

**Báo cáo NET102**

**Dotnet C# 2**

**LAB 2**

Mã số sinh viên : PS28709

Họ tên sinh viên : Quan Bích Vân

Lớp : IT18322

Giảng viên hướng dẫn : Trần Thế Bảy

MỤC LỤC

Contents

[Tạo Bảng menu Switch case và UI 3](#_Toc135851058)

[Phần Program.cs 3](#_Toc135851059)

[Câu 1: 4](#_Toc135851060)

[Bài 1.1 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau : 4](#_Toc135851061)

[Bài 1.2 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau : 6](#_Toc135851062)

[Bài 1.3 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau : 7](#_Toc135851063)

[Bài 1.4 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau : 8](#_Toc135851064)

[Câu 2: 9](#_Toc135851065)

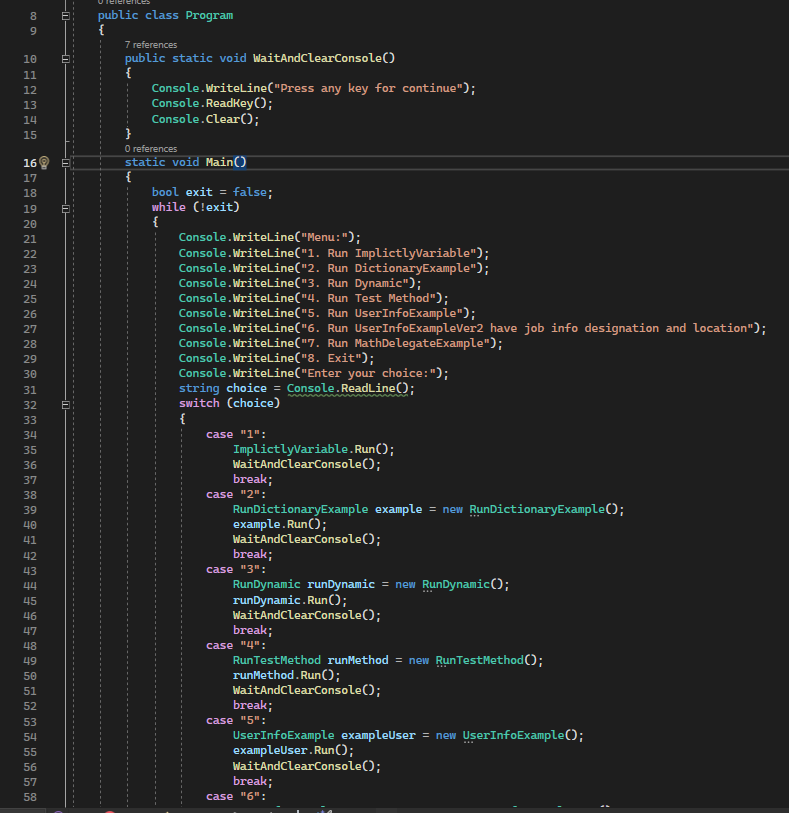
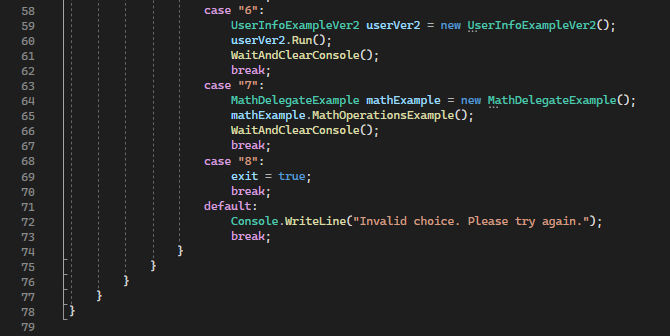
[Bài 2.1: Sử dụng Anonymous typed, Anonymous method và fix bug hoàn thiện các đoạn code sau: 9](#_Toc135851066)

[Bài 2.2: Sử dụng Anonymous typed, Anonymous method và fix bug hoàn thiện các đoạn code sau: 10](#_Toc135851067)

[Bài 2.3: Sử dụng Anonymous typed, Anonymous method và fix bug hoàn thiện các đoạn code sau: 11](#_Toc135851068)

# Tạo Bảng menu Switch case và UI

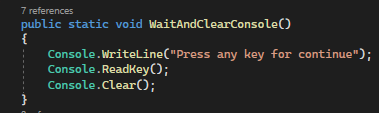
## Phần Program.cs

Trong hàm Main(), chương trình chạy trong một vòng lặp while với biến exit được khởi tạo là false.

Trong mỗi lần lặp, chương trình in ra màn hình menu các lựa chọn cho người dùng và yêu cầu người dùng nhập lựa chọn bằng cách đọc một dòng nhập từ bàn phím và lưu vào biến choice.

