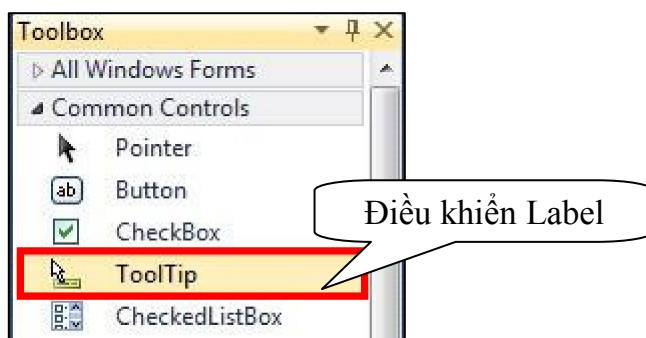


CHƯƠNG 4: CÁC ĐIỀU KHIỂN ĐẶC BIỆT

4.1. Điều khiển Tooltip, HelpProvider, ErrorProvider

4.1.1. Điều khiển Tooltip

Điều khiển *Tooltip* là điều khiển cho phép hiển thị các thông tin chú thích khi người dùng đưa chuột qua các điều khiển có thiết lập *Tooltip*. Điều khiển *Tooltip* được đặt trong nhóm Common Controls của cửa sổ Toolbox như hình 4.1.

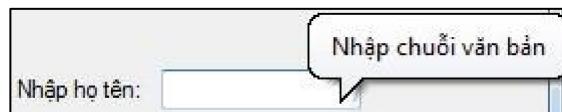


Hình 4.1: Điều khiển Tooltip

- Một số thuộc tính thường dùng của *Tooltip*:

Bảng 4.1: Bảng mô tả các thuộc tính của *Tooltip*

Thuộc tính	Mô tả
Active	Mang giá trị True hoặc False, nếu thiết lập True thì <i>Tooltip</i> có hiệu lực hiển thị thông báo, nếu mang giá trị False thì <i>Tooltip</i> không hiển thị được thông báo.
AutomaticDelay	Thiết lập thời gian xuất hiện <i>Tooltip</i> khi vừa đưa chuột đến điều khiển, thời gian tính bằng mili giây
AutoPopDelay	Thời gian hiển thị <i>Tooltip</i> cho đến khi kết thúc khi người dùng dừng dã đưa chuột đến điều khiển, thời gian tính bằng mili giây
IsBalloon	Quy định kiểu hiển thị của <i>Tooltip</i> . Nếu thiết lập False kiểu hiển thị <i>Tooltip</i> :

	 <p>Nếu thiết lập True kiểu hiển thị Tooltip:</p> 
<i>ReshowDelay</i>	Thời gian mà Tooltip tắt từ khi người dùng đưa chuột ra khỏi điều khiển, thời gian tính bằng mili giây
<i>ShowAlways</i>	
<i>ToolTipIcon</i>	Biểu tượng xuất hiện bên cạnh chuỗi khai báo trong thuộc tính <i>ToolTipTitle</i>
<i>ToolTipTitle</i>	Chuỗi hiện thị bên cạnh biểu tượng <i>ToolTipIcon</i>
<i>UseAnimation</i>	Thiết lập hiệu ứng ảnh động được biểu diễn khi Tooltip được hiển thị
<i>UseFading</i>	Thiết lập hiệu ứng mờ dần được biểu diễn khi Tooltip hiển thị

- Một số phương thức thường dùng của *Tooltip*:

Bảng 4.2: Bảng mô tả các phương thức của *Tooltip*

Phương thức	Mô tả
<i>SetTooltip()</i>	Thiết lập chuỗi hiển thị của Tooltip trên điều khiển
<i>GetTooltip()</i>	Lấy nội dung chuỗi hiển thị trên Tooltip
<i>Clear()</i>	Loại bỏ tất cả <i>ToolTipText</i> cho các điều khiển trên form

Ví dụ 4.1: Viết chương trình tạo giao diện form đăng nhập và thực hiện yêu cầu chức năng như hình 4.2.



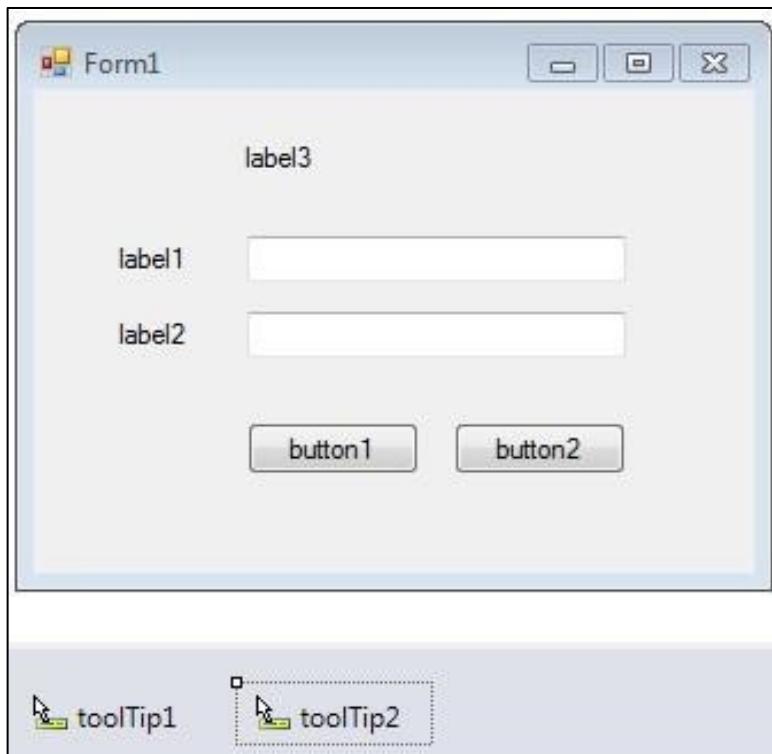
Hình 4.2: Giao diện form đăng nhập ví dụ 4.1

Yêu cầu:

- Khi rê chuột vào TextBox1: hiển thị dòng ghi chú “Nhập chuỗi ký không dấu, không khoảng trắng”
- Khi rê chuột vào TextBox2: hiển thị dòng ghi chú “Nhập ít nhất 6 ký tự, nhiều nhất 10 ký tự”
- Khi nhấn nút “Đăng nhập”: hiển thị MessageBox với nội dung “Bạn đăng nhập thành công”
- Khi nhấn nút “Thoát”: thoát khỏi chương trình

Hướng dẫn:

Bước 1: Thiết kế giao diện form ban đầu như hình 4.3. Sau đó kéo 2 *Tooltip* từ cửa sổ Toolbox thả vào Form1



Hình 4.3: Giao diện ban đầu form đăng nhập ví dụ 1

Bước 2: Thiết lập giá trị thuộc tính tính điều khiển trong cửa sổ Properties

- Form1:

Thuộc tính *Text*: “Đăng nhập”

- label3:

Thuộc tính *Text*: “Đăng nhập”

Thuộc tính *Size*: 16

- label1:

Thuộc tính *Text*: “Tên đăng nhập:”

- label2:

Thuộc tính *Text*: “Mật khẩu:”

- button1:

Thuộc tính *Name*: btnDangNhap

Thuộc tính *Text*: “Đăng nhập”

- button2:

Thuộc tính *Name*: btnThoat

Thuộc tính *Text*: “Thoát”

- toolTip1:

Thuộc tính *Name*: ttpDangNhap

- toolTip2:

Thuộc tính *Name*: ttpMatKhau

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của nút btnDangNhap:

```
private void btnDangNhap_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bạn đăng nhập thành công");
}
```

- Sự kiện *Click* của nút btnThoat:

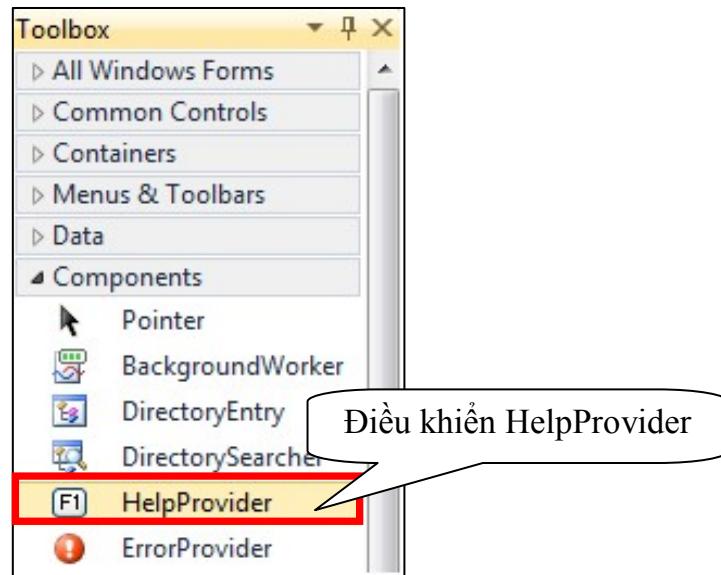
```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *Load* của Form1:

