Lớp đối tượng: Entitites

Khi khai báo ko dùng từ khóa static

Bao gồm:

* Thành phần mang thông tin
* Thành phần xử lý thông tin

Khai báo:

* Public: có thể truy cập thoải mái
* Private: chỉ có thể truy cập bằng các phần xử lý của lớp đối tượng này (thông thường, thành phần mang thông tin sẽ để private để giấu)

Cố gắng sử dụng private (ví dụ các hàm kiểm tra tính hợp lệ của dữ liệu)

Những điều cần lưu ý khi làm bài LTHDT:

* Đọc đề và phân tích:
  + Xác định đối tượng có những thông tin nào, bao gồm các hàm xử lý của đối tượng
  + Xác định cách thức lưu trữ (nếu đề yêu cầu)
  + Xác định giao diện người dùng, phương thức nhập (console hay web)
  + Xác định những hàm xử lý nghiệp vụ
* Đi vào chi tiết code cho từng phần:
  + Struct: các thành phần chứa thông tin -> { get; set; }, lưu ý hàm tạo xem có thông tin nào cần kiểm tra hoặc xử lý, các hàm của đối tượng nếu cần

Ví dụ: PhanSo -> thông tin TuSo, MauSo. Hàm tạo public PhanSo(int tuso, int mauso) có kiểm tra mẫu số khác 0 (nếu console thì cho người dùng nhập lại, nếu web thì xuất lỗi). Một số hàm như public PhanSo Cong(PhanSo B), public bool LonHon(PhanSo B)

* + Lưu trữ: tùy vào độ phức tạp mà chọn 1 trong 2 cách lưu trữ bên dưới

Ví dụ: đề cho đọc 2 phân số từ file (dạng a/b, mỗi phân số là 1 dòng) thì áp dụng cách đơn giản. Đề cho đọc danh sách sản phẩm, hóa đơn, sinh viên… chứa nhiều thông tin -> chọn cách Json.

* + Giao diện người dùng: nếu đề không yêu cầu nên chọn Console để code nhanh hơn, còn nếu đã yêu cầu dạng web thì phải viết theo dạng web.

Lưu ý cho dạng web:   
- Xác định các dữ liệu được submit

- Xác định dữ liệu trả về

- Cách hiển thị dữ liệu trả về

- Các trường hợp xảy ra nếu lỗi

* + Hàm xử lý nghiệp vụ: cầu nối giữa main program và database, tùy thuộc yêu cầu đề bài sẽ biến hóa khác nhau. Lưu ý: cài đặt Interface để sử dụng hàm

Ví dụ: Cộng 2 phân số  
- Dạng 1: nhập vào 2 phân số (giao diện web), tính tổng và xuất kết quả ra file

Main program: phần OnPost() nhận dữ liệu 2 phân số, tạo 2 phân số bằng PhanSo A = new PhanSo(TuSo1, MauSo1) , sau đó gọi hàm XuLy.TinhTong(A,B)

Hàm xử lý: void TinhTong(PhanSo A, PhanSoB) có nội dung: tạo PhanSo S = A.Cong(B); và dung hàm LuuTru.Xuat(S) để xuất ra file.

- Dạng 2: đọc 2 phân số từ file, tính tổng và xuất kết quả ra màn hình web

Main Program: phần OnGet() gọi hàm Ketqua = XuLy.TinhTong() với Ketqua sẽ được xuất ra trang web bằng @Model.Ketqua

Hàm xử lý: string TinhTong() có nội dung: Đọc 2 phân số List<PhanSo> PS = LuuTru.Doc(); tính tổng PhanSo S = PS[1].Cong(PS[2]), và return S.Xuat() (với hàm Xuat() trong phần đối tượng để chuyển PhanSo thành dạng chuỗi (ví dụ PhanSo A có TuSo = 1, MauSo = 2, chạy A.Xuat() -> chuỗi “1/2”

* + Lưu ý chung về Tính kế thừa: thông thường đề bài sẽ cho 2-3 loại đối tượng có vài điểm chung (ví dụ: NhanVien -> NhanVienVP, NhanVienSX. HoaDon -> HoaDonNhap, HoaDonXuat) thì khi tạo đối tượng, các ham xử lý sẽ có Tính kế thừa, ưu tiên dùng dạng abstract vì thường hàm sẽ trả về một giá trị nào đó, nếu dùng virtual sẽ rắc rối.

