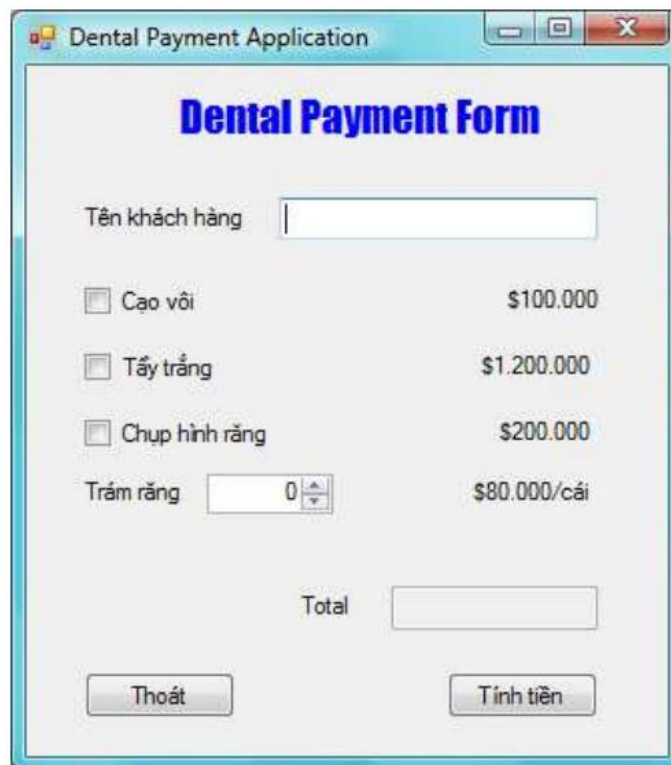


Ứng dụng WinForm (basic)

Bài 1: Tạo một ứng dụng Windows Form cơ bản tính tiền công dịch vụ cho một lần đi khám tại phòng nha:

Với mỗi khách hàng, các dịch vụ cung cấp gồm: tẩy răng, cạo vôi, chụp hình răng và trám răng. Mỗi loại sẽ có chi phí riêng. Cuối cùng tính tổng các chi phí mà người khách phải trả. Lưu ý: chỉ tính tiền khi phần thông tin tên khách hàng đã được nhập (nếu thông tin này chưa có thì chương trình phát sinh MessageBox cảnh báo).

Ứng dụng có giao diện đơn giản như hình dưới:

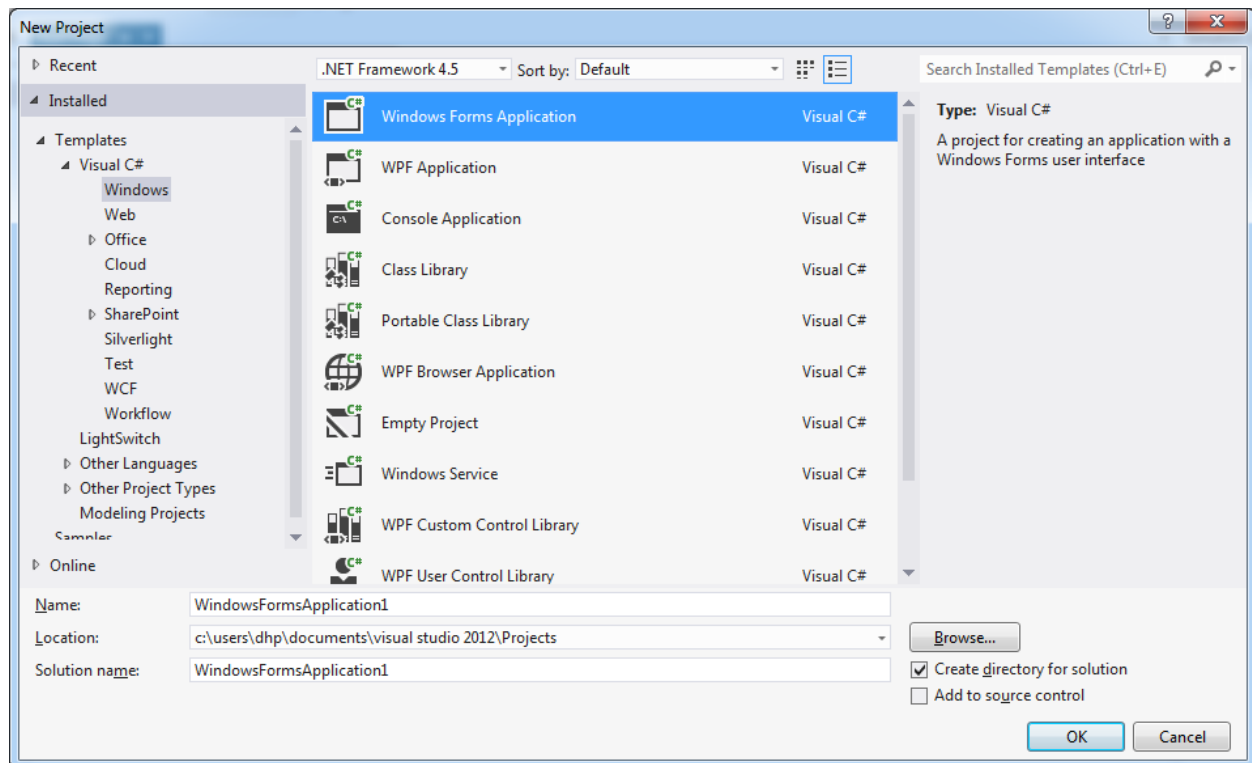


Dịch vụ	Chi phí
<input type="checkbox"/> Cạo vôi	\$100.000
<input type="checkbox"/> Tẩy trắng	\$1.200.000
<input type="checkbox"/> Chụp hình răng	\$200.000
Trám răng: 0	\$80.000/cái
Total	

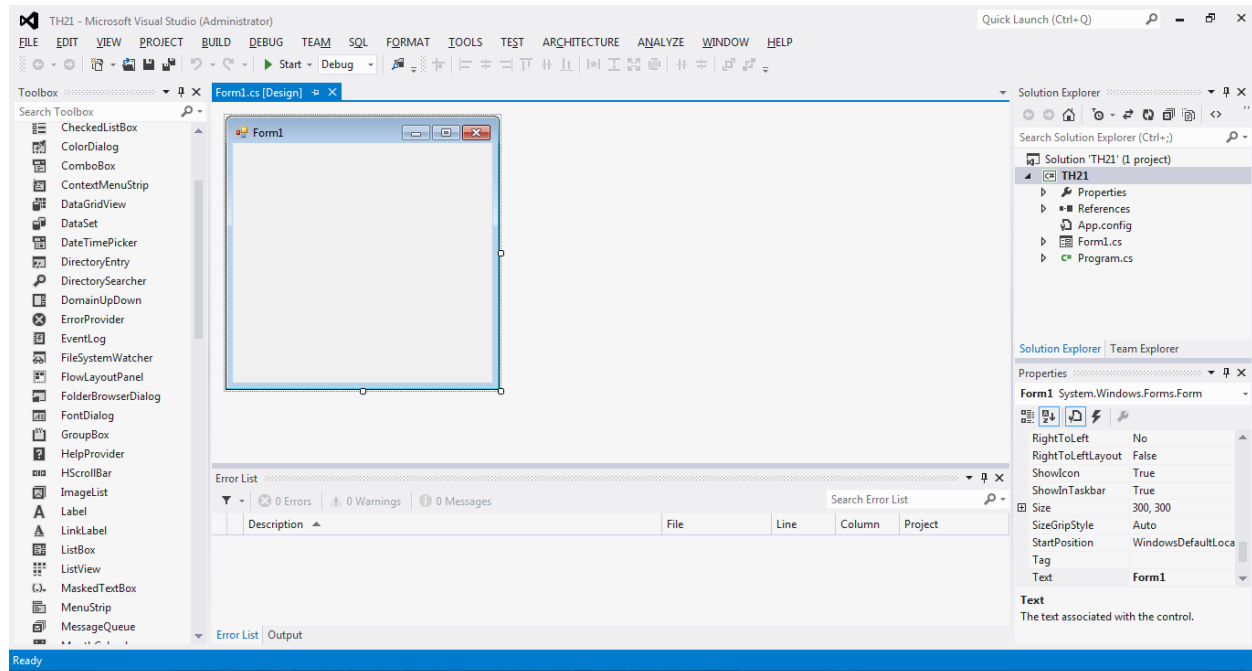
Hướng dẫn thực hiện:

