Bài 1: Trả lời câu hỏi

1. Sự khác nhau giữa các bổ từ truy xuất (Access Modifiers):

|  |  |  |
| --- | --- | --- |
| Bổ từ truy xuất | Áp dụng cho | Mô tả |
| Public | Bất kỳ loại (type) hoặc thành phần (member) nào đó. | Không có bất cứ giới hạn nào khi truy cập vào các thành viên công cộng. |
| Protected | Bất kỳ thành phần của một loại và bất kỳ loại được lồng. | Bị giới hạn trong phạm vi định nghĩa của lớp và bất kỳ các lớp con cháu thừa kế từ lớp này (lớp con cháu). |
| Private | Bất kỳ thành phần của một loại và bất kỳ loại được lồng. | Truy cập bị giới hạn trong phạm vi định nghĩa của lớp. Đây là loại phạm vi truy cập mặc định nếu không được chính thức chỉ định private. |
| Internal | Bất kỳ loại (type) hoặc thành phần (member) nào đó. | Bị giới hạn trong phạm vi Assembly chứa lớp của dự án hiện tại (láng giềng gần). |
| Protected internal | Bất kỳ thành phần của một loại và bất kỳ loại được lồng. | Hội tụ hai tầm vực internal và protected. Thuộc tính được truy xuất bất kỳ đâu trong cùng một Assembly chứa lớp tương ứng hoặc các lớp con cháu. |
| Private protected (C# 7.2) | Bất kỳ thành phần của một loại và bất kỳ loại được lồng. | Trái ngược với “protected internal”. Có nghĩa là chỉ là protected hoặc chỉ là internal. Quyền truy cập chỉ được phép đối với các loại dẫn xuất nằm trong cùng một Assembly, nhưng không được phép từ các Assembly khác. |

1. Truy cập mức luận lý (logical): public, private và protected
2. Truy cập mức vật lý (physical) - assembly: internal
3. Phân biệt giữa class, static class và partial class:

**Class bình thường:** được khai báo với từ khóa class. Cần phải tạo ra một đối tượng cụ thể của lớp.

**Partial class:** được khai báo với từ khóa partial class. Với từ khóa này thì sự định nghĩa 1 lớp có thể chia ra nhiều file. Các phần file này có cùng mức độ truy cập giống nhau. Trình biên dịch C# sẽ kết hợp tất cả các phần này thành một sự định nghĩa 1 lớp.

**Static class:** được khai báo với từ khóa static class. Có thể sử dụng lớp này bất cứ khi nào trong chương trình mà không cần tạo ra các intance (thực thể, thể hiện) của lớp. Thường được dùng để khai báo các tiện ích hoặc hằng số.

1. Ý nghĩa thuộc tính: Khi khai báo Access Modifier là private thì việc truy cập thuộc tính sẽ phụ thuộc vào việc cài đặt các thuộc tính giao tiếp (Accessors) là:

**readonly:** khi chỉ khai báo get accessor. (làm hạn chế tính hữu dụng của đối tượng, hạn chế sử dụng trong thiết kế phần mềm)

**writeonly:** khi chỉ khai báo set accessor.

**Read-write:** khi khai báo cả 2 loại trên (set; get).

1. Các constructor và destructor trong một lớp, sự khác nhau giữa chúng:

|  |  |  |
| --- | --- | --- |
| Constructors | Phương thức khởi tạo mặc định | Do trình biên dịch tự động phát sinh khi LTV không khai báo phương thức này. |
| Phương thức khởi tạo không tham số | Nhằm gán các giá trị khởi đầu cho các thuộc tính. |
| Phương thức khởi tạo có một phần tham số | Nếu được khởi tạo cho cùng một field thì nên dùng cú pháp đặc biệt là constructor initializer.  Phương thức khởi tạo nhiều tham số hơn sẽ thực thi trước bất kỳ code nào trong thân của phương thức khởi tạo có ít tham số hơn. |
| Phương thức khởi tạo đầy đủ tham số |
| Destructors/ Finalizers | Chỉ có một destuctor (có dấu “~” trước nó) | Được gọi khi CLR phát hiện rằng một đối tượng lâu quá không cần nó nữa.  Hay nói cách khác là khi đối tượng hết phạm vi sử dụng thông qua bộ thu dọn rác tự động (GC) thì destructor sẽ được gọi để tránh xảy ra những tình trạng ko muốn hoặc rò rỉ bộ nhớ. |

1. Mục đích, ý nghĩa các thành viên của lớp:

|  |  |
| --- | --- |
| Các thành viên lớp | Mục đích, Ý nghĩa |
| Các field | Là các thuộc tính vật lý, chúng được khai báo là private nhằm ngăn chặn không cho người dùng tác động trực tiếp lên các dữ liệu nhạy cảm của lớp.  Các static field tạo ra những biến dữ liệu dùng chung cho tất cả đối tượng của riêng một lớp, thường dùng để đếm số lượng đối tượng được tạo ra. |
| Các property | Là các thuộc tính giao tiếp dùng để hỗ trợ người dụng truy cập và cập nhật dữ liệu. |
| Các phương thức | Bao gồm các phương thức khởi tạo, phương thức hủy, các phương thức hỗ trợ (Helper) người dùng truy xuất (ge()) và lưu trữ - cập nhật dữ liệu (get()), các phương thức tĩnh để tạo tiện ích người dùng và phương thức hằng. |

1. Các mối quan hệ không có tính kế thừa giữa 2 class, minh họa:

Dependency (quan hệ - uses):

public class ClassB

{

//members

}

public class ClassA

{

//members

public void doSomeThing(ClassB c)

{

}

}

Aggregation (quan hệ - has):

public class Address

{

//members

//field types ClassB

private Address studentAddress;

//properties

public ClassB StudentAddress

{

get

{

return studentAddress;

}

set

{

studentAddress = value;

}

}

}

Composition (quan hệ - owns):

Public class CompanyLocation

{

String blockName, address, direction;

}

Public class Company

{

//members

String CompanyName;

Public CompanyLocation location;

Public company()

{

CompanyName = “VNToiYeu”;

Location = new CompanyLocation();

}

}