1. ***Ôn lại nhanh em của ngày hôm qua – bài cũ***

* Kiến trúc 3 lớp 3 – layer achitecture kh I làm 1 cái app (kết quả) – solution (quá trình làm)
* Slultion

**Project 1 - UI**

* Console app .exe .dll main()
* WPF .EXE .DLL va main() để xuất phát

**Project 2 – REPOSITORIES – cơ sở dữ liệu**

* Class library .dll
* Entities
  + Class liên quan đến table, map xuống table của database

**Project 3 – SERVICES – THAO TÁC TRÊN DỮ LIỆU TRONG RAM**

* + - * CLASS LIBRARY .DLL
        + CLASS CHỨA NHIỀU OBJECT – CABINET/ BOX/ CONTAINER/ BAG
        + List<thứ cần xử lí, lưu trữ - entities: student, lecturer>
        + Crud method
* **Xài mảng, list<???>**
* **Xài generic cabinet<???>**
* **Add reference tới project cần thiết – add dependency (check option copy local = yes)**

============================================================================ĐÀO - PHỞ - PIANO

DATA TYPE - VALUE - VARIABLE

Kiểu dữ liệu - giá trị - biến, định danh, nickname

Tổng kết - - (gọi tên cho 1 điều gì đó)

Tên gọi về hình thái, hình dạng - - identifier

**GIẢI MÃ CÔNG THỨC:**

1. **Công thức #1 về mối liên hệ của bộ 3 huyền thoại – GÓC ĐỘ KHÁI NIỆM**

DATA TYPE | VALUE -> VARIABLE

| VALUE -> VARIABLE

1. **Công thức #1 về mối liên hệ của bộ 3 huyền thoại – GÓC ĐỘ THỰC THI**

Int yob = 2004;

Int curentYear = 2025;

Student s1 = new Student(“se1”, “an ngyễn”, 2004, 8.6); (using constuctor as nomal)

Student s2 = new Student(id: “se1”, name: “an ngyễn”, yob: 2004, gpa: 8.6); (name argument)

Student s3 = new Student() {Id = “se1”, Name = “an ngyễn”, Yob = 2004, Gpa = 8.6}; (object initialize

* + - * Hiểu được mối quan hệ của 3 món trên
      * Hiểu được **DELEGATE** là gì?? ở mức bản chất luôn
* Từ đó ta h iểu được các cái khái niệm liên quan phía sau
* Lập trình hàm – functional programing
* Dependency injection
* Linq ~ stream api bên java!!!
* Anonymous function, Lambda expression
* Call back function

1. **NGỮ PHÁP TIẾNG VIỆT - DANH TỪ LÀ GÌ?**
2. **ĐỊNH NGHĨA DANH TỪ: là 1 từ hay vài từ nói về**

Là 1 từ hay vài từ dùng để đặt tên gọi cho 1 sự vật, hiện tượng, sự kiện nào đó quanh ta

* Ví dụ: con mèo, cơn mưa, bài hát….

1. **Danh từ chung: [tên gọi chung chung]**

\* định nghĩa: là những từ diễn tả khái niệm chung nhất, nói về 1 nhóm các thứ gì đó quanh ta

là tên gọi cho 1 nhóm đối tượng, value, vật thể quanh ta

\* ví dụ: thủ đô, quốc gia, châu lục, hành tinh, đại dương, quốc lộ, ngôi nhà, con người, con chó

1. **Danh từ riêng: [tên gọi cụ thể 1 món gì đó]**

\* định nghĩa: là những từ dùng để gọi tên, đặt tên cho, tên gọi cho 1 thứ cụ thể nào đó quanh ta, là tên, nickname cho 1 thứ cụ thể quanh ta, nó dùng để phân biệt thứ này với thứ khác, xác định rõ ràng duy nhất 1 thứ nào đó ( định danh – định ra tên gọi cho thứ gì đó là identifier(n), identify(n))

\* ví dụ: hà nội, việt nam, châu á, trái đất, thái bình dương, quốc lộ 1a

==🡺 chốt hạ: danh từ chung là tên gọi cho 1 nhóm nhiều thứ => dog person

Danh từ riêng là tên gọi cho 1 thứ duy nhất nào đó => mi lu tí, tèo

1. **BỘ 3 HUYỀN THOẠI : DATA TYPE | VALUE | VARIABLE**
2. **THING, THINGS, VẠN VẬT QUANH TA**

**-** những gì xuất hiện quanh ta thì gọi là vạn vật, thứ gì đó – thing, things (iot)

**-** vạn vật quanh ta đến với ta, trong nhận thức, cách gọi, cách mô tả của con người thì vạn vật có 2 hình thức: Hình thức đơn giản/ định dạng đơn giản, và hình thức phức tạp:

**\* ví dụ**: 5 10 15 20 25 30

đúng/ sai, đậu/ rớt

nguyễn thanh tùng – 1994 – chúng ta của hiện tại

messi-10-barcelona

1. **SIMPLE VALUE/ SINGLE VALUE – GIÁ TRỊ ĐƠN, GIÁ TRỊ ĐƠN GIẢN**

\* định nghĩa: là những con số, những đại lượng, trạng thái nào đó, hình thức thể hiện đơn giản và ngắn gọn, cực ít từ, chữ để viết ra nó!!!