Do không thể sử dụng tài liệu, vì thế có những phần code phải học thuộc:

* Code lưu trữ:
* Form, input, table
* OnGet, OnPost, try and catch
* Tính kế thừa, Interface
* Khi nhập nhiều dữ liệu, luôn tạo mới biến chứ không dùng 1 biến để nhập nhiều lần

Code lưu trữ:

* Dạng không dùng json:

using System.IO;

using Entities;

public static void Luu(PhanSo P)

{

StreamWriter file = new StreamWriter("D:\\phanso.txt");

file.Write("{0}/{1}", P.TuSo, P.MauSo);

file.Close();

}

public static List<PhanSo> Doc()

{

List<PhanSo> L = new List<PhanSo>();

StreamReader file = new StreamReader("D:\\phanso.txt");

while (file.EndOfStream == false)

{

string data = file.ReadLine();

string[] s = data.Split("/");

PhanSo P = TaoPhanSo(int.Parse(s[0]), int.Parse(s[1]));

L.Add(P);

}

file.Close();

return L;

}

file.ReadToEnd() để đọc cả file

* Dạng dùng json:

using Entities;

using System.IO;

using Newtonsoft.Json;

public static List<HDnhap> DocHDNhap()

{

StreamReader file = new StreamReader("wwwroot/data/hdnhap.json");

string data = file.ReadToEnd();

file.Close();

List<HDnhap> danhsachHD = JsonConvert.DeserializeObject<List<HDnhap>>(data);

return danhsachHD;

}

public static bool LuuDSNhap(List<HDnhap> danhsachHD)

{

StreamWriter file = new StreamWriter("wwwroot/data/hdnhap.json");

string data = JsonConvert.SerializeObject(danhsachHD);

file.Write(data);

file.Close();

return true;

}

Code cơ bản:

Console:

Chương trình chính:

static void Main(string[] args)

{

Console.WriteLine(1 / 2);

Console.ReadLine();

}

Web:

Trang giao diện:

@page

@model ThiHocKy.Pages.Cau1

@using Entities;

<form method="POST">

<label for="keyword">Nhập từ khóa: </label>

<input type="text" name="keyword" />

<input type="submit" value="Tìm kiếm"/>

</form>

@Model.ThongBao

<table class="table table-striped">

<thead>

<tr>

<th>Tên loại hàng</th>

<th>Số lượng tồn</th>

</tr>

</thead>

<tbody>

@foreach (TonkhoLH t in Model.DSloc)

{

<tr>

<td>@t.TenLH</td>

<td>@t.TonKho</td>

</tr>

}

</tbody>

</table>

Chương trình chính:

namespace ThiHocKy.Pages

{

public class Cau1 : PageModel

{

//Khai báo biến, Interface

public Cau1 ()

{

xuly = new XuLyTonKho(); //Gọi Interface xử lý

}

public void OnGet()

{

ThongBao = String.Empty; //tùy biến theo biến

}

public void OnPost()

{

try

{

//code

}

catch (Exception ex)

{

ThongBao = ex.Message;

}

}

}

}

**Khai báo đối tượng:**

Cơ bản:

public class HocSinh

Đối tượng kế thừa:

public abstract class SinhVien { }

public class SinhVienTaiNang : SinhVien {}

public class SinhVienChinhQuy : SinhVien {}

Lưu ý: khi sử dụng dạng này sẽ không thể dùng hàm tạo SinhVien() để tạo mới mà phải có 1 hàm riêng.   
Ví dụ như trên, ta có khi nhập thông tin SV bao gồm mã, họ tên, loại SV -> XuLySinhVien.ThemSinhVien(thông tin) -> dùng switch để phân ra

switch (loaisv)

{

case 0:

SinhVien st = new SinhVienTaiNang();

//Xử lý gì đó

break;

case 1:

SinhVien sc = new SinhVienChinhQuy();

break;

default:

break;