1) Tạo ứng dụng Windows Form:

- Tạo project mới (Ctrl+Shift +N),
- Trong cửa sổ new project chọn Visual C# - Windows
- Phần template chọn Windows Forms Application
- Đặt tên project trong phần Name



2) Sau khi hoàn tất các bước trên VS.NET sẽ phát sinh ra một project Windows Form mẫu, cho phép người lập trình bắt đầu xây dựng các ứng dụng. Giao diện của VS.NET 2012 cho ứng dụng vừa tạo có dạng như hình bên dưới.



Màn hình VS.NET cho ứng dụng Windows Form bao gồm các phần cơ bản:

- Toolbox: chứa các control cho phép kéo thả vào form;
- Màn hình thiết kế form, có thể chuyển sang phần code editor, ...
- Cửa sổ Solution Explorer: cho phép người lập trình có thể quản lý các thành phần trong project, hỗ trợ định vị nhanh chóng đến các file mã nguồn.
- Cửa sổ property: cho phép user có thể custom lại các thành phần control trên form như: thiết lập các thuộc tính cho control, form, component, cho phép khai báo trình xử lý sự kiện của các control trên form, ...

3) Thiết kế form theo mô tả như sau:

STT	Name	Control	Thiết lập properties
1	lblTitle	Label	Text = “Dental Payment Form”, Font = “Impact, Size = 17”, ForeColor = Blue.
2	lblName	Label	Text = “Tên khách hàng”
3	txtName	TextBox	
4	chkClean	CheckBox	Text = “Cạo vôi”
5	lblCleanCost	Label	Text = “\$100000”
6	chkWhitening	CheckBox	Text = “Tẩy trắng”
7	lblWhiteningCost	Label	Text = “\$1200000”
8	chkXRay	CheckBox	Text = “Chụp hình răng”
9	lblXRayCost	Label	Text = “\$200000”
10	lblFilling	Label	Text = “Trám răng”
11	numFilling	NumericUpDown	
12	lblFillCost	Label	Text = “\$80000”
13	lblTotal	Label	Text = “Total”
14	txtTotal	TextBox	Enable = False
15	btnExit	Button	Text = “Thoát”
16	btnCalc	Button	Text = “Tính tiền”

Dental Payment Form

1. Title: Dental Payment Form

2. Text label: Tên khách hàng

3. Text input field

4. Checkbox: Cạo vôi

5. Text label: \$100.000

6. Checkbox: Tẩy trắng

7. Text label: \$1.200.000

8. Checkbox: Chụp hình răng

9. Text label: \$200000

10. Text label: Trám răng

11. Spin box with value 0

12. Text label: \$80000/cái

13. Text label: Total

14. Text input field

15. Button: Thoát

16. Button: Tính tiền

4) Tạo sự kiện cho Button “Thoát” và “Tính tiền”

Cách 1: Kích đúp vào button cần tạo trình xử lý sự kiện trong màn hình Form design view: khi đó VS sẽ tạo trình xử lý sự kiện gắn với sự kiện Click của button “Thoát”

Dental Payment Form

Tên khách hàng

☐ Cạo vôi \$100.000

☐ Tẩy trắng \$1.200.000

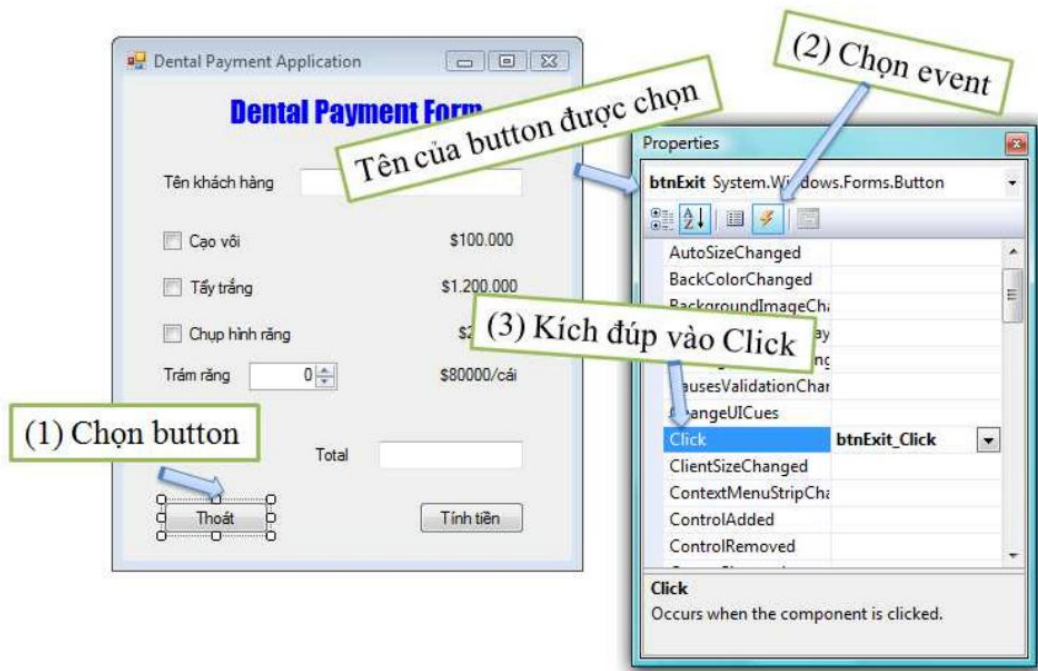
\$200000

\$80000/cái

Total

Kích đúp vào button để tạo trình xử lý sự kiện

Cách 2: chọn button cần tạo trình xử lý, sau đó kích tab event trong cửa sổ Properties, kích đúp vào mục Click trong cửa sổ event.



