1. ***Phỏng vấn .net fresher cần chuẩn bị những gì?***

Ôn kỹ C#, OOP, cấu trúc dữ liệu giải thuật

Ngoài ra SQL thì chuẩn 123 rồi các câu lệnh query

+1NF: thuộc tính đơn trị, chỉ nhận giá trị nguyên tố

+2NF: không có phụ thuộc hàm bộ phận vào khóa chính

+3NF: không có phụ thuộc hàm bắt cầu vào khóa chính

+BCNF: không có thuộc tính khóa phụ thuộc hàm vào thuộc tính không khóa

Thêm ít design pattern: hay dùng factory method, singleton, abstract factory, prototype( dùng để tạo bản sao), builder, decorator.

SOLID

Mô hình MVC

API

RESTful

Cty: fsoft or nasthech

1. ***Lập trình hướng đối tượng là gì? Các tính chất của lập trình hướng đối tượng, Theo em tính chất nào của lập trình hướng đối tượng là quan trọng nhất? và em áp dụng những tính chất nào vào lập trình hướng đối tượng vào code hay trong project của mình không?khác nhau của đa hình vs kế thừa?Overide vs overloading?bài test thì mấy cái bài số nguyên tố, ước số chung***

[Lập trình](https://wiki.tino.org/request-va-response-trong-lap-trinh-web/) hướng đối tượng (còn được gọi là Object Oriented Programming, viết tắt là “OOP”) là một phương thức lập trình mà lập trình viên lấy đối tượng làm nền tảng để viết chương trình. Theo một định nghĩa khác, lập trình hướng đối tượng là một phương pháp lập trình dựa trên khái niệm về lớp và đối tượng và tập trung vào các đối tượng thao tác, giúp code dễ quản lý, tái sử dụng được và dễ bảo trì.

***ĐỐI TƯỢNG****LÀ NHỮNG SỰ VẬT, SỰ VIỆC MÀ NÓ CÓ NHỮNG TÍNH CHẤT, ĐẶC TÍNH, HÀNH ĐỘNG GIỐNG NHAU VÀ TA GOM GÓP LẠI THÀNH ĐỐI TƯỢNG GIỐNG TRONG THỰC TẾ CUỘC SỐNG. KHI LẬP TRÌNH OOP, CHÚNG TA SẼ ĐỊNH NGHĨA CÁC LỚP (CLASS) ĐỂ GOM (MÔ HÌNH) CÁC ĐỐI TƯỢNG THỰC TẾ.*

**Đối tượng**

Một đối tượng bao gồm 2 thông tin: **thuộc tính** và **phương thức**.

* **Thuộc tính**chính là những thông tin, đặc điểm của đối tượng. Ví dụ: con người có các đặc tính như mắt, mũi, tay, chân…
* **Phương thức**là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: một người sẽ có thể thực hiện hành động nói, đi, ăn, uống, . . .

### ****Lớp****

Một lớp là một kiểu dữ liệu bao gồm các thuộc tính và các phương thức được định nghĩa từ trước. Đây là sự trừu tượng hóa của đối tượng. Khác với kiểu dữ liệu thông thường, một lớp là một đơn vị (trừu tượng) bao gồm sự kết hợp giữa các phương thức và các thuộc tính. Hiểu nôm na hơn là các đối tượng có các đặc tính tương tự nhau được gom lại thành một lớp đối tượng.

**Sự khác nhau giữa đối tượng và lớp**

Lớp bạn có thể hiểu nó như là khuôn mẫu, đối tượng là một thực thể thể hiện dựa trên khuôn mẫu đó. Ví dụ: Ta nói về loài chó, bạn có thể hiểu nó là class (lớp) chó có:

* Các thông tin, đặc điểm: 4 chân, 2 mắt, có đuôi, có chiều cao, có cân nặng, màu lông…
* Các hành động như: sủa, đi, ăn, ngủ…

Đối tượng thì chính là con chó Phú Quốc ta đang nuôi trong nhà cũng mang đặc tính của lớp chó.

## **Các nguyên lý cơ bản của OOP**

### ****Tính đóng gói (Encapsulation)****

**Note:**

* Các dữ liệu và phương thức có liên quan với nhau được đóng gói thành các lớp để tiện cho việc quản lý và sử dụng. Tức là mỗi lớp được xây dựng để thực hiện một nhóm chức năng đặc trưng của riêng lớp đó.
* Ngoài ra, đóng gói còn để che giấu một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không thể nhìn thấy.

