#include <iostream>

#include <vector>

#include <list>

using namespace std;

//

//// STL - Standart Template Library

//

//int main()

//{

// vector<int>v;

// cout << v.capacity() << endl;

//

// for (size\_t i = 0; i < 15; i++)

// {

// v.push\_back(i + 1); // sona element elave edir

// }

//

// cout << v.capacity() << endl; // yaddasda vector ucun ayirdigi yeri verir

// cout << v.size() << endl; // cari element sayinin verir

//

// for (size\_t i = 0; i < v.size(); i++)

// {

// cout << v[i] << " ";

// }

// cout << endl;

//

// cout << v[2] << endl; // eyni seylerdir

// cout << v.at(2) << endl; // eyni seylerdir

//

//

//

// vector<int>v2;

// v2.push\_back(10);

// v2.push\_back(11);

// v2.push\_back(12);

//

// for (int i = 0; i < v2.size(); i++)

// {

// cout << v2.at(i) << " ";

// }

// cout << endl;

//

// v2.insert(v2.begin() + 1, 555);

//

// // v2.begin() ilk element

// // v2.end() son element

//

// for (int i = 0; i < v2.size(); i++)

// {

// cout << v2.at(i) << " ";

// }

// cout << endl;

//

// v2.insert(v2.begin() + 1, {44,55,66,77});

//

// for (int i = 0; i < v2.size(); i++)

// {

// cout << v2.at(i) << " ";

// }

// cout << endl;

//

// v2.insert(v2.end() - 1, -10);

//

// for (int i = 0; i < v2.size(); i++)

// {

// cout << v2.at(i) << " ";

// }

// cout << endl;

//

// v2.pop\_back(); // sonuncu elementi silir

// v2.pop\_back(); // sonuncu elementi silir

//

// for (int i = 0; i < v2.size(); i++)

// {

// cout << v2.at(i) << " ";

// }

// cout << endl;

//

// v2.clear(); // vectoru silir

//

// vector<int>v3;

// v3.push\_back(10);

// v3.push\_back(11);

// v3.push\_back(12);

// v3.push\_back(13);

// v3.push\_back(14);

// v3.push\_back(15);

//

// for (int i = 0; i < v3.size(); i++)

// {

// cout << v3.at(i) << " ";

// }

// cout << endl;

//

// int index = 0;

// cout << "Enter index : ";

// cin >> index;

//

// v3.erase(v3.begin() + index);

//

// for (int i = 0; i < v3.size(); i++)

// {

// cout << v3.at(i) << " ";

// }

// cout << endl;

//

// v3.clear();

// cout << "Elements are deleted" << endl;

//

// if (v3.empty())

// {

// cout << "No Elements" << endl;

// }

// else

// {

// for (int i = 0; i < v3.size(); i++)

// {

// cout << v3.at(i) << " ";

// }

// cout << endl;

// }

//

// vector<int>v4;

// v4.push\_back(10);

// v4.push\_back(11);

// v4.push\_back(12);

// v4.push\_back(13);

// v4.push\_back(14);

//

// cout << v4.capacity() << endl;

// v4.shrink\_to\_fit(); // bacardigi qeder kenarda ayirdigi yerleri atir

// cout << v4.capacity() << endl;

//

//

//

//

// vector<int>v7;

//

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

// {

// v7.push\_back(i + 1);

// }

//

// for (auto i = v7.begin(); i != v7.end(); i++)

// {

// cout << \*i << endl; // i iteratordur

// }

// //|

// //eyni seydirler

// //|

// //Bunun arxasinda isleyir

//

// for (auto i : v7)

// {

// cout << i << endl;

// }

// cout << "Capacity : " << v7.capacity() << endl;

// v7.reserve(200); // yer ayirir

// cout << "Capacity : " << v7.capacity() << endl;

//

// v7.resize(5); // olcusunu deyisir artiq olanlari artir

//

// for (auto i : v7)

// {

// cout << i << endl;

// }

//

// return 0;

//}

//

//int main()

//{

// // double linked list mentiqi ile isleyir

// list<int> mylist;

//

// for (int i = 0; i < 5; i++)

// {

// mylist.push\_back(i + 10);

// }

//

// //for (auto i = mylist.begin(); i != mylist.end(); i++)

// //{

// // cout << \*i << endl;

//

// //}

//

// // : - in

// for (auto item : mylist)

// {

// cout << item << endl;

// }

//

// vector<float> v;

// for (int i = 0; i < 5; i++)

// {

// v.push\_back(i + 20);

// }

//

// mylist.assign(v.begin(), v.end());

// cout << endl;

// cout << endl;

// for (auto item : mylist)

// {

// cout << item << endl;

// }

//

// return 0;

//}

//

//

//class Student

//{

//private:

// int age;

// string name;

// string surname;

//public:

// Student() : age(0), name(""), surname("") {}

//

// Student(const int& age, const string& name, const string& surname)