Nội dung của trình xử lý sự kiện Click của button btnExit như sau:

```
private void btnExit_Click(object sender, EventArgs e)
{
    //Close form
    this.Close();
}
```

5) Tạo chức năng tính tiền, chức năng này được kích hoạt khi button “Tính tiền” được chọn. Mô tả chức năng GetPay() như sau (GetPay() là phương thức thành viên của lớp Form chính):

- Kiểm tra xem tên khách hàng có được nhập hay không?
 - Nếu chưa: xuất thông báo, yêu cầu nhập tên khách.
 - Đã nhập: thực hiện các bước sau:
 - Total = 0
 - If (cạo vôi) Total += 100.000
 - If (tẩy trắng) Total += 1.200.000
 - If (chụp hình răng) Total +=200000
 - Total += (sốrăng trám)*80000
 - Xuất số tiền ra TextBox txtTotal

Tạo trình xử lý sự kiện cho button btnCalc rồi trong trình xử lý sự kiện này gọi chức năng GetPay():

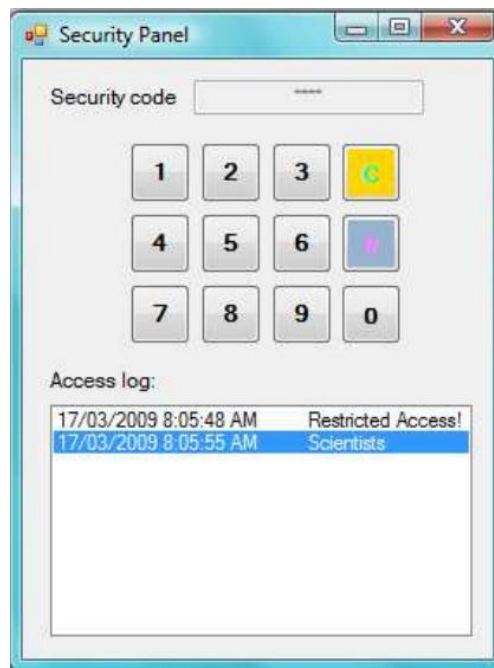
```
private void btnCalc_Click(object sender, EventArgs e)
{
    GetPay();
}
```

Bài 2: Xây dựng ứng dụng theo yêu cầu sau:

Một phòng lab muốn thiết lập một Security Panel đặt bên ngoài cửa. Chỉ cho phép những nhân viên có trách nhiệm mới được vào và mỗi lần đăng nhập họ phải nhập các security code được cung cấp. Những security code (hay còn gọi là access code) sau được xem là hợp lệ và cung cấp cho các nhóm nhân viên như bảng sau:

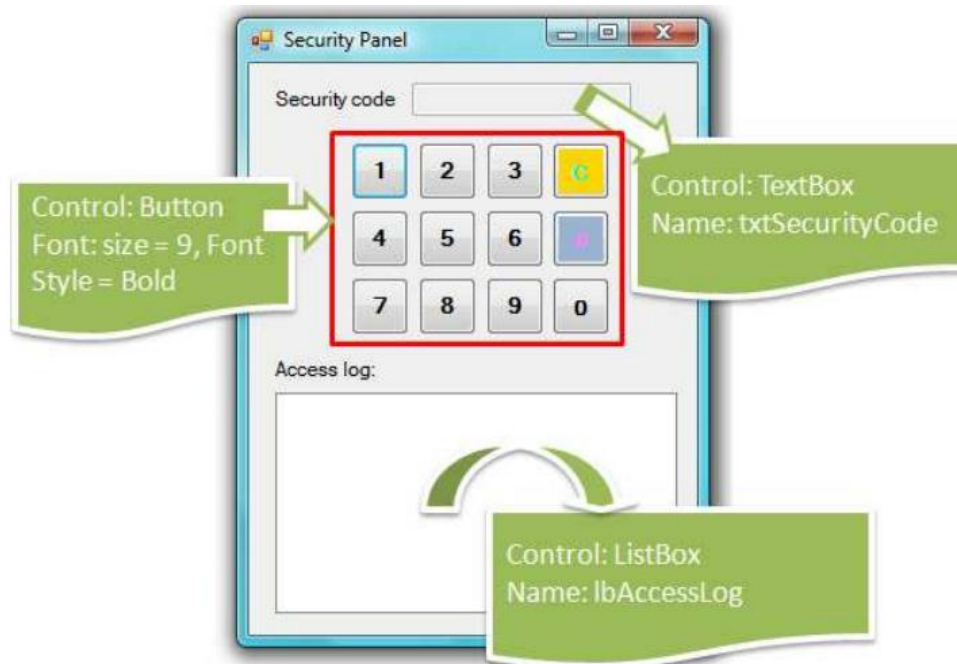
Value	Group
1645 or 1689	Technicians
8345	Custodians
9998, 1006 - 1008	Scientist

Một khi nhập access code thì sẽ có hai trạng thái: granted hoặc denied. Tất cả thông tin truy cập sẽ được hiển thị trong một khung thông tin bên dưới keypad. Nếu access là granted thì ngày, thời gian, group (technician, custodian, scientist) sẽ được hiển thị ở khung thông tin. Trường hợp access là denied thì ngày, giờ và thông tin “Access denied” sẽ hiển thị ở khung bên dưới. Ngoài ra user nếu chỉ nhấn một con số security code thì sẽ hiển thị ra thông báo là ngày, giờ và “Restricted Access”.

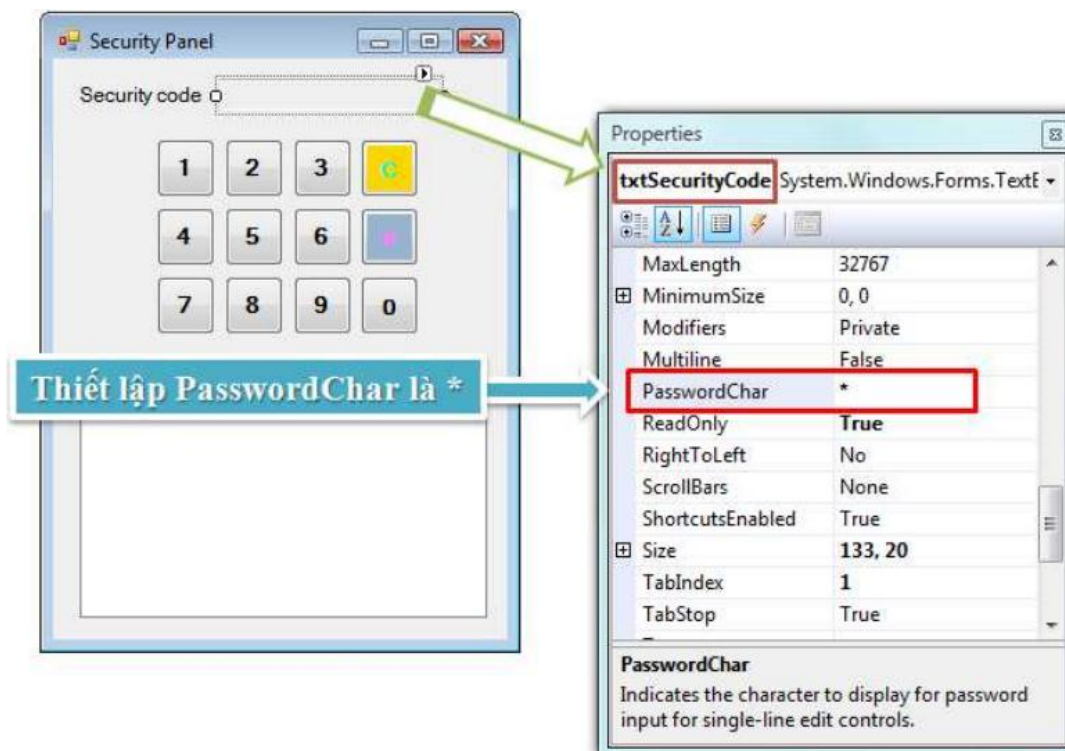


Hướng dẫn:

- 1) Tạo ứng dụng Windows Forms Application ...
- 2) Thiết kế Form như hình 1 minh họa
 - a) Mô tả các control trên form



b) Thiết lập thuộc tính PasswordChar của TextBox Security Code là “*”



3) Xử lý cho các Button trên form (Sinh viên tự viết)

4) Bổ sung chức năng log file: tất cả thông tin login dù access granted hay denied đều được ghi nhận vào file dạng text. File này được lưu trữ cùng với thư mục của ứng dụng.