**Giải thích về WaitAndClearConsole()**



Dòng code Console.WriteLine("Press any key for continue"); sẽ in ra chuỗi "Press any key for continue" trên màn hình console.

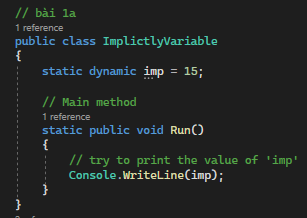
Dòng code Console.ReadKey(); chờ người dùng nhấn một phím bất kỳ trên bàn phím để tiếp tục. Khi người dùng nhấn phím, chương trình sẽ tiếp tục thực hiện các lệnh tiếp theo.

Dòng code Console.Clear(); sẽ xóa nội dung hiện tại trên màn hình console, làm cho màn hình trở nên trống trắng. Điều này giúp tạo ra một giao diện sạch sẽ khi chương trình tiếp tục chạy.

# Câu 1:

## Bài 1.1 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau :

Code:



Khai báo một lớp public có tên là ImplictlyVariable.

public class ImplictlyVariable

Dòng 3: (static dynamic imp = 15;) Khai báo một biến tĩnh (static) có kiểu dữ liệu là dynamic và giá trị ban đầu là 15.

dynamic là một kiểu dữ liệu động trong C#, cho phép biến có thể chứa các giá trị của bất kỳ kiểu dữ liệu nào và có thể thay đổi kiểu dữ liệu trong quá trình chạy. Trong trường hợp này, biến imp được khai báo là dynamic và gán giá trị ban đầu là 15.

Dòng 7-11: Khai báo một phương thức public tĩnh (static) có tên là Run. Và in ra giá trị của biến imp bằng cách sử dụng Console.WriteLine.

Console.WriteLine được sử dụng để in giá trị của biến hoặc biểu thức ra màn hình console. Trong trường hợp này, giá trị của biến imp sẽ được in ra.

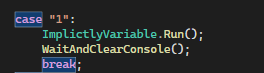
static public void Run()

{

Console.WriteLine(imp);

}

Trong hàm main:



**ImplictlyVariable.Run();**

Gọi phương thức Run() của lớp ImplictlyVariable. Điều này sẽ thực hiện các lệnh trong phương thức Run() để in giá trị của biến imp ra màn hình console.

**WaitAndClearConsole();**

Gọi phương thức WaitAndClearConsole(). Điều này sẽ hiển thị thông báo cho người dùng và chờ cho đến khi người dùng nhấn một phím bất kỳ để tiếp tục.

Dòng này gọi phương thức tĩnh (static) WaitAndClearConsole().

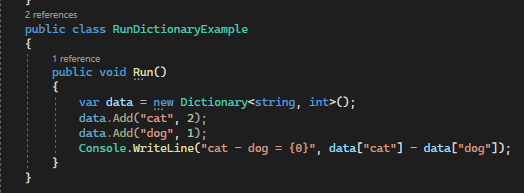
**break;**

Dòng này sẽ thoát khỏi switch case và chương trình tiếp tục thực hiện lệnh sau switch case.

break được sử dụng để thoát khỏi một vòng lặp hoặc một switch case. Trong trường hợp này, khi case "1" được thực thi xong, dòng break sẽ kết thúc switch case và chương trình sẽ tiếp tục thực hiện các lệnh sau switch case nếu có.

## Bài 1.2 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau :

Code:



Dòng 1: Khai báo một lớp public có tên là RunDictionaryExample.

Dòng 3-8: Khai báo một phương thức không tĩnh (non-static) có tên là Run.

Dòng 5: Khởi tạo một đối tượng Dictionary với kiểu key là string và kiểu value là int, và gán đối tượng đó cho biến data.

* Dictionary<TKey, TValue> là một cấu trúc dữ liệu trong C# được sử dụng để lưu trữ các cặp key-value.
* Trong trường hợp này, đối tượng Dictionary được tạo với kiểu key là string và kiểu value là int.

Dòng 6-7: Thêm cặp key-value vào data.

* data.Add("cat", 2) thêm một cặp key-value vào data, key là "cat" và value là 2.
* data.Add("dog", 1) thêm một cặp key-value khác vào data, key là "dog" và -value là 1.

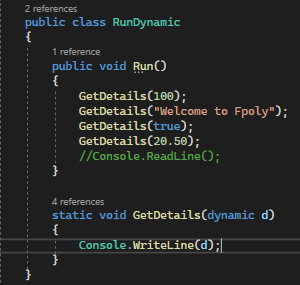
Dòng 8: In ra kết quả "cat - dog = {0}" với giá trị được tính bằng data["cat"] - data["dog"].

* data["cat"] truy cập giá trị tương ứng với key "cat" trong data, trả về 2.
* data["dog"] truy cập giá trị tương ứng với key "dog" trong data, trả về 1.
* Kết quả của phép tính data["cat"] - data["dog"] là 2 - 1 = 1.