\* ví dụ:

-242 -1 0 5 10 9% 0.3 14.000.000 9.8 3.12 đúng/sai đậu/rớt

1. **COMPLEX VALUE | COMPOSITE VALUE | OBJECT VALUE | [STRUCTURE/STRUCT VALUE]**

giá trị phức tạp, giá trị phức hợp

những thứ mà do nhiều thành phần khác hợp lại tạo thành

* Định nghĩa:
* Ví dụ:

{Khương Đức Anh, 2004, Thanh Hóa, ĐH FPT}

{Nguyễn Thùy Chi, 1993, Hà Nội, Rmit, anh ơi ở lại...}

{Hoàng Thùy Linh, 1988, Hà Nội, Miền đất hứa, để mỵ nói cho mà nghe...}

{taylor allison swift, 1989, us, carilina xa ngoài kia nơi loài tôm hát...}

1. **Quản lí các things/sự vật quanh ta**

khi quanh ta có quá nhiều info, vật, sự vật, hiện tượng, ta cần phải có kĩ thuật tổ chức phân nhóm phân loại chung đặt tên gọi thế nào đó để dễ dàn nhận diện, tìm kiếm, bàn luận, xử lí các thứ đó

**[[[[[[3.1 PHÂN NHÓM DỮ LIỆU, PHÂN NHÓM ĐỐI TƯỢNG, PHÂN LOẠI VALUE]]]]]]**

Phân nhóm dữ liệu, phân loại dữ liệu, chia nhóm dữ liệu, chia nhóm value, classifi/classification/class/group/categorize => là hành động sắp xếp, gom các value có đặc điểm/ thứ tương đồng nhau ra 1 chỗ riêng, cho đám đó 1 cái tên

-----------------------------------------------------------------------

-242 -1 0 5 10 9% 0.3 14.000.000 9.8 3.12 => đều là những con số, đại lượng => 1 nhóm số thực – double , float **[danh từ chung – data type- tên gọi cho 1 đám dữ liệu chung hình dáng]**

đúng/sai đậu/rớt => 1 nhóm nói về 2 trạng thái ngược nhau **=> nhóm bool, boolean [danh từ chung – data type]**

-----------------------------------------------------------------------

-----------------------------------------------------------------------

-242 -1 0 5 10 14.000.000 => đại lượng, con số, nguyên ko lẻ miếng nào

The whole number: int, long, interger, long

9% 0.3 9.8 3.12=> đại lượng, con số, có lẻ, bị sứt mẻ chút chút

Float, double

đúng/sai đậu/rớt: boolean, bool

-----------------------------------------------------------------------

Cách chia nhóm, gom nhóm, đặt tên nhóm: tiêu chí phân loại là con người là đủ ròi, ko bàn về nghề nghiệp

=> nhóm con người, person, class person

=> dữ liệu phức tạp ta cũng phân loại và đặt tên theo tiêu chí nào đó, tên gọi này còn được gọi là class/entity – sự phân h ạng dữ liệu phức tạp

=> danh từ chung dùng cho việc đặt tên gọi nhóm dữ liệu đơn giản, phức tạp => tạo ra đata type

Cách chia nhóm theo tiêu chí nghề nghiệp: single, player

**[[[[[[3.2 ĐẶT TÊN GỌI, NICK NAME CHO TỪNG VALUE CỤ THỂ TRONG CÁI NHÓM ĐÃ ĐẶT TÊN Ở 3.1 ( ĐỊNH DANH ĐỐI TƯỢNG) ]]]]]]**

\* ví dụ:

-242 -1 0 5 10 9% 0.3 14.000.000 9.8 3.12 đúng/sai đậu/rớt

X y z grande pass môn

**\*** ví dụ: {gom các info biểu diễn 1 điều gì đó cụ thể, thing quanh ta}

{Khương Đức Anh, 2004, Thanh Hóa, ĐH FPT}

{Lê Lam, 2004, Thanh Hóa}

{Hoàng Thùy Linh, 1988, Hà Nội, Miền đất hứa}

{Sơn tùng mpt, 1994, Thái bình, chúng ta của hiện tại }

* + - * Sếp = {sơn tùng}

= {người trả lương cho tôi}

**------------------------------------------------------------------------**

1. **Chốt hạ #1: data type kiểu dữ liệu là tên gọi đại diện cho 1 nhóm dữ liệu có định dạng, hình thức danh từ chung**

Nói về data type là nói về rất rất nhiều value đồng dạng, và được gọi chung 1 cái tên

Vd: int, long, double, float, bool, char

person, student, lecturer, dogter, dog, cat, pet, phone, product, food, dringk...

\*\*\*\*\*nhận diện class chính là tìm tên gọi, danh từ riêng, danh từ chung cho 1 đám object, value phức tạp ngoài kia mà share chung đặc điểm, hành vi, chung định dạng!!!!

\*\*\*\*\*góc nhìn lưu trữ data xuống database chính là đi tìm các entity – tên thực thể - erd ra đời

1. **Chốt hạ #2: khai báo biến/variable: là cách đặt tên cho 1 thứ cụ thể nào đó quanh ta: đơn giản, phức tạp**

**Thứ đơn giản này nó là 1 value thuộc về 1 nhóm nào đó = nhóm chính là data type**

1. **Chốt hạ #3: data type tên gọi cung cho cả đám... => danh từ chung**

(nhóm phức tạp còn gọi là class, entity)

**Variable tên gọi riêng cho từng đứa trong đám => danh từ riêng**

1. **Diễn giải câu lệnh dưới đây theo style 3 nhân tố ( bộ 3 huyền thoại )**

Int yob = 2003;

Data type tên gọi value cụ thể

Cho 1 value cụ thể thuộc về tập datatype int

**\* ta nói về cái riêng 2004 yob trong cái chung int**

**\* ta nói về cái chung int và có 1 cái riêng 2003 gọi nó là yob**

int c = 3.14;

int salary = 76.000.000;

float pi = 3.14;

Player r7 = {christianno...}

Datatype variable value cụ thể

Ta nói về các trong đó có 1 bạn và thông tin cụ thể về 1 cầu thủ

Cầu thủ nói cung gọi tắt là cr7

1. **Chốt hạ về template khai báo biến, kiểu dữ liệu, value – declaration of a variable & value**

data-type variable-name = value to specify a unique/specific value/thing

kiểu dữ liệu tên biến = value

Nếu: value phức tạp thì nhớ gom info vào () hoặc {} để thành cụm composite 1 cục thông tin

value đơn giản = luôn

double grade = 10;

Player m10 = {lionel an...}

Player m10 = new Player(...)