Bài 3: Xây dựng ứng dụng Windows Forms mô phỏng theo mô tả như sau:

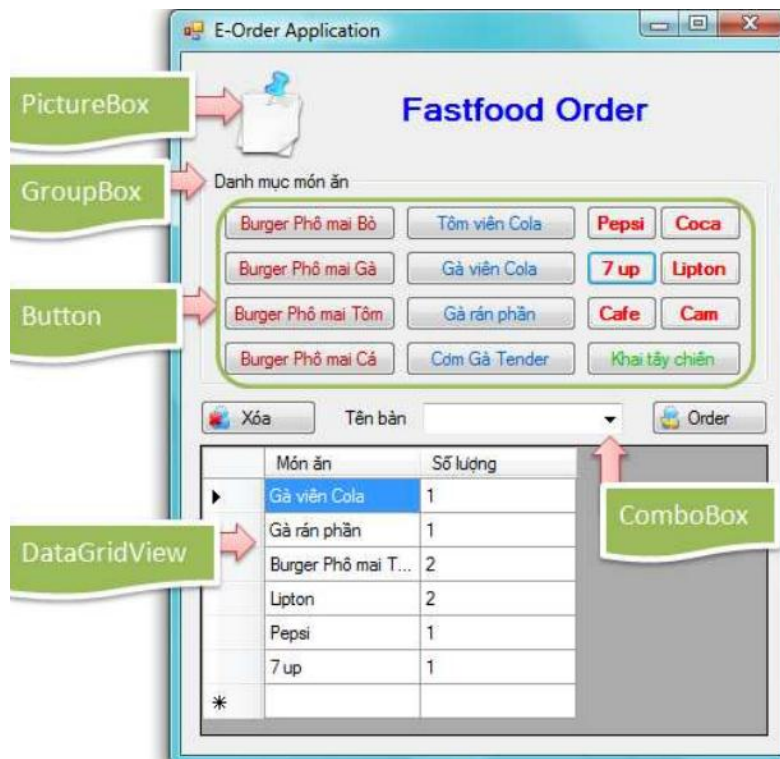
Tại một quán ăn nhanh, người ta muốn toàn bộ công việc order các món ăn được diễn ra một cách nhanh chóng và chuẩn hóa. Nên họ xây dựng một hệ thống e-order, hệ thống này được thực hiện thông qua một chương trình order cài đặt trên máy PDA, mỗi người phục vụ sẽ được cung cấp một PDA, khi khách hàng gọi món thì người phục vụ này sẽ đến tận bàn, và sử dụng chương trình e-order đó trên PDA để order món ăn. Khi việc order xong thì người phục vụ sẽ chọn chức năng send order và thông tin này sẽ được gửi xuống nhà bếp thông qua hệ thống wireless được cài đặt...

Sinh viên hãy viết lại chương trình order trên theo dạng Windows Form. Giao diện chương trình Order được thể hiện như hình 1.

Món ăn	Số lượng
Gà viên Cola	1
Gà rán phần	1
Burger Phô mai T...	2
Lipton	2
Pepsi	1
7 up	1
*	

Thao tác sử dụng:

- Người phục vụ sẽ chọn tên bàn được list trong ComboBox, sau đó tùy theo yêu cầu gọi món của client mà người phục vụ sẽ chọn món ăn, thức uống thông qua danh mục món ăn được thể hiện bởi danh sách các button. Mỗi lần chọn món ăn sẽ bổ sung thêm số lượng gọi món là 1, ví dụ: 2 lần chọn Bugar Phô mai Bò thì số lượng là 2 và danh mục gọi món của bàn đó sẽ hiển thị trong danh sách bên dưới.
- Kết thúc quá trình gọi món ăn thì người phục vụ sẽ chọn chức năng “Order”, thông tin này sẽ được gửi cho đầu bếp...



Hướng dẫn:

- Trong chương trình sử dụng lớp **DataTable** để chứa thông tin order, bao gồm có 2 cột: {FoodName} chứa tên món ăn và {Quantity} số lượng. Cách tạo bảng này như sau:
VD: biến DataTable trong Form1 là dt thì code tạo bảng chứa dữ liệu order là:

```
tb.Columns.Add("FoodName"); //thêm cột (Field) FoodName
```

```
tb.Columns.Add("Quantity"); //thêm cột Quantity
```

- Mỗi khi click vào món ăn thì chương trình sẽ tìm trong DataTable này xem có món ăn đó chưa, nếu chưa có thì thêm dòng mới vào với tên món ăn và số lượng là 1. Ngược lại đã có chọn món này thì số lượng của nó tăng 1.

Cách thêm một dòng (món ăn) mới vào DataTable dt:

```
DataRow r = tb.NewRow(); //tạo dòng mới theo mô tả bảng
```

```
//thiết lập cột FoodName với món ăn được chọn từ Button
```

```
r["FoodName"] = <tên món ăn được chọn>;
```

```
r["Quantity"] = 1; //thiết lập cột Quantity
```

```
tb.Rows.Add(r); //thêm vào bảng
```

- Sử dụng thuộc tính DataSource của DataGridView để kết buộc với dữ liệu trong DataTable
 - VD: tên của DataGridView trong chương trình là dataGridView1 và biến DataTable là dt thì code kết buộc như sau:

//binding nội dung trong DataTable cho DataGridView

```
dataGridView1.DataSource = dt;
```

Bài 4: Xây dựng ứng dụng Windows Form minh họa quản lý thông tin sinh viên khoa CNTT, mục đích quản lý các thông tin cơ bản của các sinh viên, bao gồm một số thông tin như sau:

- Họ tên
- Mã số sinh viên
- Ngày tháng năm sinh
- Địa chỉ
- Số điện thoại liên lạc
- Niên khóa

Ngoài các thông tin cơ bản trên mỗi sinh viên sẽ có thông tin là hệ đào tạo, có 3 loại hình đào tạo mà mỗi sinh viên sẽ thuộc về: đại học, cao đẳng, và bằng hai. Sinh viên thuộc hệ đại học sẽ được phân vào ba chuyên ngành: {CNPM, HTTT, Mạng MT}. Sinh viên cao đẳng thì không phân chuyên ngành. Học viên bằng hai thì có thêm thông tin: chuyên ngành bằng 1, đơn vị công tác.