Dòng Console.WriteLine("cat - dog = {0}", data["cat"] - data["dog"]); in ra kết quả "cat - dog = 1" trên màn hình console.

## Bài 1.3 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau :

Code:



Dòng 1: Khai báo một lớp public có tên là RunDynamic.

Dòng 3-9: Khai báo một phương thức không tĩnh (non-static) có tên là Run.

Dòng 5-8: Gọi phương thức GetDetails với các đối số khác nhau.

Dòng 5: Gọi phương thức GetDetails với đối số là 100.

Dòng 6: Gọi phương thức GetDetails với đối số là "Welcome to Fpoly".

Dòng 7: Gọi phương thức GetDetails với đối số là true.

Dòng 8: Gọi phương thức GetDetails với đối số là 20.50.

Dòng 11-14: Khai báo một phương thức tĩnh (static) có tên là GetDetails với một tham số kiểu dynamic.

Dòng 11: Khai báo phương thức GetDetails có một tham số dynamic có tên là d.

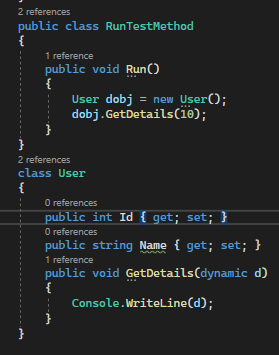
Dòng 12: In ra giá trị của tham số d bằng cách sử dụng Console.WriteLine.

Phương thức GetDetails có thể nhận đối số của bất kỳ kiểu dữ liệu nào do dynamic làm tham số, và in giá trị của đối số đó ra màn hình console.

Lúc chạy, phương thức Run sẽ gọi GetDetails với các đối số khác nhau và in giá trị của mỗi đối số đó ra màn hình console.

## Bài 1.4 Dùng từ khóa var hoặc dynamic và fix bug hoàn thiện các đoạn code sau :

Code:



**Phần RunTestMethod:**

Dòng 1: Khai báo một lớp public có tên là RunTestMethod.

Dòng 3-7: Khai báo một phương thức không tĩnh (non-static) có tên là Run.

Dòng 5: Tạo một đối tượng User mới và gán cho biến dobj.

Dòng 6: Gọi phương thức GetDetails của đối tượng dobj với đối số là 10.

**Lớp User:**

Dòng 9: Khai báo một lớp có tên là User.

Dòng 11-15: Khai báo một phương thức GetDetails có một tham số kiểu dynamic.

* Dòng 11: Khai báo phương thức GetDetails có một tham số dynamic có tên là d.
* Dòng 12: In ra giá trị của tham số d bằng cách sử dụng Console.WriteLine.

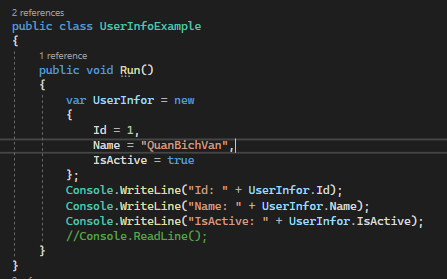
Phương thức GetDetails có thể nhận đối số của bất kỳ kiểu dữ liệu nào do dynamic làm tham số, và in giá trị của đối số đó ra màn hình console.

Khi chạy phương thức Run trong RunTestMethod, một đối tượng User mới được tạo, sau đó phương thức GetDetails của đối tượng User được gọi với đối số là 10. Phương thức GetDetails in ra giá trị của đối số lên màn hình console.

# Câu 2:

## Bài 2.1: Sử dụng Anonymous typed, Anonymous method và fix bug hoàn thiện các đoạn code sau:

Code:



Dòng 1: Khai báo một lớp public có tên là UserInfoExample.

Dòng 3-10: Khai báo một phương thức không tĩnh (non-static) có tên là Run.

Dòng 5: Tạo một đối tượng với kiểu ẩn (var) và gán cho biến UserInfor.

Dòng 6-8: Khai báo các thuộc tính của đối tượng UserInfor sử dụng cú pháp new { } để tạo một đối tượng với các thuộc tính được định nghĩa ngay lúc tạo.

* Thuộc tính Id có giá trị là 1.
* Thuộc tính Name có giá trị là "QuanBichVan".
* Thuộc tính IsActive có giá trị là true.