Player m10 = new Player(...) {...}

Player m10 = new (...)

player r7 = {christianno...}

player m10 = {lionel an...}

single sếp = {nguyễn thanh tùng, ...}

-------------------------------------------------------------------------------------------------------------------

NGOẠI TRUYỆN: json – javascript object notation

J S O N

Cách biều diễn thông tin của 1 đối tượng/ thứ value phức tạp theo style text dễ đọc kể cả với ngườ ngoại đạo, được dùng trong javascript

{“Id” : “se186606”, “Name” : “Khương Đức Anh”, “yob” : 2004, “city” : “Thanh Hóa”}

=======================================================================

**>>>>>>>>>>MỐI QUAN HỆ GIỮA 3 THỨ HUYỀN THOẠI**

1 DATA TYPE CÓ NHIỀU VALUE

1 VALUE ỨNG VỚI 1 BIẾN NÀO ĐÓ

BIẾN CÓ THỂ ĐỔI SANG VALUE KHÁC int yob = 2004; = 2004;

1. **DELEGATE – MIND SHIFT – SỰ DỊCH CHUYỂN TRONG TƯ DUY NHẬN THỨC/ TƯ DUY, PHÁ TƯ DUY CŨ, CỞI MỞI CHO ĐIỀU MỚI LẠ**
2. **[ 1ST – SHIFT – GENERIC – LẦN ĐẦU TIÊN TRÁI THANH LONG CÓ TRONG MÌ TÔM**

**PASS A DATA TYPE AS A PARAMETER, ĐƯA VALUE CHO HÀM KO LÀ VALUE THÔNG THƯỜNG MÀ LẠI LÀ 1 DATA TYPE]**

**\*Hàm truyền thống:**

Void AddItem(int n) void AddItem(Student s) void AddItem(Lecturer 1)

**\*Khi dùng**: AddItem(5) AddItem(new Student(){}) AddItem(new Student(){})

AddItem(10)

Tui biết trước cần đưa value gì vào hàm qua parameter

\***Hàm generic** – có đến 2 thứ cần đưa vào

Void AddItem(<T> value)

\*dùng

AddItem(<Student> new Student(){})

AddItem(<int> 5)

Sau khi xác định được kiểu dữ liệu ta mới đưa tiếp value của kiểu dữ liệu đó – 2 lần đưa, 2 lần đưa datatype được đưa vào!!! => **Generic**

1. **DELEGATE LÀ GÌ??? Là cách đặt tên nhóm cho 1 đám dữ liệu/ value phức tạp, class name đó em**

**Đám value phức tạp này là đám nào, có phải là {...} hay ko**

Đi tìm danh từ chung cho 1 đám value nào đó, ko phải value đơn giản, phức tạp như ở trên MÀ LÀ 1 THỨ PHỨC TẠP KHÁC LẠ LẮM À NHEN!!!

Vẫn là đi tìm data type cho 1 đám lạ lắm à nhen

**\* [2nd MIND SHIFT – TA NHÌN SOURCE CODE CỦA TA (DATA + METHOD, BIẾN /DỮ LIỆU VÀ HÀM) ]**

\* TA NHÌN THẤY CÓ RẤT RẤT NHIỀU HÀM (XỬ LÍ INFO/DATA) TRONG CÁC FILE SOURCE

\* VALUE – CÁC HÀM

Void SayHello() {...} no input??? no Return???

Void SayHelloToEveryOne() {...} no input??? no Return???

Void F1(){...} no input??? no Return???

Void ComputePower(int n) {...} //tính mũ 2, in ra kết quả - input???

Void computerSquareRoot(int n) {...} //tính căn bậc 2, in ra kết quả - input???

Void F2(int x) {...} //tính toán cl gì đó, in ra kq - input???

Int computePowerV2() {...} //tính mũ 2, trả về kq – return????

Int computerSquareRootV2() {...} //tính căn bậc 2, in ra kết quả làm tròn – return????

Int F3(int x) {...} //tính toán cl gì đó, in ra kq – return????

int computerSquareRootV3(int n) {...} //tính căn bậc 2, in ra kết quả input??? Return???

Int computerSquareRootV3(int ahihi) {...} //tính căn bậc 2, in ra kết quả input??? Return???

Int F4(int num) {...} //tính toán cl gì đó, in ra kq input??? Return???

**\* ĐẶT TÊN NHÓM CHO CÁC HÀM ĐỒNG DẠNG/ CÙNG HÌNH DẠNG, GIỐNG ĐỊNH DẠNG, CÙNG STYLE: INPUT/OUTPUT**

Data type đại diện cho nhiều hàm/ hàm được xem là value!!! => hàm được xem là value, ta có quyền truyền hàm vào trong hàm khác qua ngả parameter!!!

Lập trình hàm vừa xuất hiện – functional programming !!!!!!

Callback function vừa xuất hiện!!!!!!

Void SayHello() {...} no input??? no Return???

Void SayHelloToEveryOne() {...} no input??? no Return???

Void F1(){...} no input??? no Return???

-------------------------------------------------------------------------------------------------------------------------------

*Tên nhóm hàm này là x*

*Viết lại tên nhóm cho đẹp, rõ ngữ nghĩa của nhóm value ko vào ko ra*

***Delegate void X();***

*Vietsub: x là tên của 1 nhóm hàm (value) mà có đặc điểm chung là ko vào ko ra*

Void ComputePower(int n) {...} //tính mũ 2, in ra kết quả - input???

Void computerSquareRoot(int n) {...} //tính căn bậc 2, in ra kết quả - input???

Void F2(int x) {...} //tính toán cl gì đó, in ra kq - input???

-------------------------------------------------------------------------------------------------------------------------------

*Tên nhóm hàm này là y*

*Viết lại tên nhóm cho đẹp, rõ ngữ nghĩa của nhóm value có vào ko ra*

***Delegate void Y(int);***

*Vietsub: y là tên của 1 nhóm hàm (value) mà có đặc điểm chung là có vào ko ra*

Int computePowerV2() {...} //tính mũ 2, trả về kq – return????

Int computerSquareRootV2() {...} //tính căn bậc 2, in ra kết quả làm tròn – return????

Int F3(int x) {...} //tính toán cl gì đó, in ra kq – return????

-------------------------------------------------------------------------------------------------------------------------------

*Tên nhóm hàm này là Z*

*Viết lại tên nhóm cho đẹp, rõ ngữ nghĩa của nhóm value ko vào có ra*

***delegate int NamEm();***

*Vietsub: y là tên của 1 nhóm hàm (value) mà có đặc điểm chung là ko vào có ra*

int computerSquareRootV3(int n) {...} //tính căn bậc 2, in ra kết quả input??? Return???