Yêu cầu: Viết chương trình dạng Windows Form thực hiện các chức năng:

Phần code:

- Thông tin của một sinh viên sẽ được lưu vào một đối tượng SinhVien, chương trình có 1 đối tượng ArrayList chứa danh sách các đối tượng SinhVien.
- Lưu ý xây dựng một lớp SinhVien làm lớp cơ sở cho các lớp:
 - SinhVienDaiHoc
 - SinhVienCaoDang
 - SinhVienBangHai
- Tùy theo thông tin của sinh viên được nhập vào mà chương trình sẽ tạo các đối tượng tương ứng, ví dụ sinh viên đại học sẽ lưu vào đối tượng SinhVienDaiHoc, sinh viên cao đẳng sẽ được lưu vào đối tượng SinhVienCaoDang...

Phần GUI:

- Xây dựng Form 1 thành form nhập thông tin của sinh viên.
- Hiển thị danh sách toàn bộ sinh viên, cho phép chọn một sinh viên rồi sửa hoặc xóa.

Hướng dẫn:

1) Tạo ứng dụng Windows Application từ VS .NET:

- a. Chọn chức năng New -> Project.
- b. Trong cửa sổ New Project
- c. VS .NET phát sinh ra khuôn mẫu dạng ứng dụng Form

d. Sau bước Wizard của VS.NET, project được tạo với 2 lớp Form1 và Program.

- Lớp Form1 là lớp quản lý form của hình 2, lớp Form1 dẫn xuất từ lớp Form của namespace System.Windows.Forms.

Lớp Form1 được VS chia thành 2 phần, một phần code do VS phát sinh theo sự thiết kế của người lập trình trên form, code này chứa trong file form1.Designer.cs. Còn phần code cho phép người lập trình bổ sung là Form1.cs. Một lớp được chia thành nhiều file trong VS.NET nhờ kỹ thuật **partial**.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace StudentManaging
{
    public partial class Form1 : Form
    {
        public Form1() ← Lớp Form1
        {
            InitializeComponent(); ← Hàm khởi tạo các thành phần từ màn hình thiết kế form1, do VS quản lý
        }
    }
}
```

Code của Form1 chứa trong file Form1.Designer.cs.

```

partial class Form1
{
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Windows Form Designer generated code

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(284, 262);
        this.Name = "Form1";
        this.Text = "Form1";
        this.ResumeLayout(false);
    }

    #endregion
}

```

Phần code do VS.NET khởi tạo trong file Form1.Designer.cs.

- Lớp Program chứa hàm Main, trong đó sử dụng lớp Application để gọi Form1 thực hiện.

```

using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace StudentManaging
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