}

Trường hợp giữa các đối tượng ko có nhập thông tin nào khác, ta có thể dùng hàm nhập chung cho tất cả

Lưu ý: do SinhVien là dạng abstract nên không thể tạo new SinhVien(), trong switch case phải dùng các tên biến khác nhau, nếu dùng chung thì chương trình sẽ báo lỗi đã khai báo dạng này, không thể dùng dạng khác. Một lưu ý khác: nếu khai báo List<SinhVien> mà ko gán giá trị ngay, khai báo 1 List rỗng -> List<SinhVien> = new List<SinhVien>();

**Services:**

public class XuLySinhVien

{

public void ThemSinhVien(Thông tin);

}

**Interface:**

public interface IXuLySinhVien

{

void ThemSinhVien (Thông tin);

}

Khai báo khi sử dụng:

public IXuLySinhVien xuly;

public Cau1()

{

xuly = new XyLySinhVien();

}

Hoặc khi chỉ dùng cho 1 hàm nào đó:

IXuLySinhVien xuly = new XyLySinhVien();

**Service Result:**

public class ServiceResult<T>

{

public bool IsSuccess { get; set; }

public T Data { get; set; }

public string Message { get; set; }

public ServiceResult(bool isSuccess, T data, string message)

{

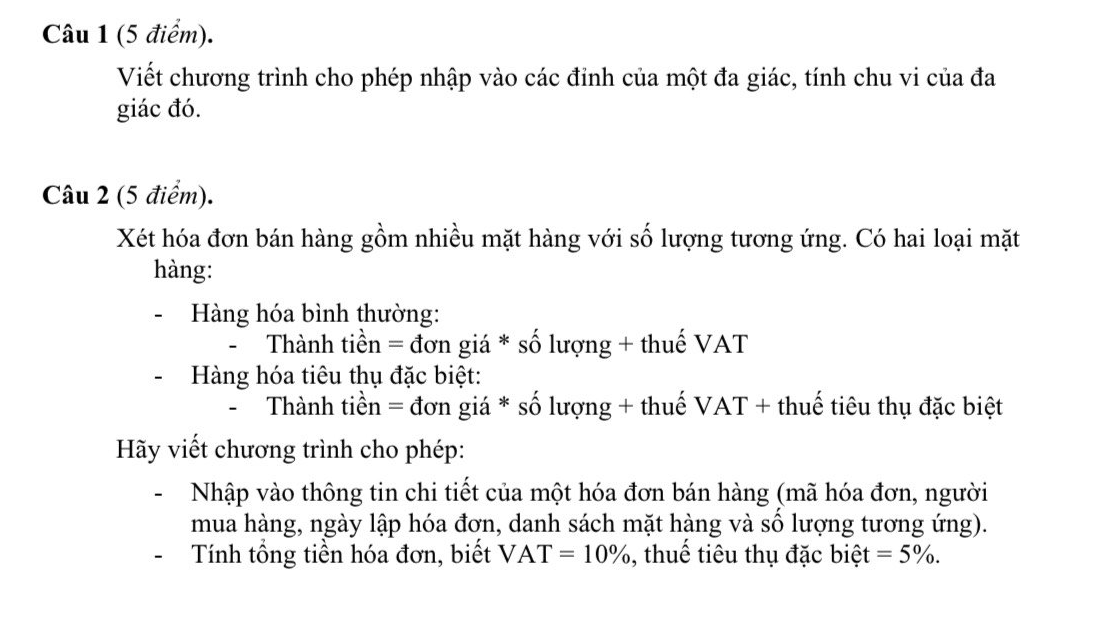
IsSuccess = isSuccess;

Data = data;

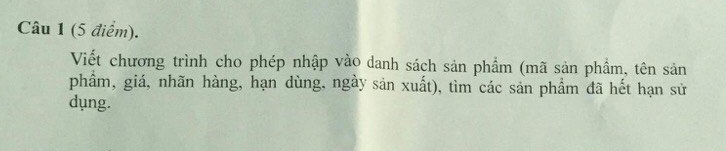
Message = message;

}

}



Bài giải: DaGiac.cs HoaDon.cs



Bài giải: SanPham.cs