Int computerSquareRootV3(int ahihi) {...} //tính căn bậc 2, in ra kết quả input??? Return???

Int F4(int num) {...} //tính toán cl gì đó, in ra kq input??? Return???

-------------------------------------------------------------------------------------------------------------------------------

*Tên nhóm hàm này là zzz*

***delegate int ZZZ(int)***

int, long, float, double mang rõ ý nghĩa của value nó đại diện!!!!

int: những thằng nguyên co

float: những thằng lẻ lẻ

student: mấy cu đang đi học

employee: mấy thằng nhân viên

drink: những value, món thức uống

x, y, z , zzz ở trên, rõ nghĩa chưa??? Phân biệt được chưa???

Cách đặt tên như trên chưa rõ ràng ý nghĩa!!! Nó cần phải được ghi chú rõ hơn vì ta có đến 4 loại hàm khác nhau!!

=> đặt thế này nè, mới rõ nghĩa nè:

X là nhóm hàm ko nhận vào, ko trả ra => viết gọn hơn cho đẹp: **delegate void X();**

Y là nhóm hàm có nhận vào vào, ko trả ra **Delegate void y(int);**

Z là nhóm hàm có nhận vào vào, có trả ra **Delegate int X();**

Zzz là nhóm hàm có nhận vào, có trả ra **Delegate int ZZZ(int);**

**===>Kể từ nay về phía dưới của code, bạn được quyền xài x, y, z, zzz như là 1 data type thông thường khác như student, lecturer, pet, dog, cat, ...**

**\* x đại diện cho 1 đám hàm có style Delegate void X(); tui muốn nói về 1 hàm cụ thể nào đó trong đám này:**

Void SayHello() {...} no input??? no Return??? f1

Void SayHelloToEveryOne() {...} no input??? no Return??? a

Void F1(){...} no input??? no Return??? aa

------------------------------------------------------------

**Delegate void X(); //tạo ra datatpe tên là x do loại value quá phức tạp, có cả thuật toán trong hàm/value**

**X f1 = bằng được cái hàm SayHello( ) nữa là ngon r**

**X a = bằng được cái hàm SayHellotToEveryOne( ) nữa là ngon r**

**= ???**

**Vietsub: f1 là 1 cái hàm có style kiểu x, tức là 1 cái hàm ko vào ko ra dạng void x() và cụ thể hàm này chính là hàm SayHello( ) ở trên hoặc 1 hàm khác cụ thể ở trên nhưng phải cùng style x**

**F1 = ???????? gán được hàm ở trên là xong!!!!!!!!**

**F1 được gọi là con trỏ hàm vì nó = hoặc trỏ đến 1 hàm cụ thể - function pointer!!!!!**

**Data type biến = value**

5 a

10 b

15 c

------------------------------------------------------------

int

Int a = 5

**============================================================================**

1. **Cách thể hiện, đinh nghĩa ra data type cho nhóm dữ liệu phức tạp {...} {...} void F( ) { }**

**1. data type dữ liệu đơn giản**

**\* data type:** danh từ chung, là tên gọi chung cho 1 nhóm dữ liệu, thông tin đơn giản mà đồng dạng, cùng hình dạng/ format

\* int: 5 10 15 20 100 1000 -5 ... (nguyên con, ko lẻ miếng)

\* float: 3.14 9.8 10% ¾ (có lẻ miếng)

**2.1 data type dữ liệu phức tạp – ko phải là hàm mà là đối tượng nào đó có nhiều info**

**\* data type:** danh từ chung, là tên gọi chung cho 1 nhóm dữ liệu, thông tin đơn giản mà đồng dạng, cùng hình dạng/ format

\* person: {Tên..., ngày sinh..., nơi sinh .....} {Nguyễn Thanh Tùng, 1994, Thái Bình}

\* single: {Tên..., ngày sinh..., nơi sinh ....., nickname...} {Nguyễn Thanh Tùng, 1994, Thái Bình, M-TP}

\* player, footballer: {Tên..., ngày sinh..., nơi sinh ....., nickname..., number..., goals..., clb..., awards...}

{messi, 1988, ...}

Person, single, player: nhóm đối tượng có nhiều info => gọi tên khác là: ENTITY (DATABASE)

CLASS (LẬP TRÌNH)

DATATYPE: dạng dữ liệu

Class: sự phân nhóm, phân hạng

>>>>> code để thể hiện ý tưởng lưu trữ và xử lí các info ngoài đời bằng máy tính (ABSTRACTION – TRỪU TƯỢNG HÓA – BIỂU DIỄN INFO NGOÀI ĐỜI THỰC QUA MÁY TÍNH, PERSON NGOÀI ĐỜI => CLASS TRONG OOP)

public class Person { //java

Private Sring id;

Private String name;

}

**//data type là tên chung/danh từ chung cho rất nhiều object phức tạp {...} ngoài kia, gồm bạn và tôi**

public class Person public class Person

{ //c# {

Private string \_id; public string Id {get; set; }

Private string \_name; public string Name{get; set; }

Private int\_yob; public int Yob {get; set; }

} {

//tạo ra dữ liệu => BIẾN ĐI KÈM LÀ DANH TỪ RIÊNG, TÊN TẮT, NICKNAME CHO 1 OBJECT THẬT NGOÀI KIA

//Person an = {xxx, An, 2003, ...}

Person an = new Person (...) { ... } //C#: object initializer – tạo object, tạo value = dùng hàm Set(...)