```
private void Form1_Load(object sender, EventArgs e)
{
    ttpDangNhap.SetToolTip(txtDangNhap,
        "Nhập chuỗi ký không dấu, không khoảng trắng");
    ttpMatKhau.SetToolTip(txtMatKhau,
        "Nhập ít nhất 6 ký tự, nhiều nhất 10 ký tự");
}
```

4.1.2. Điều khiển HelpProvider

Điều khiển *HelpProvider* cung cấp cửa sổ trợ giúp cho điều khiển. Với những ứng dụng có sử dụng *HelpProvider*, người dùng có thể gọi sự trợ giúp bằng cách ấn phím F1. Điều khiển *HelpProvider* được đặt trong nhóm Components của cửa sổ Toolbox như hình 4.4.



Hình 4.4: Điều khiển HelpProvider

- Một số thuộc tính thường dùng của *HelpProvider*:

Bảng 4.3: Bảng mô tả các thuộc tính của *HelpProvider*

Thuộc tính	Mô tả
<i>HelpNamespace</i>	Chỉ định tên tập trình trợ giúp định dạng chm hoặc html.

Điểm đặc biệt là khi thêm điều khiển *HelpProvider* vào form thì một số thuộc tính như: *HelpKeyword* on *helpProvider*, *HelpNavigator* on *helpProvider*, *HelpString* on *helpProvider* và *ShowHelp* on *helpProvider* sẽ xuất hiện trên tất cả các điều khiển có trên form.

Bảng 4.4: Bảng mô tả các thuộc tính của điều khiển khi thêm *HelpProvider*

Thuộc tính	Mô tả
<i>HelpKeyWord</i>	Từ khóa tìm kiếm, từ khóa này là chỉ mục hoặc chủ đề được truyền vào tập tin tìm kiếm. Thuộc tính <i>HelpNavigator</i> sẽ quy định từ khóa này tìm kiếm theo chủ đề hay theo chỉ mục.
<i>HelpNavigator</i>	Thiết lập cách thức hiển thị của tập tin trợ giúp. Gồm các thuộc tính thành viên như: <ul style="list-style-type: none"> - <i>AssociateIndex</i>: Mở tập tin trợ giúp và hiển thị danh sách chỉ mục có ký tự trùng với ký tự đầu tiên trong thuộc tính <i>HelpKeyWord</i>.

Ví dụ 4.2: tập tin trợ giúp là tập tin đuôi .chm và thuộc tính HelpKeyword chứa từ khóa “HLOOKUP”



- Find: Giúp hiển thị nội dung trợ giúp có chuỗi ký tự trùng với từ khóa trong thuộc tính HelpKeyword.
- Index: Hiển thị danh mục các chỉ mục trong tập tin trợ giúp.



- KeywordIndex: Hiển thi danh mục là các chỉ mục như từ khóa trong thuộc tính HelpKeyword

Ví dụ 4.3: Từ khóa HelpKeyword là HLOOKUP



- TableOfContents: Hiển thị tất cả các nội dung trong tập tin trợ giúp



	<ul style="list-style-type: none"> - Topic: Hiển thị danh mục các chủ đề trong tập tin trợ giúp, áp dụng với tập tin có hỗ trợ danh mục các chủ đề. - TopicId: Hiển thị chủ đề có mã trùng với mã chủ đề chỉ định, áp dụng với tập tin có hỗ trợ danh mục các chủ đề và các chủ đề được đánh số.
<i>HelpString</i>	Hiển thị chuỗi trợ giúp cho điều khiển. Nếu thuộc tính <i>HelpProvider</i> không được thiết lập thì khi người dùng nhấn phím F1 sẽ hiển thị chuỗi trợ giúp này
<i>ShowHelp</i>	Nếu thiết lập giá trị True thì cho phép nội dung trợ giúp hiển thị trên một điều khiển nào đó. Nếu thiết lập giá trị False thì không hiển thị được

➤ Một số phương thức thường dùng của *HelpProvider*:

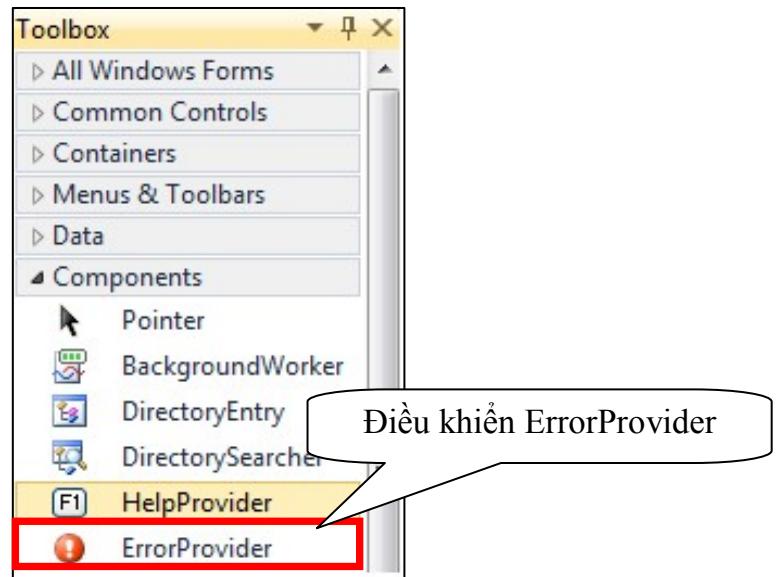
Các thuộc tính HelpKeyWord, HelpNavigator, HelpString, ShowHelp có thể được thiết lập giá trị trong cửa sổ Properties hoặc có thể sử dụng các phương thức như bảng 4.5 để thiết lập giá trị

Bảng 4.5: Bảng mô tả các phương thức của điều khiển khi thêm *HelpProvider*

Phương thức	Mô tả
<i>SetHelpKeyword</i>	Thiết lập giá trị cho thuộc tính HelpKeyword
<i>SetHelpNavigator</i>	Thiết lập giá trị cho thuộc tính HelpNavigator
<i>SetHelpString</i>	Thiết lập giá trị cho thuộc tính HelpString
<i>SetShowHelp</i>	Thiết lập giá trị cho thuộc tính ShowHelp

4.1.3. Điều khiển *ErrorProvider*

ErrorProvider giúp báo cho người dùng biết thông tin lỗi của điều khiển trên form. Thông thường khi điều khiển trên form lỗi, *ErrorProvider* sẽ cung cấp một biểu tượng  để thông báo lỗi bên cạnh điều khiển đó. Điều khiển *ErrorProvider* được đặt trong nhóm Components của cửa sổ Toolbox như hình 4.5.



Hình 4.5: Điều khiển ErrorProvider

- Một số thuộc tính thường dùng của *ErrorProvider*:

Bảng 4.6: Bảng mô tả các thuộc tính của *HelpProvider*

Thuộc tính	Mô tả
<i>Icon</i>	Chọn biểu tượng thể hiện lỗi của điều khiển
<i>BlinkRate</i>	Tốc độ nhấp nháy của biểu tượng trong thuộc tính <i>Icon</i> . Tốc độ tính theo mili giây
<i>BlinkStyle</i>	Kiểu nhấp nháy của biểu tượng. Nếu thiết lập giá trị <i>NeverBlink</i> thì biểu tượng sẽ hiển thị mà không nhấp nháy.

- Một số phương thức thường dùng của *ErrorProvider*:

Bảng 4.7: Bảng mô tả các phương thức của *HelpProvider*

Phương thức	Mô tả
<i>SetError(<Điều khiển>, <Thông báo lỗi>)</i>	Giúp hiển thị lỗi và thông báo lỗi của điều khiển. Thông báo lỗi hiển thị dưới dạng Tooltip
<i>Clear()</i>	Xóa biểu tượng <i>ErrorProvider</i> của điều khiển tương ứng trên form.
<i>GetError()</i>	Lấy chuỗi thông báo lỗi của điều khiển.

Ví dụ 4.4: Viết chương trình tạo form đăng nhập như hình 4.6. Yêu cầu ở Textbox nhập tên tài khoản không được có khoảng trắng; Textbox mật khẩu phải là ký tự số và không

được để trống; Hiển thị trợ giúp cho điều khiển Textbox tên tài khoản, cũ thẻ khi nhấn F1
sẽ hiện trợ giúp tạo mật khẩu từ website:

<http://phunutoday.vn/kham-pha-cong-nghe/cac-nguyen-tac-tao-mat-khau-an-toan-33828.html>



Hình 4.6: Giao diện formm đăng nhập ví dụ 4.4

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 4.7



Hình 4.7: Giao diện ban đầu form đăng nhập ví dụ 4.4

Bước 2: Thiết lập giá trị thuộc tính cho các điều khiển trong cửa sổ Properties

- Form1:
 - Thuộc tính Text: “Đăng nhập”
- label1:
 - Thuộc tính Text: “Đăng nhập
 - Thuộc tính Size: 14
- label2:
 - Thuộc tính Text: “Tên:”
- label3:
 - Thuộc tính Text: “Mật khẩu:”
- button1:
 - Thuộc tính Text: “Đăng nhập”
 - Thuộc tính Name: btnDangNhap
- button2:
 - Thuộc tính Text: “Thoát”
 - Thuộc tính Name: btnThoat
- TextBox1:
 - Thuộc tính Name: txtTen
- TextBox2:
 - Thuộc tính Name: txtMatKhau
 - Thuộc tính PasswordChar: *
- errorProvider1:
 - Thuộc tính Name: LoiTenDN
- errorProvider2:
 - Thuộc tính Name: LoiMatKhau
- helpProvider1:
 - Thuộc tính Name: TroGiupMatKhau

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của nút btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *Load* của Form1:

```
private void Form1_Load(object sender, EventArgs e)
{
    //TabIndex giúp thiết lập thứ tự điều khiển khi ấn phím
    //tab
    txtTen.TabIndex = 0;
    txtMatKhau.TabIndex = 1;
    btnDangNhap.TabIndex = 2;
    btnThoat.TabIndex = 3;
    TroGiupMatKhau.SetShowHelp(txtTen, true);
    TroGiupMatKhau.HelpNamespace =
        "http://phunutoday.vn/kham-pha-cong-nghe/cac-nguyen-tac-
        tao-mat-khau-an-toan-33828.html";
}
```

- Sự kiện *TextChanged* của TextBox txtTen:

```
private void txtTen_TextChanged(object sender, EventArgs e)
{
    if (txtTen.Text.IndexOf(' ') != -1)
        LoiTENDN.SetError(txtTen, "Nhập tên không được có
        khoảng trắng");
    else
        LoiTENDN.Clear();
}
```

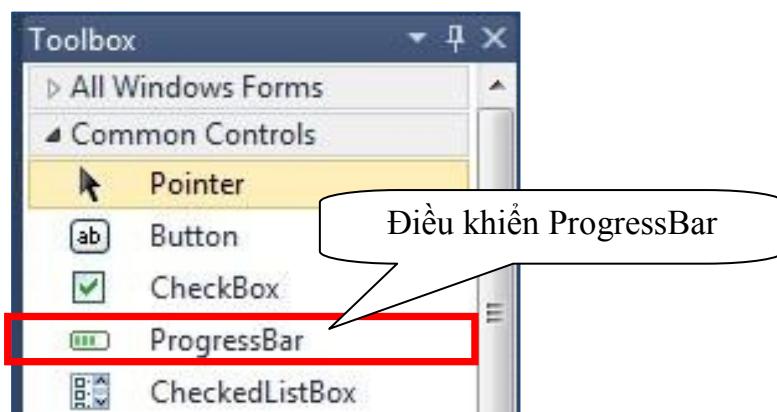
- Sự kiện *TextChanged* của TextBox txtMatKhau:

```
private void txtMatKhau_TextChanged(object sender, EventArgs e)
{
    long so = 0;
    try
    {
        so = Convert.ToInt64(txtMatKhau.Text);
        LoiMatKhau.Clear();
    }
    catch (Exception ex)
    {
        LoiMatKhau.SetError(txtMatKhau, "Phải nhập ký tự số
        và không được để trống");
    }
}
```

4.2. Điều khiển ProgressBar và Timer

4.2.1. Điều khiển ProgressBar

ProgressBar sử dụng để hiển thị thời gian thực hiện của một công việc nào đó. *ProgressBar* được đặt trong nhóm Common Controls của cửa sổ Toolbox như hình 4.8



Hình 4.8: Điều khiển ProgressBar

- Một số thuộc tính thường dùng của *ProgressBar*:

Bảng 4.8: Bảng mô tả các thuộc tính của *ProgressBar*

Thuộc tính	Mô tả
<i>Maximum</i>	Giá trị tối đa của <i>ProgressBar</i> . Khi <i>ProgressBar</i> được lấp đầy nghĩa là <i>ProgressBar</i> đã đạt giá trị <i>Maximum</i> .
<i>Minimum</i>	Giá trị nhỏ nhất của <i>ProgressBar</i> . Khi <i>ProgressBar</i> trống rỗng nghĩa là <i>ProgressBar</i> đang có giá trị <i>Minimum</i> .
<i>Value</i>	Giữ giá trị hiện tại của <i>ProgressBar</i> , giá trị này nằm trong đoạn <i>Minimum</i> và <i>Maximum</i> .
<i>Style</i>	Kiểu hiển thị của <i>ProgressBar</i> .
<i>Step</i>	Lượng giá trị thêm vào <i>Value</i> khi phương thức <i>PerformStep()</i> được gọi.

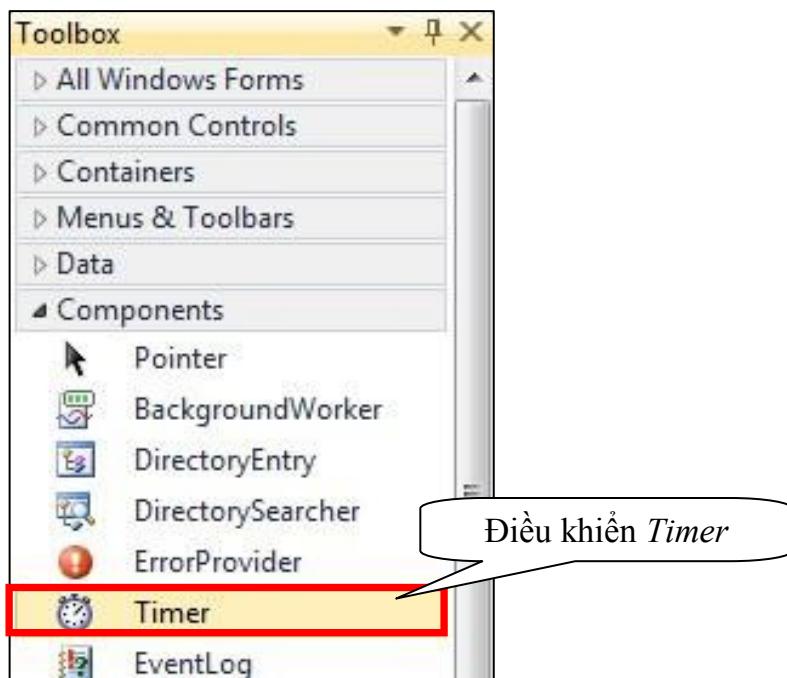
- Một số phương thức thường dùng của *ProgressBar*:

Bảng 4.9: Bảng mô tả các phương thức của ProgressBar

Phương thức	Mô tả
<i>PerformStep()</i>	Phương thức giúp tăng <i>ProgressBar</i> . Giá trị tăng là giá trị được thiết lập trong thuộc tính <i>Step</i> .
<i>Increment(<giá trị>)</i>	Phương thức giúp tăng <i>ProgressBar</i> . Giá trị tăng là tham số đầu vào <giá trị> của phương thức.

4.2.2. Điều khiển Timer

Điều khiển *Timer* cho phép thực thi lại một hành động sau một khoảng thời gian xác định. *Timer* được đặt trong nhóm Components của cửa sổ Toolbox như hình 4.9



Hình 4.9: Điều khiển Timer

- Một số thuộc tính thường dùng của *Timer*:

Bảng 4.10: Bảng mô tả các thuộc tính của Timer

Thuộc tính	Mô tả
<i>Interval</i>	Thiết lập giá trị là một số nguyên. Giá trị nguyên này là thời lượng của một chu kỳ (tính

	bằng đơn vị mili giây).
<i>Enable</i>	Thiết lập giá trị <i>True</i> hoặc <i>False</i> . Nếu là giá trị <i>True</i> thì điều khiển <i>Timer</i> hoạt động, nếu là <i>False</i> thì điều khiển <i>Timer</i> không hoạt động.

- Một số phương thức thường dùng của *Timer*:

Bảng 4.11: Bảng mô tả các phương thức của *Timer*

Phương thức	Mô tả
<i>Start()</i>	Kích hoạt điều khiển <i>Timer</i> hoạt động. Phương thức này tương ứng với việc thiết lập giá thuộc tính <i>Enable</i> là <i>True</i>
<i>Stop()</i>	Dừng hoạt động của điều khiển <i>Timer</i> . Phương thức này tương ứng với việc thiết lập giá thuộc tính <i>Enable</i> là <i>False</i> .

- Một số sự kiện thường dùng của *Timer*:

Bảng 4.12: Bảng mô tả các sự kiện của *Timer*

Sự kiện	Mô tả
<i>Tick</i>	Sự kiện được gọi trong mỗi chu kỳ Interval

Ví dụ 4.5: Viết chương trình hỗ trợ người dùng học giải phương trình bậc nhất: $ax + b = 0$. Thiết kế giao diện form như hình 4.10.

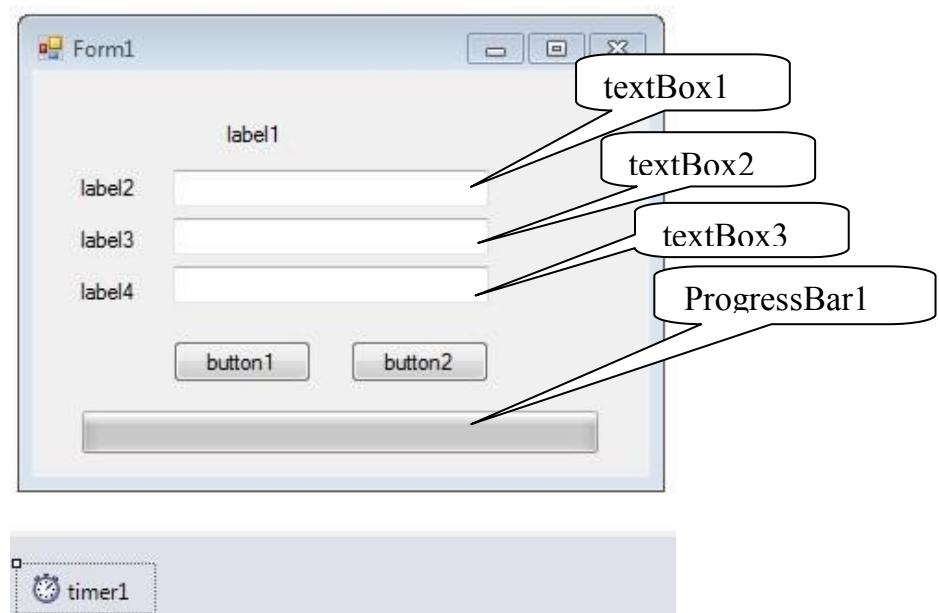


Hình 4.10: Giao diện form giải phương trình bậc nhất

Yêu cầu: Phát sinh ngẫu nhiên hệ số a và b của phương trình. Sau đó người dùng nhập kết quả và ấn nút trả lời. Nếu trả lời đúng thì hiện MessageBox với nội dung “Bạn đã làm đúng”, nếu trả lời sai thì hiển MessageBox với nội dung “Bạn đã trả lời sai”. Lưu ý: Thời gian để hoàn thành giải phương trình là 30 giây hiển thị tương ứng với ProgressBar, trong khoảng thời gian hết 30 giây người dùng không giải được sẽ hiển thị MessageBox với nội dung “Hết giờ làm bài”.

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 4.11



Hình 4.11: Giao diện ban đầu form giải phương trình bậc nhất

Bước 2: Thiết lập giá trị thuộc tính trong cửa sổ Properties cho điều khiển

- label1:

Thuộc tính Text: “Phương trình bậc nhất: $ax + b = 0$ ”

Thuộc tính Size: 14

- label2:

Thuộc tính Text: “Hệ số a:”

- label3:

Thuộc tính Text: “Hệ số b:”

- label4:

Thuộc tính Text: “Nhập nghiệm:”

- textBox1:

- Thuộc tính Name: txtA
Thuộc tính Enable: False
- textBox2:
Thuộc tính Name: txtB
Thuộc tính Enable: False
 - textBox3:
Thuộc tính Name: txtX
 - button1:
Thuộc tính Name: btnKiemTra
Thuộc tính Text: “Kiểm tra”
 - button2:
Thuộc tính Name: btnThoat
Thuộc tính Text: Thoát
 - ProgressBar1:
Thuộc tính Name: ProGressTG
Thuộc tính Minimum: 0
Thuộc tính Maximum: 30000
Thuộc tính Step: 1000
Thuộc tính Style: Blocks
 - timer1:
Thuộc tính Name: ThoiGian
Thuộc tính Enable: True
Thuộc tính Interval: 1000

Bước 3: Viết mã lệnh cho các điều khiển

- Khai báo các biến