// {

// SetAge(age);

// SetName(name);

// SetSurname(surname);

// }

//

// int GetAge() const

// {

// return age;

// }

// string GetName() const

// {

// return name;

// }

// string GetSurname() const

// {

// return surname;

// }

//

// void SetAge(const int& age)

// {

// this->age = age;

// }

// void SetName(const string& name)

// {

// this->name = name;

// }

// void SetSurname(const string& surname)

// {

// this->surname = surname;

// }

//

// void Show() const

// {

// cout << "Age : " << age << endl;

// cout << "Name : " << name << endl;

// cout << "Surname : " << surname << endl;

// }

//};

//

//class Group

//{

//private:

// vector<Student> students;

// int student\_count;

//public:

//

//

// //int GetStudentCount() const

// //{

// // return student\_count;

// //}

// //vector<Student> GetStudents() const

// //{

// // return students;

// //}

// //void SetStudentCount(const int& student\_count)

// //{

// // this->student\_count = student\_count;

// //}

// //void Setstudents(vector<Student>& students)

// //{

// // this->students = students;

// //}

//

// void AddStudent(Student& student)

// {

// students.push\_back(student);

// }

//

// void ShowStudents() const

// {

// for (auto student : students)

// {

// student.Show();

// }

// }

//};

//

//int main()

//{

// Student\* s1 = new Student(12, "Name 1", "Surname 1");

// Student\* s2 = new Student(15, "Name 2", "Surname 3");

// Student\* s3 = new Student(14, "Name 3", "Surname 3");

//

// Group students;

// students.AddStudent(\*s1);

// students.AddStudent(\*s2);

// students.AddStudent(\*s3);

// students.ShowStudents();

//

// return 0;

//}

//class Iterator

//{

//private:

// int\* ptr;

//

//public:

// Iterator() : ptr(nullptr)

// {

//

// }

// Iterator(int\* ptr)

// {

// this->ptr = ptr;

// }

//

// Iterator& operator++()

// {

// ptr++;

// return \*this;

// }

// Iterator& operator++(int)

// {

// ptr++;

// return \*this;

// }

// bool operator!=(const Iterator& other)

// {

// return this->ptr != other.ptr;

// }

// bool operator==(const Iterator& other)

// {

// return this->ptr == other.ptr;

// }

// int& operator\*()

// {

// return \*ptr;

// }

//};

//

//class Array

//{

// int\* data;

// int size = 0;

//public:

// Array(int size) : size(size)

// {

// data = new int[size] {};

// for (int i = 0; i < size; i++)

// {

// data[i] = rand() % 30;

// }

// }

//

// Iterator begin()

// {

// return Iterator(data);

// }

// Iterator end()

// {

// return Iterator(data + size);

// }

//

//

//};

//

//void main()

//{

// Array a(10);

//

// for (auto i = a.begin(); i != a.end(); i++)

// {

// cout << \*i << endl;

// }

//

// for (auto i : a)

// {

// cout << i << endl;

// }

//}

#include <assert.h>

class Name

{

private:

string name;

public:

Name() = default;

Name(const string& name)

{

SetName(name);

}

void SetName(const string& name)

{

assert(!name.empty() && "Name should be filled!");

this->name = name;

}

string GetName() const

{

return name;

}

void ShowName() const

{

cout << "Name : " << name << endl;

}

};

class BankCard

{

private:

string long\_code;

string password;

double balance;

string expired\_month;

string expired\_year;

public:

BankCard() = default;

BankCard(const string& longcode, const string& password, const double& balance, const string& expiredmonth, const string& expiredyear)

{

SetLongCode(longcode);

SetPassword(password);

SetBalance(balance);

SetExpiredMonth(expiredmonth);

SetExpiredYear(expiredyear);

}

string GetLongCode() const

{

return long\_code;

}

string GetPassword() const

{

return password;

}

double GetBalance() const

{

return balance;

}

string GetExpiredMonth() const

{

return expired\_month;

}

string GetExpiredYear() const

{

return expired\_year;

}

void SetLongCode(const string& longcode)

{

assert(!longcode.empty() && "Long code must be filled!");

assert(longcode.length() > 8 && "Long code length must be at least 8!");

this->long\_code = longcode;

}

void SetPassword(const string& password)

{

assert(!password.empty() && "Password must be filled!");

assert(password.length() > 8 && "Password length must be at least 8!");

this->password = password;

}

void SetBalance(const double& balance)

{

assert(balance >= 0 && "Balance cannot be less than 0!");

this->balance = balance;

}

void SetExpiredMonth(const string& expiredmonth)

{

assert(!expiredmonth.empty() && "Expired month must be filled!");

this->expired\_month = expiredmonth;

}

void SetExpiredYear(const string& expiredyear)

{

assert(!expiredyear.empty() && "Expired year must be filled!");

this->expired\_year = expiredyear;