* 1. **data type dữ liệu phức tạp – là quá trời hàm, hàm có: ???f\_name (???) {...}**

**- tên hàm**

**- có trả về hay ko**

**- 0 có đầu vào hay ko**

**- code bên trong hàm để xử lí info nói chung {...}**

**\* data type:** danh từ chung, là tên gọi chung cho 1 nhóm dữ liệu, thông tin đơn phức tạp mà đồng dạng, cùng hình dạng/ format

Void SayHello( ) {...}

Void NhanEm( ) {...}

Void TellHer( ) {...}

-------------------------------

Delegate void x( );

Vietsub: x là tên gọi chung, danh từ chung cho 1 nhóm hàm đồng dạng bề ngoài: ko nhận đầu vào cũng ko trả ra gì cả, chỉ xử lí gì đó bên trong {...}

**Ngầm trong việc thể hiện code, x là 1 class, 1 nhóm phức tạp**

**Data type cho nhóm phức tạp, tên nhóm phức tạp thì còn gọi là class, class person, class pet, class rectangle, ...**

1

2

3

4

----

Int x = 10

Student s = new Student(...) {...};

Void SayHello( ) {...}

Void NhanEm( ) {...}

Void TellHer( ) {...}

-------------------------------

Delegate void x( ); >>>>> X LÀ 1 CLASS NHƯ STYLE PERSON, DOG, CAT, CAR, PET, LECTURER, DOGTER, USER, ...

Public class x [void x()] delegate public class Person

{ {

\_fName/ Tên hàm cần lưu Id, Name, Yob, ...

//vì x là tên chung cho nhiều hàm //Person đại diện cho nhiều công dân

//nên x cần có chỗ để lưu tên hàm //person phải có chỗ để lưu info công dân

//constructor – phễu đổ info bên ngoài vào bên trong cái class/khuôn

Public x (tên hàm muốn lưu trữ) public Student(id, name, yob )

{ {

\_fName = tên hàm muốn lưu trữ Id = id

} }

//get set

} }

Public void Invoke() //hàm có sẵn khi tạo class Delegate X, hoặc tạo theo style nhanh, giống kiểu ToString() có sẵn kế thừa từ ông tổ object – INHERITANCE OOP

{

fName() //tên hàm được lưu trữ gửi vào sẽ được run = cách thêm dấu () để run  
}

=============================================================================

>>>>>>>**ĐẶC BIỆT CHO CÁI CLASS MÀ ĐẠI DIỆN CHO 1 NHÓM HÀM NÀO ĐÓ, TA KO CẦN MẤT SỨC VIẾT THEO STYLE CLASS TRUYỀN THỐNG<<<<<<<**

**delegate void X( )** LÀ ĐỦ RỒI**, KO CẦN** CODE THEO STYLE DÀI DÒNG CLASS TRUYỀN THỐNG

**LÍ DO: -** TA CHỈ CẦN LƯU CÁI TÊN HÀM, DO TA ĐẠI DIỆN CHO 1 ĐÁM HÀM CÙNG STYLE

- ĐẠI DIỆN HÀM NÀO THÌ CHỈ CẦN NÓI TÊN HÀM LÀ ĐỦ, QUA TÊN HÀM THÌ MÒ VÀO CHI TIẾT XỬ LÍ CỦA HÀM Ở {...}, code nếu có cũng cũng ngắn, ko cần thiết public class... {...}

\*\*\*\*\*code ngắn gọn cho cái class đại diện cho nhóm hàm !!!!!!!

Delegate Style nhóm hàm ở đây kèm cái tên chung là đủ rồi

Delegate void X( ) // X: tên chung cho nhóm hàm ko vào ko ra

Delegate void Y(int) // Y: tên chung cho nhóm hàm nhận vào 1 con int ko ra

Deletgate void YY (int, int) // YY: tên chung cho nhóm hàm nhận vào 2 con int, ko ra

Delegate int Z( ) // Z: tên chung cho nhóm hàm ko nhận vào nhưng trả ra

Delegate int ZZ(int) //ZZ: tên chung cho nhóm hàm có 1 vào, trả ra

Nhóm hàm nhiều – class nhiều

Cũng như nhiều class ngoài kia: Person, car, dog, pet, rectangle, circle, triangle, file, ...

* 1. **Dùng data type thì sao????????**

**\*** Khai báo biến > thuộc data type > gán value = new (...) {...}

Int x = 10;

Student x = new Student(...) {...}; //đưa info sv đây cho tao, tao gọi hắn

là s, an, bình, ...

X f = new X (tên hàm cụ thể nào đó có ở đâu đó); //đưa tên hàm vào cho tao, và gọi là F

f = new X(SayHello);

KỂ TỪ NAY VỀ SAU, CÁI HÀM CỤ THỂ NÀO ĐÓ ví dụ hàm SayHello được gọi tên là F

**F là nickname của hàm SayHello ở đoạn ghi chú trên!!!!!**

1. **Delegate class thêm có một trò đặc biệt – nó có khả năng lưu nhiều tên hàm cùng 1 lúc new**

**MULTICAST DELEGATES**

Public class x [void x()] delegate

{

\_fName/ Tên hàm cần lưu

//vì x là tên chung cho nhiều hàm

//nên x cần có chỗ để lưu tên hàm

List< > danh sách các tên hàm cùng style

[ ] mảng các tên hàm cùng style

//tùy lúc mình gán tên hàm vào biến f delegate thì sẽ xài đơn hàm, hay đa hàm cùng 1 lúc!!!

//constructor – phễu đổ info bên ngoài vào bên trong cái class/khuôn

Public x (tên hàm muốn lưu trữ)

{

\_fName = tên hàm muốn lưu trữ

}

//get set

}

Public void Invoke() //hàm có sẵn khi tạo class Delegate X, hoặc tạo theo style nhanh, giống kiểu ToString() có sẵn kế thừa từ ông tổ object – INHERITANCE OOP

{

fName() //tên hàm được lưu trữ gửi vào sẽ được run = cách thêm dấu () để run

List< > các hàm được gọi = cách thêm ( ) trên từng h àm để run( )

}

Hàm dấu += để ghép thêm tên các h àm vào danh sách đại diện!!!

//kĩ thuật override các operator, định nghĩa lại các dấu + - \* / thay vì làm trong toán học

//giờ toán trong object!!!