```
private int a=0;  
private int b=0;  
private float x = 0;  
Random rd = new Random();
```

- Sự kiện *Click* của nút btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)  
{  
    Close();  
}
```

- Sự kiện *Load* của Form1:

```
private void Form1_Load(object sender, EventArgs e)
{
    a = rd.Next(-10, 10);
    txtA.Text = a.ToString();
    b = rd.Next(-10, 10);
    txtB.Text = b.ToString();
    x = -b / (float)a;
}
```

- Sự kiện *Tick* của *Timer ThoiGian*:

```
private void ThoiGian_Tick(object sender, EventArgs e)
{
    if (ProGressTG.Value == 30000)
    {
        ThoiGian.Enabled = false;
        MessageBox.Show("Hết giờ làm bài");
    }
    ProGressTG.PerformStep();
}
```

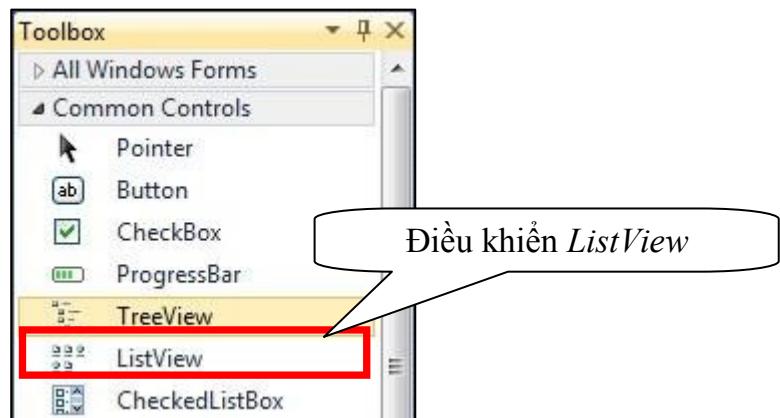
- Sự kiện *Click* của nút *btnKiemTra*:

```
private void btnKiemtra_Click(object sender, EventArgs e)
{
    float kq = float.Parse(txtX.Text);
    if (Math.Abs(kq - x) < 0.01)
    {
        MessageBox.Show("Bạn đã làm đúng");
        Close();
    }
    else
        MessageBox.Show("Bạn đã trả lời sai");
}
```

4.3. Điều khiển ListView

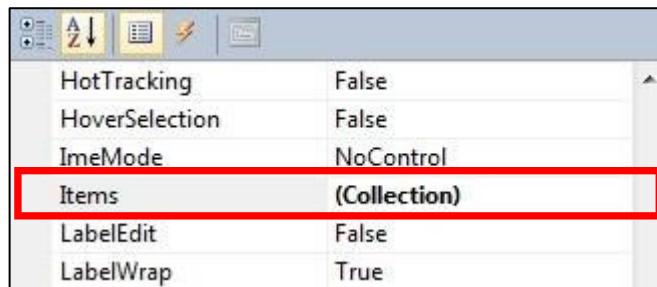
ListView là điều khiển cho phép hiển thị danh sách các đối tượng. Mỗi đối tượng hiển thị trong ListView được gọi là Item. Item là đối tượng được tạo từ lớp ListViewItem. Mỗi Item có thuộc tính Text là chuỗi ký tự hiển thị ở cột đầu tiên trong ListView, mỗi

Item có các SubItem hiển thị ở các cột tiếp theo trong ListView. Điều khiển ListView đặt trong Common Controls của cửa sổ Toolbox như hình 4.12.



