sales: " << getGrossSales() << "; commission rate: " << getCommissionRate();

}

// In ra thông tin của nhân viên hoa hồng, bao gồm doanh số bán hàng và tỷ lệ hoa hồng

// Implementation of BasePlusCommissionEmployee class

// Hiện thực hóa lớp BasePlusCommissionEmployee

BasePlusCommissionEmployee::BasePlusCommissionEmployee(const string& first, const string& last, const string& ssn, double sales, double rate, double salary)

: CommissionEmployee(first, last, ssn, sales, rate) {

setBaseSalary(salary);

}

// Constructor của lớp BasePlusCommissionEmployee, khởi tạo mức lương cơ bản

void BasePlusCommissionEmployee::sinhBPCEmployee() {

// Tạo một đối tượng của lớp CommissionEmployee

CommissionEmployee commissionEmployee;

// Gọi phương thức sinhCEmployee() trên đối tượng đó

commissionEmployee.sinhCEmployee();

// Lấy thông tin đã được tạo tự nhiên từ hàm sinhCEmployee() của lớp CommissionEmployee

string firstName = commissionEmployee.getFirstName();

string lastName = commissionEmployee.getLastName();

string ssn = commissionEmployee.getSocialSecurityNumber();

double randomgrossSales = commissionEmployee.getGrossSales();

double randomrate = commissionEmployee.getCommissionRate();

// Tạo giá trị cho baseSalary

int minRange = 400;

int maxRange = 10000;

int interval = 100;

int numIntervals = (maxRange - minRange) / interval;

int randomInterval = rand() % (numIntervals + 1);

int randomBaseSalary = minRange + (randomInterval \* interval);

setBaseSalary(randomBaseSalary);

// Đặt giá trị baseSalary cho đối tượng BasePlusCommissionEmployee đã được tạo trước đó

setBaseSalary(randomBaseSalary);

setFirstName(firstName);

setLastName(lastName);

setSocialSecurityNumber(ssn);

setGrossSales(randomgrossSales);

setCommissionRate(randomrate);

}

// Tạo ngẫu nhiên thông tin của nhân viên hoa hồng với mức lương cơ bản

void BasePlusCommissionEmployee::setBaseSalary(double salary) {

if (salary >= 0.0)

baseSalary = salary;

else

throw invalid\_argument("Salary must be >=0.0");

}

// Thiết lập mức lương cơ bản của nhân viên

double BasePlusCommissionEmployee::getBaseSalary() const {

return baseSalary;

}

// Lấy mức lương cơ bản của nhân viên

double BasePlusCommissionEmployee::earnings() const {

return getBaseSalary() + CommissionEmployee::earnings();

}

// Tính toán thu nhập của nhân viên

void BasePlusCommissionEmployee::print() const {

cout << "base-salaried ";

CommissionEmployee::print();

cout << "; base salary: " << getBaseSalary();

}

// In ra thông tin của nhân viên hoa hồng với mức lương cơ bản

* **Source file :**

#include <iostream>

#include <iomanip>

#include <vector>

#include <algorithm> // Dùng để sắp xếp vector

#include "salary.h"

using namespace std;

int main() {

srand(time(NULL));

cout << fixed << setprecision(2);

// Khởi tạo vector chứa 10 đối tượng Employee

vector<Employee\*> employees;

// Tạo 10 đối tượng cho mỗi loại Employee với thông tin ngẫu nhiên

for (int i = 0; i < 10; ++i) {

SalariedEmployee\* salariedEmployee = new SalariedEmployee("","","",0.0);

salariedEmployee->sinhSEmployee();

employees.push\_back(salariedEmployee);

CommissionEmployee\* commissionEmployee = new CommissionEmployee();

commissionEmployee->sinhCEmployee();

employees.push\_back(commissionEmployee);

BasePlusCommissionEmployee\* basePlusCommissionEmployee = new BasePlusCommissionEmployee();

basePlusCommissionEmployee->sinhBPCEmployee();

employees.push\_back(basePlusCommissionEmployee); // Lưu ý: Đã thêm dấu & để truyền địa chỉ của đối tượng

}

// Sắp xếp danh sách theo thứ tự giảm dần của thu nhập

sort(employees.begin(), employees.end(),

[](const Employee\* a, const Employee\* b) {// dùng hàm lambda

return a->earnings() > b->earnings();//nếu đúng thì trả về true

});

// In danh sách ra màn hình

cout << "Danh sach nhan vien theo thu tu giam dan caa thu nhap:\n\n";

for (size\_t i = 0; i < employees.size(); ++i) {

employees[i]->print();

cout << "\nearned $" << employees[i]->earnings() << "\n\n";

}

// Giải phóng bộ nhớ

for (size\_t i = 0; i < employees.size(); ++i) {

delete employees[i];

}

return 0;

}

**Hình ảnh kết quả đoạn code :**