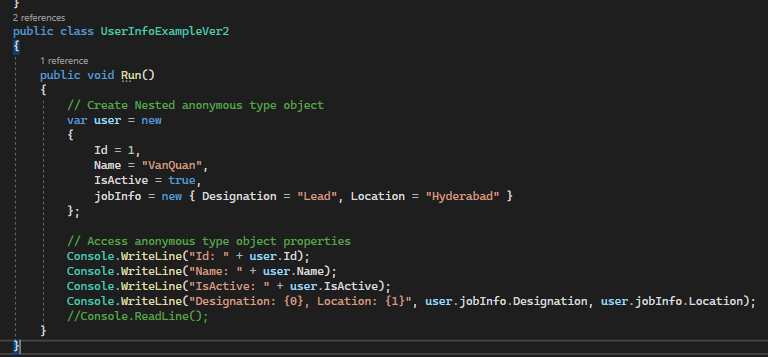
Dòng 9-11: In ra giá trị của các thuộc tính của đối tượng UserInfor bằng cách sử dụng Console.WriteLine.

* Dòng 9: In ra giá trị của thuộc tính Id của đối tượng UserInfor.
* Dòng 10: In ra giá trị của thuộc tính Name của đối tượng UserInfor.
* Dòng 11: In ra giá trị của thuộc tính IsActive của đối tượng UserInfor.

Khi chạy phương thức Run trong UserInfoExample, một đối tượng mới được tạo với các thuộc tính Id, Name và IsActive. Sau đó, các giá trị của các thuộc tính đó được in ra màn hình console.

## Bài 2.2: Sử dụng Anonymous typed, Anonymous method và fix bug hoàn thiện các đoạn code sau:

Code:



Dòng 1: Khai báo một lớp public có tên là UserInfoExampleVer2.

Dòng 3-11: Khai báo một phương thức không tĩnh (non-static) có tên là Run.

Dòng 5: Tạo một đối tượng với kiểu ẩn (var) và gán cho biến user.

Dòng 6-10: Khai báo các thuộc tính của đối tượng user sử dụng cú pháp new { } để tạo một đối tượng với các thuộc tính được định nghĩa ngay lúc tạo.

* Thuộc tính Id có giá trị là 1.
* Thuộc tính Name có giá trị là "VanQuan".
* Thuộc tính IsActive có giá trị là true.
* Thuộc tính jobInfo là một đối tượng ẩn khác được tạo ra bằng cách sử dụng cú pháp new { } với các thuộc tính Designation và Location.

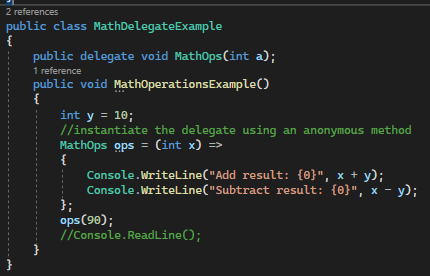
Dòng 13-16: In ra giá trị của các thuộc tính của đối tượng user bằng cách sử dụng Console.WriteLine.

* Dòng 13: In ra giá trị của thuộc tính Id của đối tượng user.
* Dòng 14: In ra giá trị của thuộc tính Name của đối tượng user.
* Dòng 15: In ra giá trị của thuộc tính IsActive của đối tượng user.
* Dòng 16: In ra giá trị của thuộc tính Designation và Location của đối tượng user.jobInfo bằng cách sử dụng cú pháp định dạng chuỗi {0} và {1}.

Khi chạy phương thức Run trong UserInfoExampleVer2, một đối tượng mới được tạo với các thuộc tính Id, Name, IsActive và jobInfo. Sau đó, các giá trị của các thuộc tính đó được in ra màn hình console, cùng với giá trị của thuộc tính Designation và Location của đối tượng jobInfo.

## Bài 2.3: Sử dụng Anonymous typed, Anonymous method và fix bug hoàn thiện các đoạn code sau:

Code:



Dòng 1: Khai báo một lớp public có tên là MathDelegateExample.

Dòng 3: Khai báo một delegate public có tên là MathOps. Delegate này chấp nhận một tham số kiểu int và không có giá trị trả về.

Dòng 5-15: Khai báo một phương thức có tên là MathOperationsExample.

Dòng 7: Khai báo một biến y có kiểu int và gán giá trị là 10.

Dòng 9-13: Tạo một đối tượng delegate ops bằng cách sử dụng một phương thức vô danh (anonymous method).

Phương thức vô danh này chấp nhận một tham số x kiểu int.

Dòng 10: In ra kết quả của phép cộng x + y sử dụng Console.WriteLine.

Dòng 11: In ra kết quả của phép trừ x - y sử dụng Console.WriteLine.

Dòng 14: Gọi delegate ops với đối số là 90, điều này sẽ kích hoạt phương thức vô danh được gán cho delegate.

Chương trình sẽ in ra kết quả của phép cộng và phép trừ trên màn hình console với giá trị x là 90 và y là 10.

Khi chạy phương thức MathOperationsExample trong MathDelegateExample, một delegate mới được tạo và gán với một phương thức vô danh. Sau đó, phương thức vô danh này được gọi thông qua delegate, và kết quả của các phép cộng và phép trừ được in ra màn hình console.