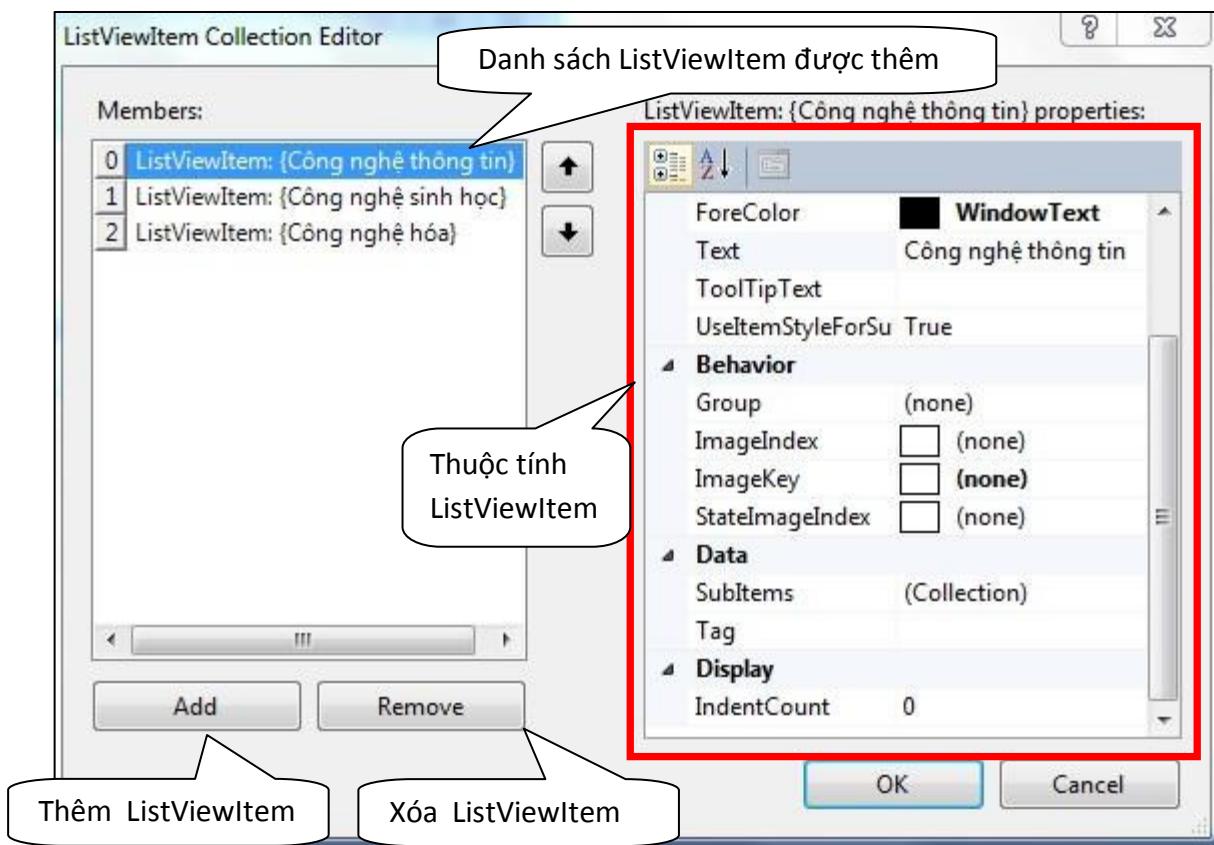
Hình 4.12: Điều khiển ListView

Lập trình viên có thể thêm ListViewItem vào ListView bằng cách chọn thuộc tính Items trong cửa sổ Properties của ListView như hình 4.13.



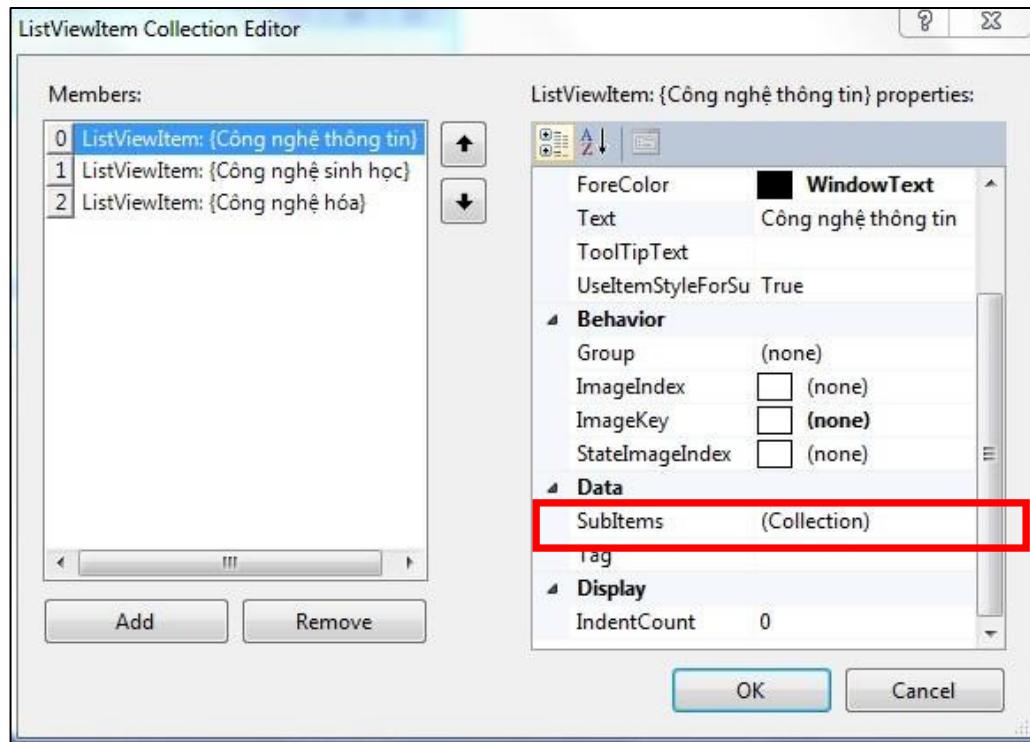
Hình 4.13: Thuộc tính Items trong cửa sổ Properties

Sau khi chọn thuộc tính Items trong cửa sổ Properties, cửa sổ ListViewItem Collection Editor sẽ hiển thị như hình 4.14. Lập trình viên có thể thêm hoặc xóa ListViewItem trong ListView bằng cách nhấn nút Add hoặc Remove.



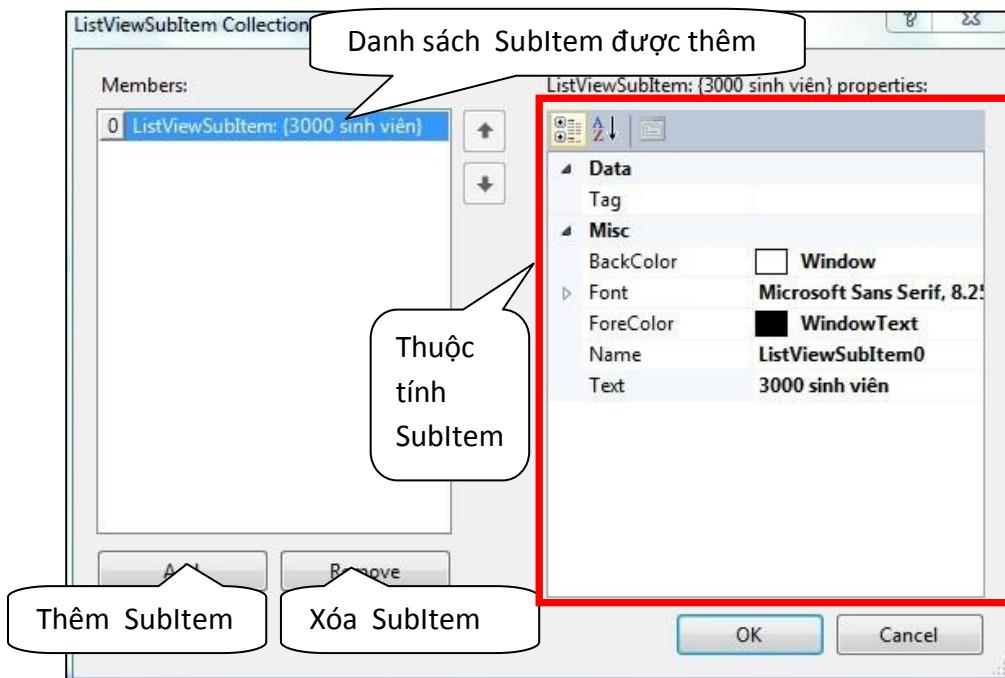
Hình 4.14: Cửa sổ ListViewItem Collection Editor

Thêm các SubItem của Item trong ListViewItem bằng cách chọn thuộc tính SubItems trong cửa sổ *ListViewItem Collection Editor* như hình 4.15.