}

============================================================================

**REVIEW NHANH LẠI VỀ DELEGATE - ỦY NHIỆM, ỦY QUYỀN, TRAO QUYỀN, ĐẠI DIỆN, SỨ GIẢ, 1 PHIÊN BẢN CLONE, 1 NICKNAME KHÁC**

**0. CÁCH GỌI KHÁC, MỘT CON TRỎ...**

**1. DELEGATE LÀ 1 CLASS NHƯ CÁC CLASS TRUYỀN THỐNG DOG, CAT, PERSON, PRODUCT...**

DELEGATE LÀ 1 CLASS NAME, LÀ TÊN GỌI CHNG, DANH TỪ CHUNG ĐẠI DIỆN CHO NHIỀU HÀM CÙNG 1 STYLE NÀO ĐÓ NÓ DÙNG ĐỂ NÓI VỀ 1 TẬP HỢP CÁC HÀM CÙNG STYLE, THÔNG QUA NÓ CÓ THỂ BIỂU DIỄN /THAM CHIẾU/NÓI VỀ CÁC HÀM

LÀ TÊN GỌI CHUNG TÊN TẬP HỢP NHÓM HÀM

X là tên gọi cho 1 nhóm hàm {F1() F2() F3() F4() ...}

**2. khai báo và sử dụng delegate:**

BỘ 3: ĐÀO PHỞ PIANO

DATA TYPE TÊN BIẾN = VALUE

Tên delegate nickname của 1 hàm 1 hàm cụ thể nào đó được gọi tên mới

Delegate void X();

X f = new Delegate(tên hàm mình đặt nick là f bên trái dấu bằng)

X f = new Delegate(TellHer) void TellHer( ) {...}

f------------------------------------------------------------------🡪 TellHer

X f = TellHer; //viết ngắn gọn, new ngầm

F = NhanEm; //trỏ vào hàm mới, tại 1 thời điểm biến lưu 1 value

f += SaySweetWords; //f đại diện cho nhiều thân chủ, nhiều hàm: NhanEm + SaySweetWord

//MULTICAST DELEGATES

gọi hàm :

f.invoke();

f(); //f = TellHer => suy ra f() cũng là TellHer() chạy hàm thôi

DELEGATE: là cách đặt tên gọi cho 1 hàm khác, 1 hàm có sẵn có thêm 1 nick mới, 1 tên mới

f() run thì cũng như TellHer run

luật sư nói thì cũng như thân chủ nói

=============================================================================

BÀI TẬP NÂNG CAO:

\* ĐỘ LẠI CÁI CLASS KUKIA ĐỂ TỰ NHIÊN HƠN NỮA

- kukia có 1 hành động riêng là: TalkMessage( message của ai thì đưa vào đây)

- kukia vẫn có hành động MeetSweetHeart() => run() 2 message kia

=============================================================================

1. **CÁC BƯỚC, CÔNG THỨC ĐỂ CHƠI VỚI DELEGATE**

**\*ĐỊNH NGHĨA NHANH LẠI VỀ DELEGATE**

- **DELEGATE** là 1 loại class đặc biệt – bản chất nó cũng là 1 data type như các loại data type: int, long, double, student, lecturer, …

-**DELEGATE** là loại class đặc biệt nó dùng để lưu info, info đặc biệt: tên các hàm, tahy vì lưu info thông thường kiểu ngày tháng năm sinh, trạng thái đơn hàng, tên tuổi…

-**DELEGATE** là tên gọi chung cho 1 nhóm hàm cùng style, cùng định dạng, cùng format (TYPE)

- chơi với delegate là sẽ giúp cho 1 cái hàm bình thường có thêm 1 tên gọi nữa, là cách đặt nickname cho 1 hàm nào đó, 1 tên gọi thứ 2 cho h àm nào đó (giống như tên ở nhà của mình là Cún), hàm có thêm 1 tên gọi khác, vì là tên gọi khác nên còn gọi là con trỏ đếm hàm, con trỏ h àm

- **DELEGATE** giống như câu chuyện luật sư = thân chủ, thân chủ là hàm gốc ủy quyền, trao quyền, ủy nhiệm, sứ giả cho luậ sư hành động thay

- 1 hàm sẽ có 2 cách gọi: gọi hàm gốc để gọi: gọi hàm gốc để chạy, gọi nickname để chạy, nói thay cho thân chủ

**Luật sư() run cũng như thân chủ run()**

**1. Bước 1:** Khai báo DELEGATE – tạo datatype chuyên đi sưu tầm lưu trữ tên hàm

Khai báo ở ngang cấp, ngang cơ, ngang lever với các class khác, interface khác

Có thể dùng thêm: internal, public, ko càn public, tùy

\* CÚ PHÁP:

Delegate kiểu\_trả\_về\_của\_hàm tên\_delegate(**tham\_số\_hàm**)

Ví dụ: delegate void F();

Public delegate void F(int x);

Internal delegate int F();

Delegate int f(int x);

**//thay int bằng kiểu data type bạn cần, hàm có thể có nhiều hơn 1 tham số!**

🡺 F kể từ nay về sau là 1 loại datatype ngang cơ int, long, double, float, Student, Lecturer, …

🡺 NÊN DÙNG CÁI DELEGATE CÓ SẴN ĐƯỢC CUNG CẤP BỞI MICROSOFT, bởi vì .net sdk thay vì phải tự tạo riêng những delegate như khai báo ở trên

Tức là bạn ko cần tự làm, ko cần khai báo delegate vẫn xài delegate, xài đổ chuẩn của microsoft => tương thích qua lại với code trên mạng, với các code của developer khác trên thế gới. Đây là cách thiên hạ đang dùng!!!

\* CÓ 4 DATATYPE NỔI TIẾNG – 4 DELEGATE NỔI TIẾNG TRONG .NET – chuẩn hóa cách đặt tên các delegate:

Action Action< > Func< > Predicate< >

Void F() void F(???) ???F(???) bool F(?)