}

void ShowBankCard() const

{

cout << "=== BANK CARD INFO === " << endl;

cout << "Long Code : " << GetLongCode() << endl;

cout << "Password : ";

int l = GetPassword().length();

for (int i = 0; i < l; i++)

{

cout << "\*";

}cout << endl;

cout << "Balance : " << GetBalance() << endl;

cout << "Expired month : " << GetExpiredMonth() << endl;

cout << "Expired year : " << GetExpiredYear() << endl;

}

};

class Human : public Name

{

private:

Name\* name;

string surname;

BankCard\* bankcard;

public:

Human() = default;

Human(const string& name, const string& surname, BankCard\* bankcard) : Name(name)

{

SetSurname(surname);

}

string GetSurname() const

{

return surname;

}

void SetSurname(const string& surname)

{

assert(!surname.empty() && "Surname should be filled!");

this->surname = surname;

}

void ShowHuman() const

{

Name::ShowName();

cout << "Surname : " << surname << endl;

bankcard->ShowBankCard();

}

};

class Group

{

private:

string name;

public:

void ShowGroup() const

{

}

};

class Teacher : public Human

{

private:

int my\_id;

int age;

vector<Group> groups;

string department;

double salary;

BankCard\* bankcard;

public:

Teacher() = default;

Teacher(const int& id, const int& age, const vector<Group> groups, const string& department, const double& salary, BankCard\*& bankcard)

{

SetMyID(id);

SetAge(age);

SetGroups(groups);

SetDepartment(department);

SetSalary(salary);

SetBankCard(bankcard);

}

int GetMyID() const

{

return my\_id;

}

int GetAge() const

{

return age;

}

vector<Group> GetGroups() const

{

return groups;

}

string GetDepartment() const

{

return department;

}

double GetSalary() const

{

return salary;

}

BankCard\* GetBankCard() const

{

return bankcard;

}

void SetMyID(const int& id)

{

assert(id > 0 && "ID cannot be less than 0!");

this->my\_id = id;

}

void SetAge(const int& age)

{

assert(age > 0 && "Age cannot be less than 0!");

this->age = age;

}

void SetGroups(const vector<Group> groups)

{

this->groups = groups;

}

void SetDepartment(const string& department)

{

assert(!department.empty() && "Department name should be filled!");

this->department = department;

}

void SetSalary(const double& salary)

{

assert(salary > 0 && "Salary must be greater than 0!");

this->salary = salary;

}

void SetBankCard(BankCard\*& bankcard)

{

this->bankcard = bankcard;

}

void ShowTeacher() const

{

Human::ShowHuman();

cout << "ID : " << GetMyID() << endl;

cout << "Age : " << GetAge() << endl;

for (auto group : groups)

{

group.ShowGroup();

}

cout << "Department : " << GetDepartment() << endl;

cout << "Salary : " << GetSalary() << endl;

}

};

class Exam

{

private:

public:

void ShowExam() const

{

}

};

class Student : public Human

{

private:

int my\_id;

string speciality;

double score;

double money;

BankCard\* bankcard;

vector<Exam> exams;

public:

Student() = default;

Student(const int& id, const string& speciality, const double& score, const double& money, BankCard\*& bankcard, const vector<Exam> exams)

{

SetMyID(id);

SetScore(score);

SetMoney(money);

SetSpeciality(speciality);

SetExams(exams);

SetBankCard(bankcard);

}

int GetMyID() const

{

return my\_id;

}

double GetMoney() const

{

return money;

}

double GetScore() const

{

return score;

}

string GetSpeciality() const

{

return speciality;

}

vector<Exam> GetExams() const

{

return exams;

}

BankCard\* GetBankCard() const

{

return bankcard;

}

void SetMyID(const int& id)

{

assert(id > 0 && "ID cannot be less than 0!");

this->my\_id = id;

}

void SetScore(const double& score)

{

assert(score >= 0 && "Score cannot be less than 0!");

assert(score <= 100 && "Score cannot be more than 100!");

this->score = score;

}

void SetMoney(const double& money)

{

assert(money >= 0 && "Money cannot be less than 0!");

this->money = money;

}

void SetSpeciality(const string& speciality)

{

assert(!speciality.empty() && "Speciality name should be filled!");

this->speciality = speciality;

}

void SetExams(const vector<Exam> exams)

{

this->exams = exams;

}

void SetBankCard(BankCard\*& bankcard)

{

this->bankcard = bankcard;

}

void ShowStudent() const

{

Human::ShowHuman();

cout << "ID : " << GetMyID() << endl;

cout << "Money : " << GetMoney() << endl;

cout << "Score : " << GetScore() << endl;

for (auto exam : exams)

{

exam.ShowExam();

}

cout << "Speciality : " << GetSpeciality() << endl;

}

};

class University

{

private:

int id;

string name;

vector<Teacher> teachers;

vector<Student> students;

};