Hình 4.15: Thuộc tính SubItems trong cửa sổ ListView Item Collection Editor

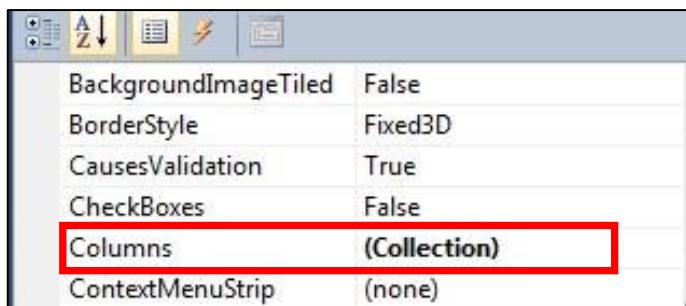
Sau khi chọn thuộc tính SubItems trong cửa sổ *ListViewItem Collection Editor*, cửa sổ *ListViewSubItem Collection Editor* như hình 4.16 được hiển thị cho phép lập trình viên thêm hoặc xóa các SubItem.



Hình 4.16: Cửa sổ ListViewSubItem Collection Editor

- Một số thuộc tính thường dùng của *ListView*:

Bảng 4.13: Bảng mô tả các thuộc tính của *ListView*

Thuộc tính	Mô tả												
<i>View</i>	<p>Thuộc tính <i>View</i> qui định cách hiển thị các <i>Item</i> trong <i>ListView</i>. Thuộc tính <i>View</i> có 5 giá trị:</p> <ul style="list-style-type: none"> - <i>Detail</i>: Một Icon (Icon lấy từ <i>ImageList</i>) và <i>Text</i> được hiển thị ở cột đầu tiên. Tiếp theo là các <i>SubItem</i> được hiển thị ở các cột tiếp theo. Tuy nhiên để hiển thị Item dạng <i>Detail</i> thì tạo thêm <i>Column Header</i> cho <i>ListView</i>:  <p><i>ListView</i> hiển thị Item dạng <i>Detail</i> như sau:</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Title</th> </tr> </thead> <tbody> <tr> <td>Co-Gai-Xuan-Thi.mp3</td> <td>Co Gai Xuan Thi</td> </tr> <tr> <td>Diep-Khuc-Mua-Xuan.mp3</td> <td>diep Khuc Mua Xuan</td> </tr> <tr> <td>Giot-Mua-Xuan.mp3</td> <td>Giot Mua Xuan</td> </tr> <tr> <td>Khuc-Xuan.mp3</td> <td>Khuc Xuan</td> </tr> <tr> <td>Kia-Xuan-Da-Ve.mp3</td> <td>Kia Xuan da Ve</td> </tr> </tbody> </table> <ul style="list-style-type: none"> - <i>LargeIcons</i>: Một biểu tượng lớn biểu diễn cho mỗi <i>Item</i> cùng với một nhãn ngay dưới icon. Các biểu tượng lớn này được lấy từ điều khiển <i>ImageList</i>, và được thiết lập trong thuộc tính <i>LargeImageList</i> của <i>ListView</i>. <p><i>ListView</i> hiển thị Item dạng <i>LargeIcons</i> như sau:</p>	Name	Title	Co-Gai-Xuan-Thi.mp3	Co Gai Xuan Thi	Diep-Khuc-Mua-Xuan.mp3	diep Khuc Mua Xuan	Giot-Mua-Xuan.mp3	Giot Mua Xuan	Khuc-Xuan.mp3	Khuc Xuan	Kia-Xuan-Da-Ve.mp3	Kia Xuan da Ve
Name	Title												
Co-Gai-Xuan-Thi.mp3	Co Gai Xuan Thi												
Diep-Khuc-Mua-Xuan.mp3	diep Khuc Mua Xuan												
Giot-Mua-Xuan.mp3	Giot Mua Xuan												
Khuc-Xuan.mp3	Khuc Xuan												
Kia-Xuan-Da-Ve.mp3	Kia Xuan da Ve												



- *List*: Mỗi Item sẽ được hiển thị như một biểu tượng nhỏ với một nhãn ở bên phải. Các biểu tượng trong *ListView* được sắp xếp theo các cột.

ListView hiển thị Item dạng *List* như sau:



- *SmallIcons*: Mỗi Item nằm trong một cột gồm có biểu tượng nhỏ cùng với nhãn. Các biểu tượng lớn này được lấy từ điều khiển *ImageList*, và được thiết lập trong thuộc tính *SmallImageList* của *ListView*.

ListView hiển thị Item dạng *SmallIcons* như sau:



- *Tiles*: Mỗi một Item sẽ hiển thị với biểu tượng có kích thước là tối đa cùng với một *label* và các *subitem* sẽ hiển thị các cột bên phải.

ListView hiển thị Item dạng *Tiles* như sau:

	 <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>Chi La Giac Mo - Vu Cat Tuong.mp3 MP3 Format Sound</td><td>có khi nao roi xa - Vu Cat Tuong.mp3 MP3 Format Sound</td></tr> <tr> <td>Danh Cho Em - Hoang Ton.mp3 MP3 Format Sound 3.64 MB</td><td>Dau Mua - Trung Quan Idol.mp3 MP3 Format Sound 3.97 MB</td></tr> </table>	Chi La Giac Mo - Vu Cat Tuong.mp3 MP3 Format Sound	có khi nao roi xa - Vu Cat Tuong.mp3 MP3 Format Sound	Danh Cho Em - Hoang Ton.mp3 MP3 Format Sound 3.64 MB	Dau Mua - Trung Quan Idol.mp3 MP3 Format Sound 3.97 MB
Chi La Giac Mo - Vu Cat Tuong.mp3 MP3 Format Sound	có khi nao roi xa - Vu Cat Tuong.mp3 MP3 Format Sound				
Danh Cho Em - Hoang Ton.mp3 MP3 Format Sound 3.64 MB	Dau Mua - Trung Quan Idol.mp3 MP3 Format Sound 3.97 MB				
<i>Items</i>	<p>Trả về các <i>Item</i> chứa trong <i>ListView</i>. Một số phương thức và thuộc tính thường dùng của <i>ListView.Items</i>:</p> <ul style="list-style-type: none"> - <i>Count</i>: Đếm số lượng Item trong <i>ListView</i> - <i>Insert(i, <Item mới>)</i>: Chèn thêm Item < Item mới> vào vị trí <i>i</i> trong <i>ListView</i> - <i>Add(<Item mới>)</i>: thêm Item <Item mới> vào cuối <i>ListView</i> - <i>Remove(<Item cần xóa>)</i>: Xóa Item <Item cần xóa> khỏi <i>ListView</i> - <i>RemoveAt(i)</i>: Xóa Item có chỉ số <i>i</i> khỏi <i>ListView</i> - <i>Contains(<Item cần tìm>)</i>: Trả về <i>True</i> nếu tìm thấy <Item cần tìm> trong <i>ListView</i>, trả về <i>False</i> nếu không có trong <i>ListView</i> - <i>IndexOf(<Item cần tìm>)</i>: Nếu <Item cần tìm> có trong <i>ListView</i> thì trả về chỉ số của Item tìm thấy trong <i>ListView</i>, nếu không tìm thấy sẽ trả về -1 				
<i>MultiSelect</i>	True/ False: Cho phép hoặc không cho phép chọn một lúc nhiều Item trong <i>ListView</i>				
<i>FullRowSelect</i>	Khi chọn dòng dữ liệu highlighted cả dòng hay chỉ ô được chọn				
<i>GridLines</i>	Nếu thiết lập True sẽ hiển thị các dòng và cột dạng lưới, thiết lập False không hiển thị dạng lưới				
<i>SelectedItems</i>	Trả về tập các Items được chọn trong <i>ListView</i>				
<i>LargeImageIcon</i>	Gán đối tượng <i>ImageList</i> cho <i>ListView</i>				
<i>SmallImageIcon</i>	Gán đối tượng <i>ImageList</i> cho <i>ListView</i>				
<i>FocusedItem.Index</i>	Trả về chỉ số dòng được chọn trong <i>ListView</i>				
<i>SelectedIndices.Count</i>	Trả về số lượng Item được chọn trong <i>ListView</i>				
<i>SelectedIndices</i>	Trả về danh sách chỉ mục các Item được chọn.				