**>>>tạo datatype**

**2. Bước 2: Xài data type – khai báo biến dùng để trỏ hàm,**

**lưu tên hàm, lưu hàm cụ thể nào luôn!**

Luật sư = thân chủ; //luật sư deal dùm tui

Data type biến = value;

\* công thức:

datatype\_delegate\_type\_tên\_nhóm\_hàm biến\_trỏ\_hàm = Hàm\_nào\_đó\_ở\_đâu\_đó;

nickname = hàm\_nào\_đó;

ví du:

F f = Hàm\_nào\_đó\_ở\_đâu\_đó; //hàm có sẵn

F f = Hàm\_nào\_đó\_ở\_ ngay\_đây; //hàm ngay đây

Anonymous function, lambda expression

Biến, nickname thân chủ

Luật sư

=====>NÊN DÙNG: KHAI BÁO NHƯ SAU:

Action f = hàm nào đó …

Action<int> f = hàm nào đó …

Action<string, string> f = hàm nào đó …

Action<int> f = hàm nào đó …

Action<double, double> f = hàm nào đó …

Predicate<double> f = hàm nào đó …

Prediace<Student> f = hàm nào đó …

Student s1 = value nào đó, sinh viên nào đó;

Int yob = value nào đó, con số nguyên cụ thể (35.000.000)

**3. Bước 3: GÁN GIÁ TRỊ CHO BIẾN ĐÃ KHAI BẢO Ở BƯỚC 2 QUA TOÁN TỬ DẤU = ( +=, -=)**

**TÊN BIẾN = VALUE;**

Biến là tên gọi cho 1 value cụ thể nằm trong tập DATA TYPE…

Yob = 2003; yob là tên gọi cho value cụ thể 2003 thuộc tập data tên là int, long, …

Delegate\_type biến = value là:

Delegate\_type biến = tên hàm nào đó cụ thể ngoài kia (hàm gốc có sẵn - explicit)

= hàm nào đó ngay đây, ko thèm để tên hàm, bán tên rồi, chỉ còn code thôi (tên hàm đi kèm code tên\_hàm() {…code…}

Chính là ta đang dùng anonymous function hoặc LAMBDA EX

**(bán linh hồn, bán cái tên, chỉ còn cái dây nịt)**

Nickname = tên hàm gốc;

Luật sư = thân chủ hàm gốc; ủy quyền rồi, thiết lập hợp đồng

Trao quyền, ủy nhiện, ủy thác,

Tuy 2 mà là 1, ai gáy, ai run() đều như nhau

+= trỏ tới nhiều hàm, 1 nick đại diện cho nhiều hàm

1 luật sư hợp đồng với nhiều thân chủ

**MULTICAST DELEGATE**

3.1 GÁN GIÁ TRỊ CHO BIẾN NICKNAME, BIẾN LUẬT SƯ = 1 HÀM CỤ THỂ NÀO ĐÓ CÓ SẴN

F f1 = TellHer; //..ko có dùng () vì gây hiểu nhầm: chạy hàm đi

//phải làm hợp đồng trước rồi mới cãi!!!

=========> hàm TellHer đã có sẵn code ở chỗ khác, nay đặt thêm cái nickname, 2nd name

Int yob = 2004;

**3.2 GÁN GIÁ TRỊ CHO BIẾN NICKNAME, BIẾN LUẬT SƯ = 1 NGAY ĐÂY, KO CÓ SẴN, TẠO NGAY LÚC GÁN – HÀM ON THE GO…**

KO CÓ CHỖ NÀO SẴN, KO CẦN LUÔN

HÀM ẨN DANH – ANONYMOUS FUNCTION

BIỂU THỨC LAMBDA - HÀM ẨN DANH KIỂU CÒN CÁI DÂY NỊT, TỐI GIẢN PHÂN SỐ

RÚT GỌN SỰ DƯ THỪA TRONG VIỆC GÕ LỆNH TẠO HÀM, BỎ ĐI TÊN HÀM,

BỎ BỚT KÍ HIỆU Ở THAM SỐ,

BỎ BỚT KÍ HIỆU Ở THÂN HÀM

F f1 = delegate( ) {code của hàm …};//vip, ẩn danh

F f1 = (…) => { code của hàm …}; //vip, ẩn danh style lambda expression

//chỉ còn cái dây nịt, phân số tối giản

**3.3 QUY TẮC CHỈ CÒN CÁI DÂY NỊT, QUY TẮC TỐI GIẢN LÀM VIỆC VIẾT HÀM CHƠI VỚI DELEGATE**

RÚT BỎ HẾT RÂU RIA CỦA CÁI HÀM, CHỈ GIỮ LẠI PHẦN LÕI CỦA 1 CÁI HÀM: TÊN THAM SỐ ĐẦU VÀO VÀ THÂN HÀM (CÓ KHI BỎ LUÔN CẢ {RETURN})

delegate\_type **BIẾN** = tên\_hàm\_nào\_đó\_ngoài\_kia; void TellHer() {...code\_hàm...}

delegate\_type **BIẾN** = delegate ( ) {...code hàm...}; //VIP; //anonymous

delegate\_type **BIẾN** = ( ) => {...code hàm...}; //VIP; //lambda

**3.3.1 RÚT GỌN Ở THAM SỐ ĐẦU VÀO CỦA HÀM**

( ) =>

\* Nếu đầu vào của hàm ko có gì cả, thì gõ: ( ) =>

\* Nếu đầu vào của hàm có 1 đầu vào, thì gõ: (int x) =>

(x) =>

x => //cách này được ưa thích trên toàn cầu

\* Nếu đầu vào của hàm có 2 đầu vào trở lên, thì gõ: (int x, int y) =>

(x, y) => //được ưa thích hơn

Từ 2 đầu vào trở lên thì k đc bỏ ngoặc tròn như 1 đầu vào

**3.3.2 RÚT GỌN Ở THÂN HÀM**

=> {thân hàm – code hàm, body of function, implementation}

\* Nếu hàm chỉ có duy nhất 1 dòng lệnh, ta dẹp luôn { } dẹp luôn return nếu có, viết thân hàm lúc này y chang expression body

=> lệnh gì đó ở đây;

\* Nếu thân hàm có nhiều dòng lệnh (từ 2 dòng trở lên) thì ko được dẹp gì cả, viết như truyền thống

=> {

...code của hàm, lệnh return nếnu có, do, while từa lưa gì đó...

};

**4. BƯỚC 4: GỌI HÀM, GỌI HÀM QUA NICKNAME THAY VÌ GỌI TRỰC TIẾP HÀM NHƯ TRUYỀN THỐNG**

Int yob = 2004; //bước 3 – gán giá trị

Student s = new Student(...) {...}; //bước 3 – gán giá trị

F f = Hàm nào đó (TellHer); //bước 3 – gán giá trị

= anonymous function; //bước 3 – gán giá trị

= biểu thức Lambda;

GỌI HÀM ĐỂ CHẠY

F( ) run hàm qua nick name, luật sư bắt đầu cãi ...

f.invoke(...); //gọi hàm gốc, hay hàm ẩn danh, chạy thôi!!