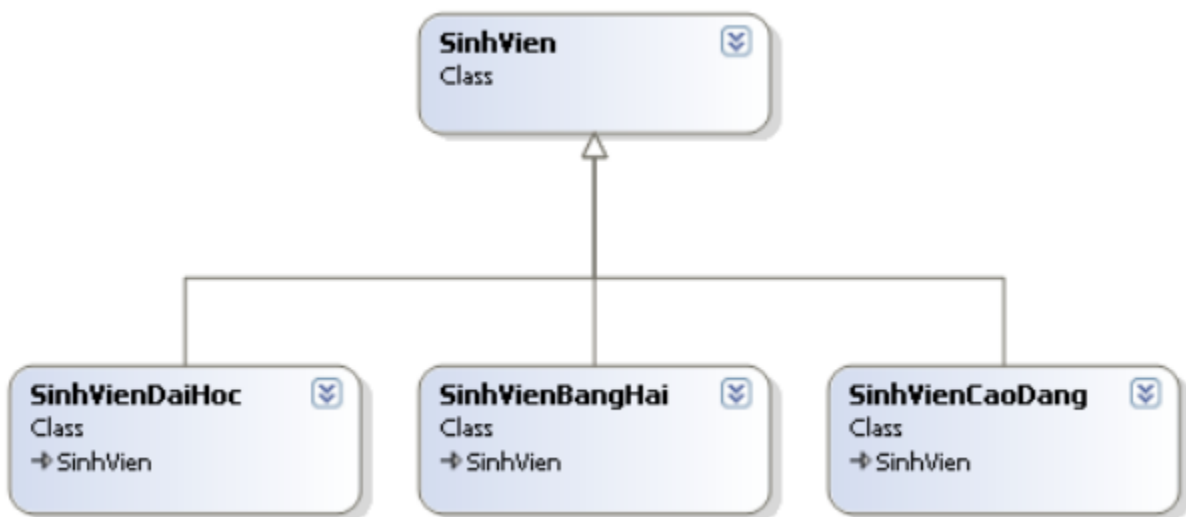
```

Phần code do VS.NET tạo

Gọi Form1 thực hiện

Tập tin program.cs.

2) Tạo các lớp liên quan đến việc lưu trữ thông tin của sinh viên. Xây dựng bốn lớp như sơ đồ lớp như hình sau:



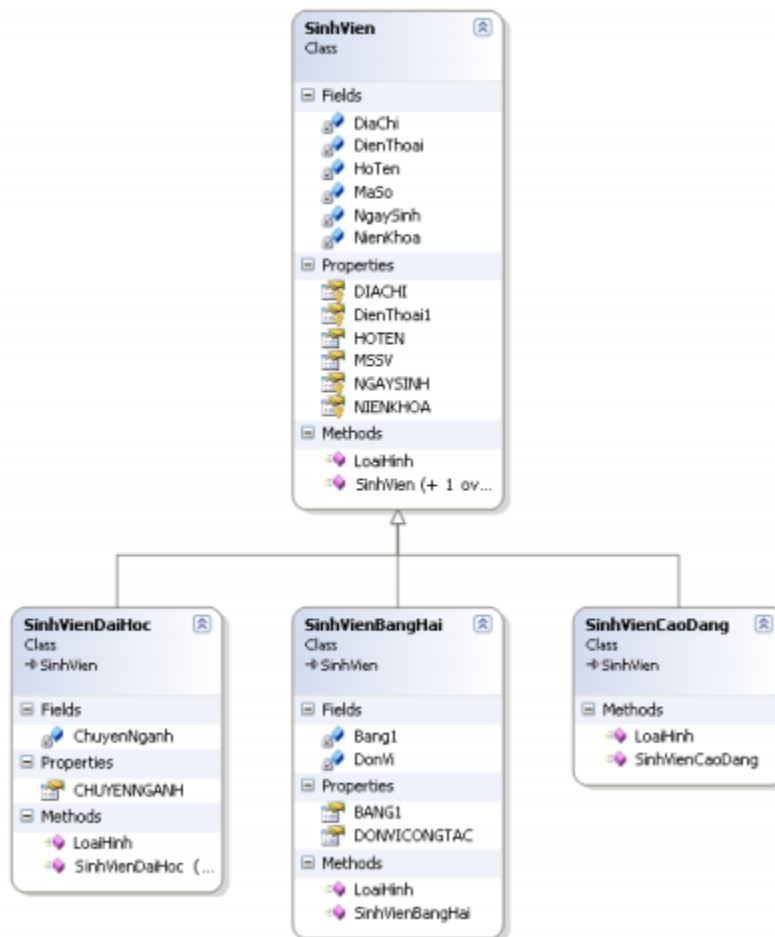
Mỗi lớp chứa trong một tập tin riêng: chọn cửa sổ Class View -> kích chuột phải vào tên của project rồi chọn chức năng Add -> class...

Phần code tạo các lớp này sinh viên tự xây dựng các phần như sau:

- Khai báo lớp

- Các hàm Constructor
- Các Property tương ứng cho các field
- Các phương thức của các lớp

Sinh viên có thể dựa theo mô tả sau để xây dựng các lớp hoặc hoàn toàn tùy ý làm cách khác.

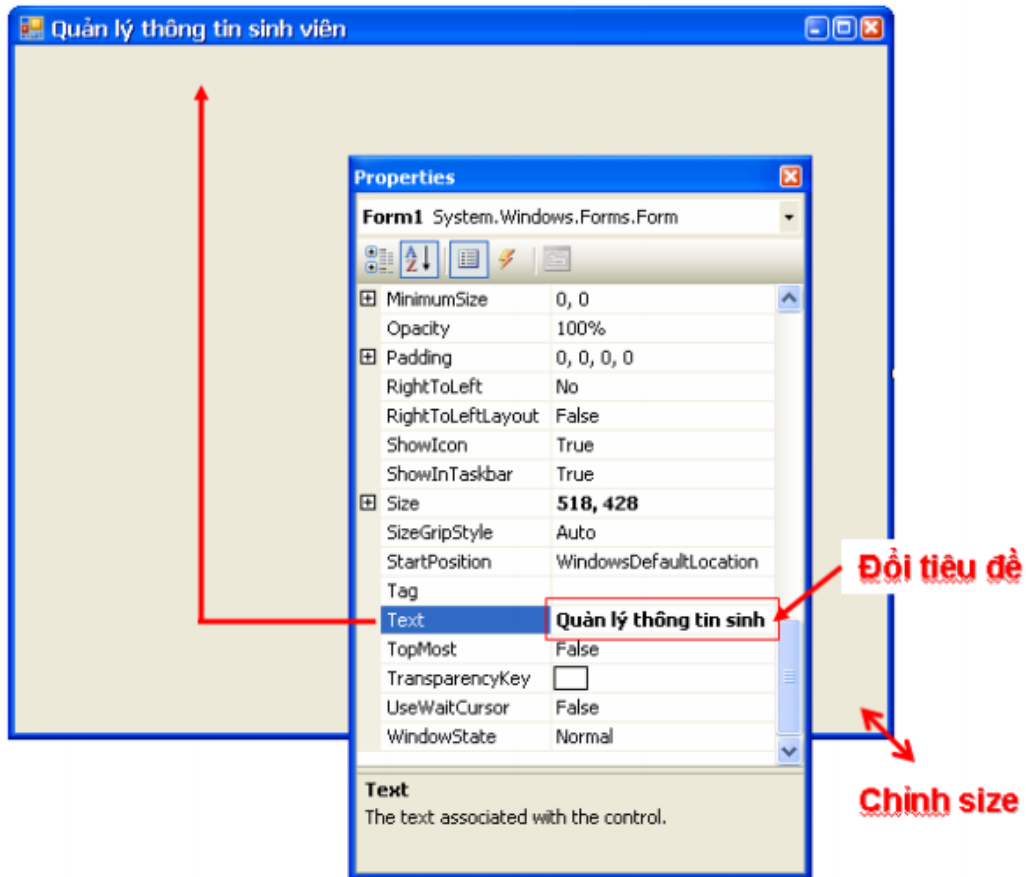


Trong đó phương thức **LoaiHinh()** trả về chuỗi cho biết thông tin loại hình đào tạo của sinh viên: {"đại học", "cao đẳng", "bằng hai"}. Đây là phương thức ảo của lớp **SinhVien**, các lớp dẫn xuất sẽ override lại...

Do tùy theo cách tiếp cận trong thiết kế nên các bạn có thể bổ sung tùy ý những phương thức nào thấy cần!

3) Thiết kế Form nhập liệu cho ứng dụng, chúng ta quay lại với Form1

- Hiệu chỉnh lại kích thước của form thích hợp hơn,
- Đổi tên tiêu đề của form, cũng chính là tiêu đề của App
- Thực hiện chỉnh sửa trong màn hình thiết kế form của Form1.



Bổ sung thông tin cho form.

Bổ sung các control vào form: hình bên dưới là 1 dạng trình bày của form1, các bạn có thể dựa trên đó để thiết kế cho tốt hơn.

Quản lý thông tin sinh viên

Quản lý thông tin sinh viên

Họ tên

Địa chỉ

Mã số SV

Điện thoại

Ngày sinh

Sunday, October 12, ▾

Niên khóa

Loại hình

☐ Đại học

☐ Bằng hai

☐ Cao đẳng

Chuyên ngành

▾

Bằng 1

Công tác

Thêm

Sửa

Xóa

Reset

Danh sách sinh viên

MSSV	Họ tên	Ngày sinh	Địa chỉ	Điện thoại	Niên khóa	Loại hình
------	--------	-----------	---------	------------	-----------	-----------

Thoát

4) Để tiện cho việc viết các phần xử lý, chúng ta sẽ đổi tên các control trong form (không cần đổi các control dạng Label và Groupbox).

a. Các đổi tên control trong form: trong màn hình design view của form, kích chọn vào control cần đổi, trong cửa sổ properties tương ứng của control đó, ta tiến hành đổi tên ở thuộc tính Name.

b. Đặt tên lại cho các control trên form1 theo hình mô tả bên dưới:

Quản lý thông tin sinh viên

Quản lý thông tin sinh viên

Họ tên: Địa chỉ:

Mã số SV: Điện thoại:

Ngày sinh: Sunday, October 12, Niên khóa:

Loại hình:

☐ Đại học ☐ Bằng hai ☐ Cao đẳng

Chuyên ngành: Bảng 1:

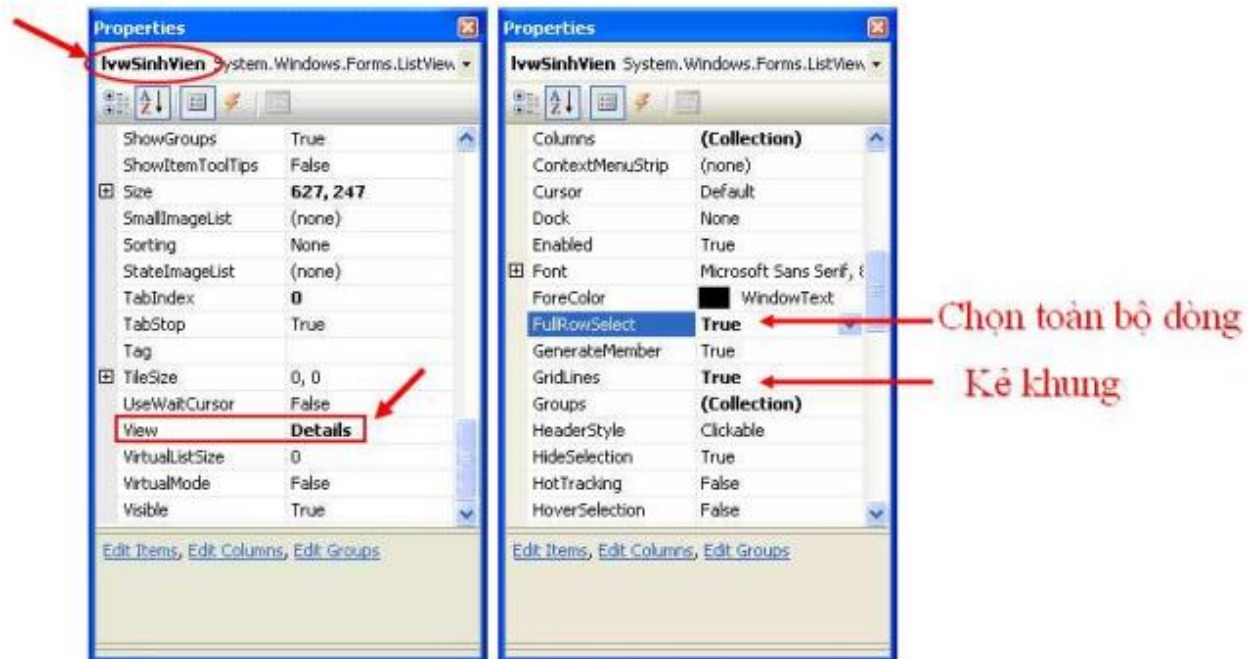
Công tác:

Danh sách sinh viên

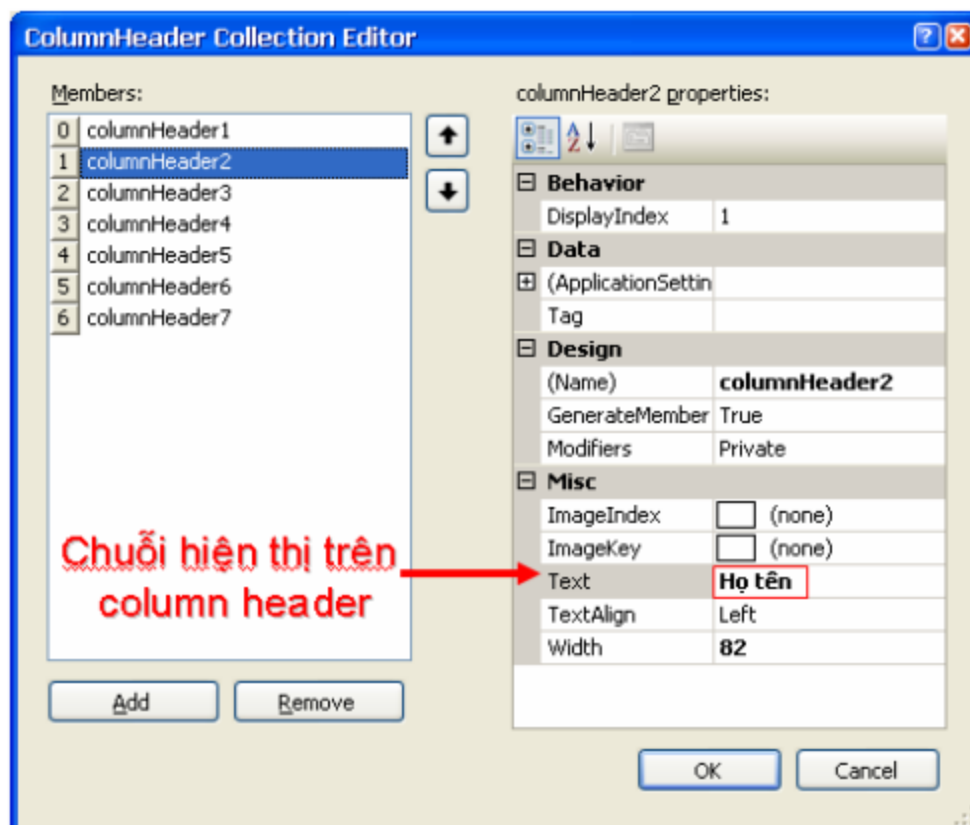
MSSV	Họ tên	Ngày sinh	Địa chỉ	Điện thoại	Niên khóa	Loại hình
<div>lvwSinhVien</div>						

5) Thiết lập các thuộc tính sau cho listview.

Thuộc tính	Giá trị	Ý nghĩa
View	Details	Hiển thị dạng bảng, có column header
FullRowSelect	True	Chọn toàn bộ dòng
GridLines	True	Hiển thị khung lưới



Thiết lập các Column header cho listview: chọn properties Columns sẽ hiện ra màn hình columnHeader Collection Editor.



Lần lượt thêm các column để ListView thể hiện có dạng sau:

7) Tạo biến kiểu ArrayList tên StudentList để chứa danh sách sinh viên được nhập vào. Biến StudentList là dữ liệu thành viên của Form1:

```
public partial class Form1 : Form
{
    // Danh sách sinh viên
    ArrayList StudentList = new ArrayList();

    public Form1()
    {
        InitializeComponent();
    }
}
```

8) Viết phương thức thành viên UpdateList của lớp Form1: phương thức này có chức năng xuất danh sách StudentList ra listview theo định dạng các cột đã mô tả.

```
void UpdateList()
{
    lvwSinhVien.Items.Clear();    // xóa các item trong listview
    int i = 0;
    foreach(SinhVien sv in StudentList) // duyệt qua các đối tượng
    {
        // thiết lập các giá trị cho item, mỗi item sẽ chứa thông tin
        // của 1 sinh viên

        // khởi tạo cột đầu tiên cho item, cột đầu chứa mã số.
        // item có một giá trị index cho biết thứ tự của item trong
        // listview, giá trị này là i
        ListViewItem item = new ListViewItem(sv.MSSV.ToString(), i++);

        // thiết lập các cột còn lại vào item
        item.SubItems.Add(sv.HOTEN);
        item.SubItems.Add(sv.NGAYSINH.ToString());
        item.SubItems.Add(sv.DIACHI);
        item.SubItems.Add(sv.DIENTHOAI.ToString());
        item.SubItems.Add(sv.NIENKHOA.ToString());
        item.SubItems.Add(sv.LoaiHinh());
        // thêm item vào danh sách items của listview

        lvwSinhVien.Items.Add(item);
    }
}
```

9) Viết phần xử lý cho button Thêm:

- a. Double click vào button Thêm trong màn hình design View của Form
- b. VS .NET sẽ tạo event handler cho Button Thêm, event handler này thuộc về lớp Form1.
- c. Nội dung của phần xử lý này, các bạn có thể tham khảo đoạn code sau:

```

private void btnAdd_Click(object sender, EventArgs)
{
    SinhVien sv = null;
    string ht = this.txtHoTen.Text;
    int ms = int.Parse(this.txtMSSV.Text);
    DateTime ns = this.dtpNS.Value;
    string dc = this.txtDiaChi.Text;
    int dt = int.Parse(this.txtDienThoai.Text);
    int nk = int.Parse(this.txtNienKhoa.Text);
    if (radDaiHoc.Checked) // nếu SV đại học -> tạo obj SinhVienDaiHoc
    {
        string cn = this.cboCN.Text;
        sv = new SinhVienDaiHoc(ht, ms, ns, dc, dt, nk, cn);
    }
    else if (radCaoDang.Checked) // SV CD -> tạo obj SinhVienCaoDang
    {
        sv = new SinhVienCaoDang(ht, ms, ns, dc, dt, nk);
    }
    else // trường hợp sinh viên bằng hai -> tạo obj SinhVienBangHai
    {
        string bang1 = this.txtBang1.Text;
        string cty = this.txtCty.Text;
        sv = new SinhVienBangHai(ht, ms, ns, dc, dt, nk, bang1, cty);
    }
    // lưu obj vừa tạo vào danh sách sinh viên
    this.StudentList.Add(sv);
    UpdateList(); // hiển thị danh sách ra listview.
}

```

10) Xây dựng form hoàn thiện hơn với chức năng chỉ cho phép nhập những dữ liệu cần thiết. Trong phần thông tin loại hình, user chọn 1 trong 3 loại hình {đại học, cao đẳng, bằng hai}, với mỗi hình thức đó sẽ có những thông tin chi tiết cụ thể. Ví dụ: user chọn loại hình Đại học, khi đó thông tin chuyên ngành là cần thiết, còn thông tin bằng 1 và cty là của loại hình bằng hai, không cần thiết, do đó ta sẽ invisible hay disable các control này.

a. Bước 1: viết một hàm xử lý sự kiện CheckedChanged chung cho cả ba radio button, hàm này sẽ kiểm tra xem radiobutton nào được check và sẽ enable những control tương ứng.

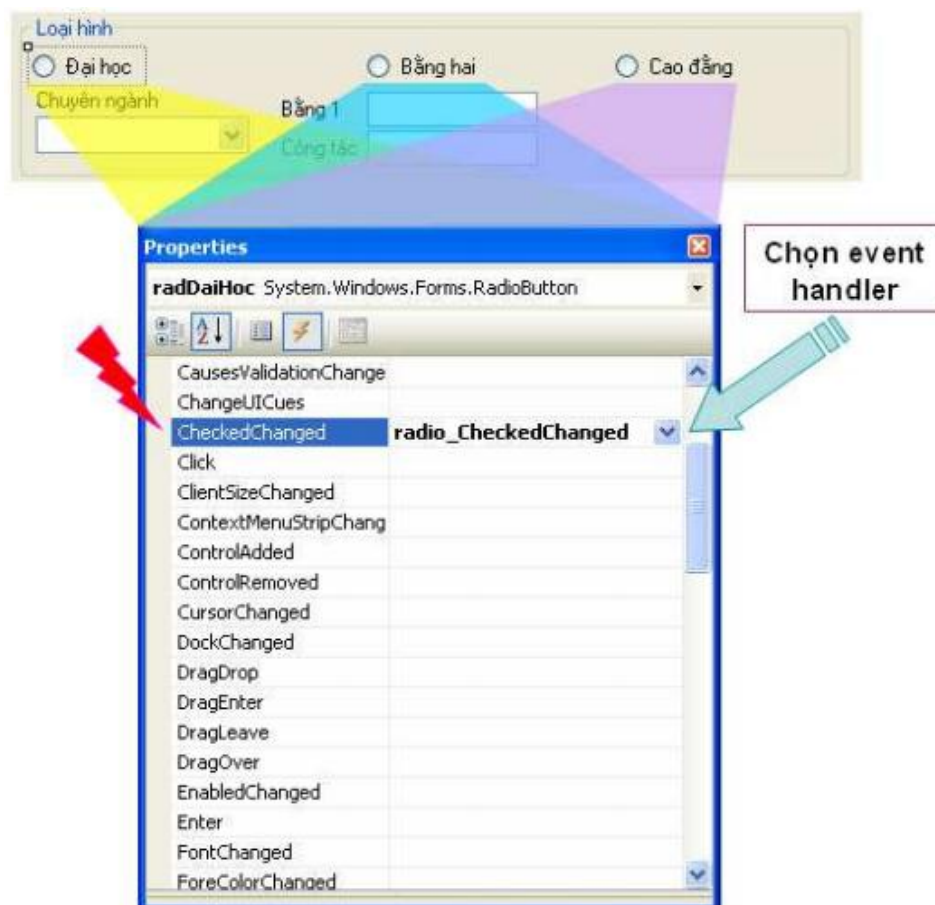
Bên dưới là hàm xử lý sự kiện CheckedChanged có tên là radio_CheckedChanged trong lớp Form1.


```
private void radio_CheckedChanged(object sender, EventArgs e)
{
    Control control = (Control)sender;
    cboCN.Enabled = false;
    txtBang1.Enabled = false;
    txtCty.Enabled = false;

    if (control.Name == "radDaiHoc")
        cboCN.Enabled = true;
    else if (control.Name == "radBang2")
    {
        txtBang1.Enabled = true;
        txtCty.Enabled = true;
    }
}
```

b. Bước 2: đăng ký xử lý sự kiện CheckedChanged cho 3 radio button trên form1, cách thực hiện:

- Chọn radiobutton
- Trong cửa sổ properties chọn Event
- Trong cửa sổ event chọn item CheckedChanged và nhập tên trình xử lý sự kiện là radio_CheckedChanged.



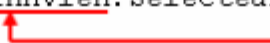
Sau khi hoàn tất các bước trên, ứng dụng của chúng ta cho phép user nhập vào các thông tin của sinh viên và khi user chọn button Thêm, một đối tượng sinh viên sẽ được tạo ra với các thông tin do user nhập vào, đối tượng này được lưu vào danh sách sinh viên StudentList, danh sách này sau đó được hiển thị trên listview thông qua phương thức UdateList của Form1.

11) Chức năng Xóa: khi user chọn 1 hoặc nhiều sinh viên trong danh sách vào kích button Xóa, thì chương trình sẽ thực hiện xóa các sinh viên được chọn đó ra khỏi danh sách.

Trong listview có một property là **SelectedItems**, chứa danh sách các items được chọn trong listview. SelectedItems có kiểu dữ liệu là **ListView.SelectedListViewItemCollection**. Dựa trên thông tin này chúng ta biết được các sinh viên được chọn, và lần lượt duyệt qua danh sách các sinh viên để xác định chính xác sinh viên bị xóa.

Chức năng Xóa được minh họa như sau, phương thức bên dưới chính là trình xử lý sự kiện của button “Xóa”.

```
private void btnRemove_Click(object sender, EventArgs e)
{
    /*xóa 1 item được chọn trong listview*/

    /*lấy danh sách các item được chọn*/
    ListView.SelectedListViewItemCollection SelectedList;
    SelectedList = lvwSinhVien.SelectedItems;
    
    // duyệt qua các item được chọn
    foreach (ListViewItem item in SelectedList)
    {
        // lấy mã số sinh viên trong item
        string ms = item.Text;
        // duyệt qua từng sinh viên trong ds
        foreach (SinhVien sv in StudentList)
        {
            // nếu mã số sinh viên trùng
            if (sv.MSSV.ToString() == ms)
            {
                StudentList.Remove(sv); // xóa đối tượng
                break; // thoát vòng lặp ds
            }
        }
    }
    //cập nhật lại danh sách sinh viên mới cho listview
    UpdateList();
}
```

12. Chức năng Sửa: cho phép user chọn một sinh viên trong danh sách và thực hiện thao tác chỉnh sửa lại các thông tin như: ngày sinh, địa chỉ, điện thoại, các thông tin khác không cho phép sửa.

Cách thực hiện:

- Xác định đối tượng sinh viên cần sửa gọi là sv.
- Hiển thị các thông tin của sv ra các control tương ứng
- Disable các control chứa thông tin không cho phép user sửa
- Sau khi user sửa xong sẽ xác nhận cập nhật lại thông tin cần sửa -> cập nhật lại đối tượng sv.