***CÁC THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA ĐỐI TƯỢNG CẦN CHO VIỆC GIẢI QUYẾT BÀI TOÁN ĐÃ ĐƯỢC CHỌN RA SẼ ĐƯỢC ĐÓNG GÓI VÀO MỘT KIỂU DỮ LIỆU GỌI LÀ CLASS. NHẰM CHE DẤU THÔNG TIN KHỎI CÁC ĐỐI TƯỢNG BÊN NGOÀI, THUỘC TÍNH NÀO THÌ PHƯƠNG THỨC NẤY. TỨC LÀ CODE BÊN NGOÀI KHÔNG TRỰC TIẾP NHÌN THẤY VÀ THAY ĐỔI TRẠNG THÁI CỦA ĐỐI TƯỢNG ĐÓ.***

Nói chung trạng thái đối tượng không hợp lệ thường do: chưa được kiểm tra tính hợp lệ, các bước thực hiện không đúng trình tự hoặc bị bỏ qua nên trong OOP có một quy tắc quan trọng cần nhớ đó là phải luôn khai báo các trạng thái bên trong của đối tượng là private và chỉ cho truy cập qua các public/protected method/property.

Ví dụ ta thấy một viên thuốc chữa cảm. Chúng ta chỉ biết nó chữa cảm sổ mũi nhức đầu và một số thành phần chính, còn cụ thể bên trong nó có những hoạt chất gì thì hoàn toàn không biết.

### ****Tính kế thừa (Inheritance)****

Nó cho phép xây dựng một lớp mới dựa trên các định nghĩa của lớp đã có. Có nghĩa là lớp cha có thể chia sẽ dữ liệu và phương thức cho các lớp con. Các lớp con khỏi phải định nghĩa lại, ngoài ra có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới. Tái sử dụng mã nguồn 1 cách tối ưu, tận dụng được mã nguồn. Một số loại kế loại kế thừa thường gặp: đơn kế thừa, đa kế thừa, kế thừa đa cấp, kế thừa thứ bậc.

Khi bắt đầu xây dựng ứng dụng chúng ta sẽ bắt đầu thiết kế định nghĩa các lớp trước. Thông thường một số lớp có quan hệ với những lớp khác, chúng có những đặc tính giống nhau.

VD: 2 lớp Android, iPhone

Mỗi lớp đều đại diện cho một loại smartphone khác nhau nhưng lại có những thuộc tính giống nhau như gọi điện, nhắn tin, chụp hình. Thay vì sao chép những thuộc tính này, ta nên đặt chúng vào một lớp chung gọi là lớp cha. Chúng ta có thể định nghĩa lớp cha – trong trường hợp này là Smartphone và có những lớp con kế thừa từ nó, tạo ra một mối quan hệ cha/con.

### ****Tính đa hình (Polymorphism)****

Tính đa hình là một hành động có thể được thực hiện bằng nhiều cách khác nhau. Đây lại là một tính chất có thể nói là chứa đựng hầu hết sức mạnh của lập trình hướng đối tượng.

Hiểu một cách đơn giản hơn: Đa hình là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau.

Ví dụ như ở phần trên, mỗi một smartphone kế thừa từ lớp cha Smartphone nhưng có thể lưu trữ dữ liệu trên cloud theo những cách khác nhau. Android lưu trữ bằng Google Drive iPhone lưu trên iCloud.

Một ví dụ về đa hình trong thực tế. Ta có 2 con vật: chó, mèo. Cả 2 con vật này đều là lớp động vật. Nhưng khi ta bảo cả 2 động vật kêu thì con chó sẽ kêu gâu gâu, con mèo sẽ kêu meo meo.

Vậy trong ví dụ chó, mèo xem như là các đối tượng. 2 con vật có thể hiểu cùng kêu nhưng theo các cách khác nhau.

### ****Tính trừu tượng (Abstraction)****

Trừu tượng có nghĩ là tổng quát hóa một cái gì đó lên, không cần chú ý chi tiết bên trong. Nó không màng đến chi tiết bên trong là gì và người ta vẫn hiểu nó mỗi khi nghe về nó.

Ví dụ: Bạn chạy xe tay ga thì có hành động là tăng ga để tăng tốc, thì chức năng tăng ga là đại diện cho trừu tượng (abstraction). Người dùng chỉ cần biết là tăng ga thì xe tăng tốc, không cần biết bên trong nó làm thế nào.

Ở đây trong lập trình OOP, tính trừu tượng nghĩa là chọn ra các thuộc tính, phương thức của đối tượng cần cho việc giải quyết bài toán đang lập trình. Vì một đối tượng có rất nhiều thuộc tính phương thức, nhưng với bài toán cụ thể không nhất thiết phải chọn tất cả.