=========================================================

BIẾN BÌNH THƯỜNG = GIÁ TRỊ; RỒI XONG!!

Int yob = 2004; xong

Student s = new Student(...) {...} rồi xong

BIẾN DELEGATE = GIÁ TRỊ TÊN HÀM; HÀM CỤ THỂ; HÀM ẨN DANH, HÀM SYLE LAMBDA; CHƯA XONG

GỌI HÀM ĐỂ RUN( ) GỌI HÀM GIÁN TIẾP QUA NICKNAME, ỦY QUYỀN, ỦY NHIỆM, ỦY THÁC...

========================================================

1. **TỔNG KẾT NHANH GỌN VỀ DELEGATE – CHUYỆN LUẬT SƯ = THÂN CHỦ = ĐẠI DIỆN NÓI HỘ, PHÁT NGÔN HỘ**

**NICK NAME = HÀM NÀO ĐÓ;**

Hàm nào đó()

{

Code…

}

Con trỏ, biến trỏ đến hàm nào đó – hàm có tên gọi khác nữa

Code và hàm ở trên có thể thay = anonymous func và lambda expression

**I. Khai báo DELEGATE dùng đại diện cho 1 nhóm hàm**

1. tự khai báo delegate riêng

=> delegate void F(…);

=> delegate ??? F(…);

2. dùng DELEGATE có sẵn cung cấp bởi .NET SDK

=> action : hàm void F(ko đầu vào)

=> Action<???> : hàm void F(từ 1…16 đầu vào khác nhau – dùng generic)

=> Func<???, ?> : hàm có trả về giá trị ? ( từ 0 – 16 đầu vào khác nhau)

Vd: Func<???, int>

=> Predicate<?> : tương đương hàm Func<?, bool>

Hàm có trả về giá trị bool đúng sai, nhưng có nhận đúng duy nhất 1 đầu vào

Generic luôn

**II. KHAI BÁO VÀ GÁN GIÁ TRỊ CHO BIẾN DELEGATE**

**ĐẶT TÊN CHO 1 HÀM NÀO ĐÓ, TẠO DỰNG CÂU CHUYỆN LUẬT SƯ = THÂN CH Ủ**

=> Action f = Một\_hàm\_nào\_đó; //k trả kq

=> Action<int> f = Một\_hàm\_nào\_đó\_mà\_có\_một\_đầu\_vào\_int; //k trả kq

=> Func<int> f = Một\_hàm\_nào\_đó\_mà\_có\_trả\_về\_int\_ko\_đầu\_vào //có trả kq

=> Func<int, string> f = Một\_hàm\_nào\_đó\_mà\_có\_trả\_về\_string\_đầu\_vào\_int //có trả kq

=> Predicate<string> = Một\_hàm\_nào\_đó\_mà\_có\_trả\_về\_bool\_đầu\_vào\_string //có trả kq

=> Predicate<Student> = Một\_hàm\_nào\_đó\_mà\_có\_trả\_về\_bool\_đầu\_vào\_Student//có trả kq

**III. RUN HÀM, CHẠY HÀM, GỌI HÀM CHẠY 1 CÁCH GIÁN TIẾP QUA NICKNAME, QUA LUẬT SƯ, NHẬN ỦY QUYỀN VÀ ACTION; KO GỌI HÀM TRỰC TIẾP (MẶC DÙ OK); RIÊNG HÀM ẨN DANH MẤT TÊN RỒI, GỌI QUA NICKNAME**

=> Biến(tham số đưa vào nếu cần)

F(???);

=> Biến.Invoke(tham số đưa vào nếu cần);

f.Invoke(???);

\* kí hợp đồng – gán nickname – luật sư = thân chủ

\*> xong rồi mới call, run, invoke

=============================================================================

1. **DELEGATE IN USE, IN ACTION – CÁCH DELEGATE DƯỢC ỨNG DỤNG TRONG THỰC TẾ**

**1. TRUYỀN THÁI Y – TRUYỀN DELEGATE VÀO TRONG 1 HÀM KHÁC (CÂU CHUYỆN 2 HÀM)**

void HamTraiBao(int n);

void HamTraibao(Action f) f = hàm\_nào\_đó\_có\_sẵn\_hoặc\_on\_the\_go\_code\_ngay\_lệnh\_gán

= delegate( ) { } hoặc = ( ) => { }

Anonymou lambda

{

Hàm trai bao cứ for do while thoải mái

Tao làm gì với f here??

=> gọi nó vì nó là hàm bên ngoài đưa vào – hàm thì được quyền run( )

F( ); //f.Invoke( ); //call back, tao gọi mày nè!!!

//lập trình hàm – functional programming

//hàm mình chơi láng với các hàm bên ngoài

//gọi hàm bên ngoài, bên ngoài làm gì thì ai biết???

//f có logic xử lí riêng, được đưa vào hàm mình gọi => inject -

Chích

//dependency injection

//tao chơi với mọi thứ bên ngoài, loose coupling, flexible

}

\*\*\*\*\*ta thiết kế 1 cái hàm hub, hàm đầu mối, hàm style DoOnDemand(yêu cầu bên ngoài đưa vào)

CUNG CẤP DỊCH VỤ LÀM THEO YÊU CẦU CỦA AI ĐÓ ĐƯA ĐẾN – HÀM (CÓ CODE) ĐƯA VÀO HÀM CHÍNH CỦA TA NHẬN THẦU THEO YÊU CẦU, VẠN SỰ TÙY DUYÊN, TA PHÓ MẶC CHO CODE BÊN NGOÀI ĐƯA VÀO!!!

[CODE BÊN NGOÀI XÀI DATA BÊN TRONG CỦA TA]

**2. TRUYỀN HÀM CÓ ĐẦU VÀO 1 HÀM KHÁC**

void DoOnDemand(Action<int> f) ========== void F(?) void F(int x)

{

//gọi f thế nào

f( ); //đưa tham số vào cho thằng f

f(5);

}

Math.Sqrt(25);