	Ví dụ 4.6: myListView.SeletedIndices[0]: trả về chỉ mục của Item đầu tiên được chọn trong danh sách các Item được chọn trong <i>ListView</i>
--	---

- Một số phương thức thường dùng của *ListView*:

Bảng 4.14: Bảng mô tả các phương thức của *ListView*

Phương thức	Mô tả
<i>Clear()</i>	Xóa tất cả các <i>Item</i> và <i>Column</i> trong <i>ListView</i>
<i>Sort()</i>	Sắp xếp các <i>Item</i> trong <i>ListView</i>
<i>GetItemAt(x,y)</i>	Lấy <i>Item</i> tại vị trí toan độ x và y (x và y có thể lấy được thông qua sự kiện <i>Click</i> chuột)

- Một số sự kiện thường dùng của *ListView*:

Bảng 4.15: Bảng mô tả các sự kiện của *ListView*

Sự kiện	Mô tả
<i>SelectedIndexChanged</i>	Sự kiện phát sinh khi có sự thay đổi về chỉ mục được chọn của <i>Item</i> trên <i>ListView</i>
<i>ItemSelectionChanged</i>	Sự kiện phát sinh khi có sự thay đổi lựa chọn một <i>Item</i> trên <i>ListView</i>
<i>ItemCheck</i>	Xảy ra khi trạng thái chọn của <i>Item</i> thay đổi
<i>ColumnClick</i>	Sự kiện phát sinh khi một <i>column</i> trong <i>ListView</i> được <i>click</i>
<i>MouseClick</i>	Sự kiện phát sinh khi nhấp chuột chọn một <i>Item</i> trong <i>ListVlew</i>

Ví dụ 4.7: Thiết kế giao diện form như hình 4.17:

Hình 4.17: Giao diện form thêm lớp

Yêu cầu: Khi người dùng nhập xong tên lớp và số lượng sinh viên, sau đó nhấn nút “Thêm” thì trong ListView sẽ chèn một dòng vào cuối với tên lớp và số lượng vừa nhập.
Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 4.18

Hình 4.18: Giao diện ban đầu form thêm lớp

Bước 2: Thiết lập giá trị thuộc tính cho các điều khiển

- label1:

Thuộc tính Text: “Thêm lớp học”

Thuộc tính Size: 14

- label2:

Thuộc tính Text: “Tên lớp:”

- label3:

Thuộc tính Text: “Số lượng sinh viên:”

- button1:

Thuộc tính Text: “Thêm”

Thuộc tính Name: btnThem

- listView1:

Thuộc tính Name: myListview

Thuộc tính View: Details

Thuộc tính GridLines: True

- textBox1:

Thuộc tính Name: txtTenLop

- textBox2:

Thuộc tính Name: txtSoLuong

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của nút btnThem:

```
private void btnThem_Click(object sender, EventArgs e)
{
    if (txtSoLuong.Text != "" && txtTenLop.Text != "")
    {
        ListViewItem LVIItem = new
            ListViewItem(txtTenLop.Text);
        ListViewItem.ListViewSubItem LVSItem = new
            ListViewItem.ListViewSubItem(LVIItem, txtSoLuong.Text);
        LVIItem.SubItems.Add(LVSItem);
        myListview.Items.Add(LVIItem);
    }
}
```

- Sự kiện *Load* của Form1:

```
private void Form1_Load(object sender, EventArgs e)
{
    myListview.Columns.Add("Tên lớp", 160);
    myListview.Columns.Add("Số lượng sinh viên", 180);
}
```

4.4. Điều khiển TreeView

TreeView là điều khiển dùng để hiển thị danh sách các đối tượng dưới dạng phân cấp như hình 4.19. Đối tượng trong TreeView thường được gọi là node và cấu trúc phân cấp của TreeView được biểu diễn bởi lớp TreeNode. Mỗi một node trong TreeView có thể chứa các node khác. Node chứa một node khác gọi là node cha (RootNode) và node được chứa gọi là node con (ChildNode). Việc sử dụng điều khiển TreeView để hiển thị rất hữu ích, vì trình bày theo dạng phân cấp giúp việc hiển thị được rõ ràng và có hệ thống hơn. Điều khiển TreeView đặt trong nhóm Common Controls của cửa sổ Toolbox như hình 4.20.



Hình 4.19: Biểu diễn thư mục dạng phân cấp



Hình 4.20: Điều khiển TreeView

- Một số thuộc tính thường dùng của *TreeView*:

Bảng 4.16: Bảng mô tả các thuộc tính của *TreeView*

Thuộc tính	Mô tả
<i>Node</i>	Trả về một đối tượng thuộc lớp <i>TreeNode</i>
<i>SelectedNode</i>	Trả về node đang được chọn trong <i>TreeView</i>
<i>ShowPlusMinus</i>	Hiển thị dấu + và - trước mỗi <i>TreeNode</i>
<i>ShowRootLines</i>	Hiển thị đường thẳng nối giữa các <i>Root Node</i> trong một <i>TreeView</i>
<i>ImageList</i>	Hiển thị hình trước mỗi node trong <i>TreeView</i> . Lưu ý: Phải sử dụng thêm điều khiển <i>ImageList</i> , và gán tên đối tượng của điều khiển <i>ImageList</i> cho thuộc tính <i>ImageList</i> của <i>TreeView</i>
<i>ImageIndex</i>	Giá trị của thuộc tính <i>ImageIndex</i> là chỉ số của hình trong điều khiển <i>ImageList</i> . Khi gán chỉ số cho thuộc tính <i>ImageIndex</i> thì hình hiển thị trước mỗi node sẽ là hình có chỉ số tương ứng. Lưu ý: Phải sử dụng thuộc tính <i>ImageList</i> trước
<i>SelectedImageIndex</i>	Giá trị của thuộc tính <i>SelectImageIndex</i> là chỉ số của hình trong điều khiển <i>ImageList</i> . Khi người dùng chọn node nào thì node đó sẽ có hình tương ứng như thuộc tính <i>SelectedImageIndex</i> chỉ định

- Một số phương thức thường dùng của *TreeView*:

*Bảng 4.17: Bảng mô tả các phương thức của *TreeView**

Phương thức	Mô tả
<i>GetNodeCount()</i>	Đếm số node trong một <i>TreeView</i>
<i>ExpandAll()</i>	Hiển thị tất cả các node trên <i>TreeView</i>
<i>CollapseAll()</i>	Thu gọn tất cả các node trên <i>TreeView</i>
<i>GetNodeAt(x,y)</i>	Lấy một node tại một vị trí có tọa độ (x, y) trên màn hình. Lưu ý: Thường sử dụng sự kiện <i>MouseDown</i> hoặc <i>NodeMouseClick</i>

- Một số sự kiện thường dùng của *TreeView*:

*Bảng 4.18: Bảng mô tả các sự kiện của *TreeView**

Sự kiện	Mô tả
<i>AfterCollapse</i>	Phát sinh khi thu gọn một <i>TreeNode</i>
<i>AfterExpand</i>	Phát sinh khi hiển thị các node trong <i>TreeNode</i>
<i>AfterSelect</i>	Phát sinh khi chọn một <i>TreeNode</i>
<i>NodeMouseClick</i>	Phát sinh khi chọn một node

TreeView là điều khiển để hiển thị các node, tuy nhiên việc hiển thị này thực chất là do *TreeNode* tạo ra. Do đó để làm việc với các node cần sử dụng các thuộc tính và phương thức của lớp *TreeNode*.

- Một số thuộc tính thường dùng của *TreeNode*:

*Bảng 4.19: Bảng mô tả các thuộc tính của *TreeNode**

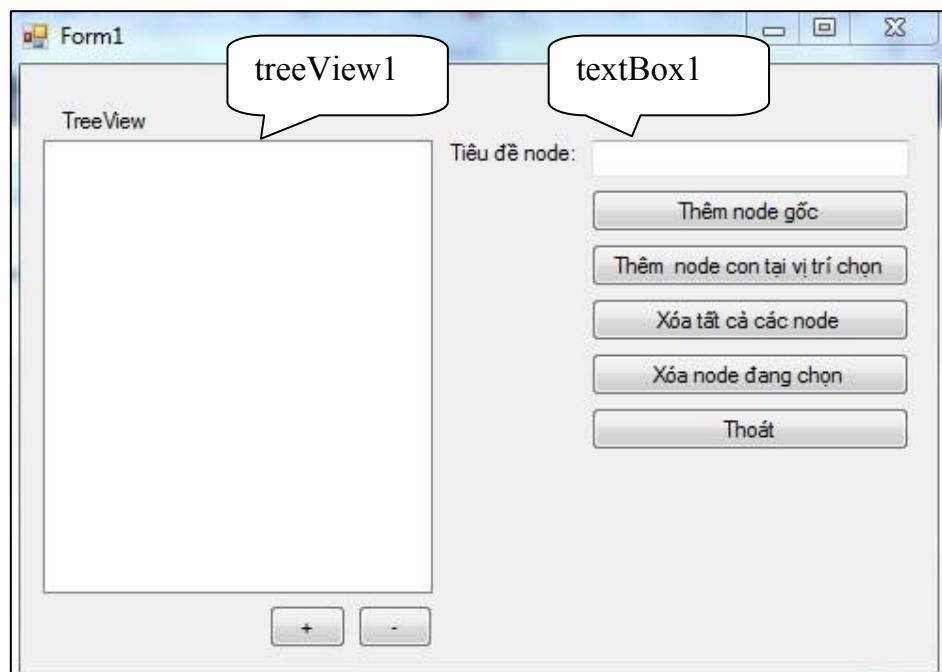
Thuộc tính	Mô tả
<i>Nodes</i>	Trả về tập các node
<i>Text</i>	Đọc/ gán chuỗi ký tự người dùng sẽ nhìn thấy ở mỗi node
<i>FirstNode</i>	Trả về node đầu tiên
<i>LastNode</i>	Trả về node cuối cùng
<i>NextNode</i>	Chuyển đến node tiếp theo
<i>PrevNode</i>	Lùi lại node trước đó
<i>Parent</i>	Trả về node cha của node hiện tại
<i>Index</i>	Trả về chỉ số của node