Ví dụ: Bài toán quản lý sinh viên chúng ta chỉ cần quản lý các thông tin như

* Họ tên
* Ngày sinh
* Giới tính
* …
* Điểm thi

mà lại không cần quản lý thêm các thông tin:

* Màu tóc
* Sở thích
* Chiều cao

Tại vì chúng thực sự không cần thiết.

## **Các ưu điểm của lập trình hướng đối tượng**

* Dựa trên nguyên lý kế thừa, trong quá trình mô tả các lớp có thể loại bỏ những chương trình bị lặp, dư. Và có thể mở rộng khả năng sử dụng các lớp mà không cần thực hiện lại. Tối ưu và tái sử dụng code hiệu quả.
* Đảm bảo rút ngắn thời gian xây dựng hệ thống và tăng năng suất thực hiện.
* Sự xuất hiện của 2 khái niệm mới là **lớp**và **đối tượng** chính là **đặc trưng của phương pháp lập trình hướng đối tượng**. Nó đã giải quyết được các khuyết điểm của phương pháp lập trình hướng cấu trúc để lại. Ngoài ra 2 khái niệm này đã giúp biểu diễn tốt hơn thế giới thực trên máy tính.

Ba yếu tố quan trọng của lập trình hướng đối tượng là:

* **Tính mô-đun:** nó tách biệt các nhiệm vụ trong quá trình phát triển phần mềm dựa trên những đối tượng cụ thể. Mỗi đối tượng sẽ có một nhiệm vụ khác nhau.
* **Tính mở rộng**: các đối tượng có thể được mở rộng dễ dàng, bao gồm mở rộng thuộc tính và các hành vi mới.
* **Tính tái sử dụng:**các đối tượng cũng có thể được sử dụng lại trong một ứng dụng hoặc nhiều ứng dụng khác nhau.

Ba yếu tố trên của lập trình hướng đối tượng giúp hiệu năng phát triển phần mềm cũng được cải thiện rất nhiều, so với kỹ thuật lập trình truyền thống dựa trên thủ tục.

1. ***What is boxing and unboxing?***
2. ***Phân biệt readonly và const?***

Trong C#, một từ khóa const được sử dụng để khai báo các biến có giá trị không đổi và phải được gán giá trị khi khai báo. Giá trị của nó giống nhau trong suốt chương trình(tức là lúc compile time) hoặc nói cách khác một khi hằng số được gán giá trị thì không được thay đổi. Còn readonly cũng làm cho giá trị của biến không thể bị thay đổi, readonly được khởi tạo lúc khai báo hoặc khởi tạo trong constructor (tức là lúc runtime)

1. ***Các quy tắc RESTful API***

Sử dụng định dạng Json để gửi và nhận data

Sử dụng HTTP Method GET, POST, PUT, DELETE

Respone trả về status code

Xây dựng API endpoint và đặt tên Url đúng cách

1. ***Khi viết một câu lệnh Linq to SQL, khi nào thì câu lệnh mới thực sự được thực thi và lấy dữ liệu ra?***

Khi được gọi trong vòng lặp foreach, hoặc bằng các method như ToList, ToArray, Count, First()…

1. ***Sự khác biệt giữa Array và các loại collection khác?***

array phải khai báo size từ đầu – collection có thể thay đổi sau.

Array chỉ có thể chứa cùng một loại dữ liệu(homogeneous) – collection có thể chứa nhiều loại dữ liệu(heterogeneous – có thể tốn tài nguyên hơn do phải box và unbox)

Collection thường có các method đi kèm để hỗ trợ thao tác trên chuỗi như add, remove…

Array dữ liệu được bố trí cạnh nhau trong bộ nhớ. Collection bố trí kiểu danh sách liên kết

Ưu nhược điểm:

Nếu chỉ đọc thì nên dùng array, vì data được lưu trong các ô nhớ liên kết nhau nên đọc nhanh, còn nếu tạo mới, xóa, thêm dữ liệu vào giữa collection chẳng hạn thì nên dùng linkedlist, vì nếu thêm phần tử vào giữa list, thì chỉ cần thao tác với các phần tử kế cạnh, còn nếu thêm phần tử vào giữa array, thì phải recocate lại cả array, tốn bộ nhớ vcl ra

Ưu điểm của array là tốc độ đọc nhanh, ưu điểm của list là làm các thao tác tạo mới, xóa, chỉnh sửa tối ưu hơn.

1. ***Html : khác nhau của position và apsolute***
2. ***Fullinjoin với innerjoin***
3. ***Khác nhau của monothic vs microservice***