- Một số phương thức thường dùng của *TreeNode*:

Bảng 4.20: Bảng mô tả các phương thức của *TreeNode*

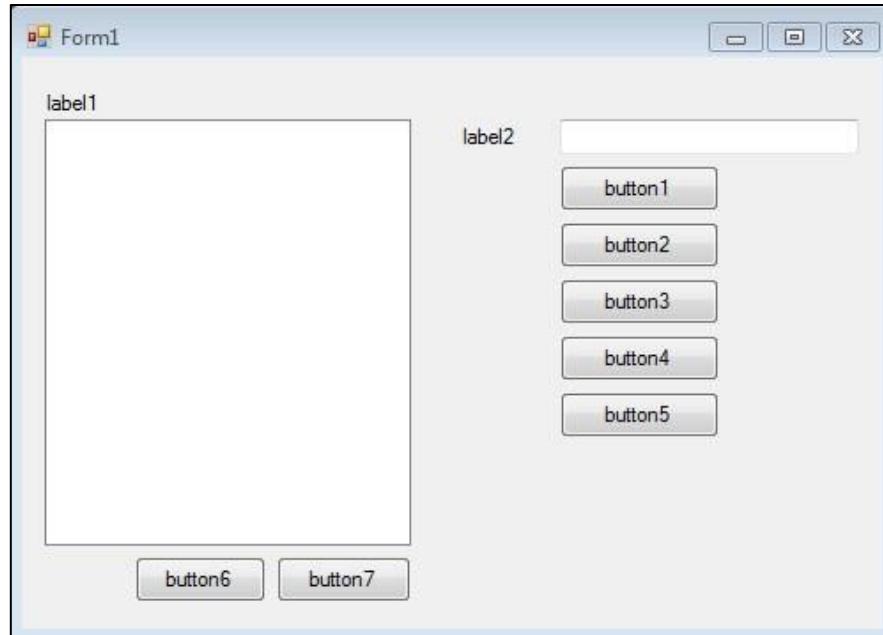
Phương thức	Mô tả
<i>Nodes.Add</i>	Thêm một node
<i>Nodes.Remove</i>	Xóa một node
<i>Nodes.Insert</i>	Chèn vào một node (chèn trước, chèn sau một node)
<i>Nodes.Clear</i>	Xóa tất cả các node con và node hiện tại

Ví dụ 4.8: Viết chương trình minh họa việc thêm sửa xóa các node trong một *TreeView*. Giao diện thiết kế như hình 4.21.



Hình 4.21: Giao diện form minh họa sử dụng *TreeView* ví dụ 4.8

Bước 1: Thiết kế giao diện ban đầu như hình 4.22



Hình 4.22: Giao diện ban đầu ví dụ 4.8

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties

- label1:
Thuộc tính Text: “TreeView”
- label2:
Thuộc tính Text: “Tiêu đề node:”
- treeView1:
Thuộc tính Name: TV_Test
- textBox1:
Thuộc tính Name: txtTieuDe
- button1:
Thuộc tính Text: “Thêm node gốc”
Thuộc tính Name: btnThemGoc
- button2:
Thuộc tính Text: “Thêm node con tại vị trí”
Thuộc tính Name: btnThemCon
- button3:
Thuộc tính Text: “Xóa tất cả các node”
Thuộc tính Name: btnXoaTatCa
- button4:

Thuộc tính Text: “Xóa node đang chọn”

Thuộc tính Name: btnXoaChon

- button5:

Thuộc tính Text: “Thoát”

Thuộc tính Name: btnThoat

- button6:

Thuộc tính Text: “+”

Thuộc tính Name: btnMoRong

- button7:

Thuộc tính Text: “-”

Thuộc tính Name: btnThuNho

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của nút btnThemGoc:

```
private void btnThemGoc_Click(object sender, EventArgs e)
{
    TV_Test.Nodes.Add(txtTieuDe.Text);
    txtTieuDe.Text = "";
}
```

- Sự kiện *Click* của nút btnThemCon:

```
private void btnThemCon_Click(object sender, EventArgs e)
{
    TV_Test.SelectedNode.Nodes.Add(txtTieuDe.Text);
    txtTieuDe.Text = "";
    TV_Test.ExpandAll();
}
```

- Sự kiện *Click* của nút btnXoaTaCa:

```
private void btnXoaTaCa_Click(object sender, EventArgs e)
{
    TV_Test.Nodes.Clear();
}
```

- Sự kiện *Click* của nút btnXoaChon:

```
private void btnXoaChon_Click(object sender, EventArgs e)
{
    TV_Test.SelectedNode.Remove();
}
```

- Sự kiện *Click* của nút btnMoRong:

```
private void btnMoRong_Click(object sender, EventArgs e)
{
    TV_Test.ExpandAll();
}
```

- Sự kiện *Click* của nút btnThuHep:

```
private void btnXoaChon_Click(object sender, EventArgs e)
{
    TV_Test.CollapseAll();
}
```

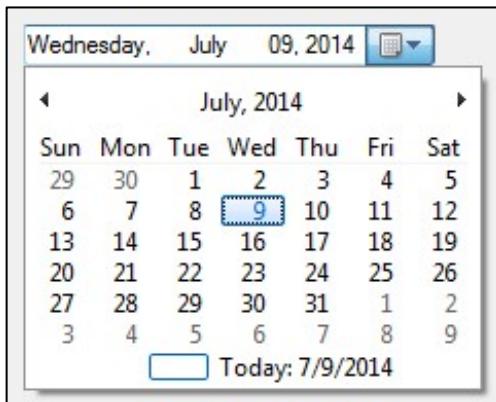
- Sự kiện *Click* của nút btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

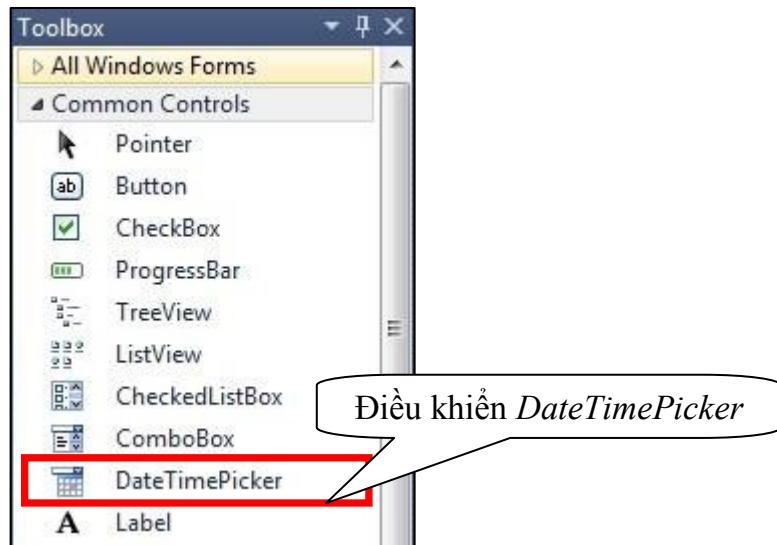
4.5. Điều khiển DateTimePicker, MonthlyCalendar

4.5.1. Điều khiển DateTimePicker

Điều khiển *DateTimePicker* cho phép người dùng chọn ngày tháng như một lịch biểu nhưng biểu diễn ở dạng ComboBox  , khi người dùng nhấp chuột vào ComboBox sẽ sổ xuống lịch biểu như hình 4.23. Các đối tượng ngày tháng biểu diễn trong *DateTimePicker* thực chất là các đối tượng thuộc lớp *DateTime*. Điều khiển *DateTimePicker* được đặt trong nhóm Common Controls của cửa sổ Toolbox như hình 4.24.



Hình 4.23: Điều khiển *DateTimePicker* dạng lịch biểu



Hình 4.24: Điều khiển DateTimePicker trong cửa sổ Toolbox

- Một số thuộc tính thường dùng của DateTimePicker:

Bảng 4.21: Bảng mô tả các thuộc tính của DateTimePicker

Thuộc tính	Mô tả
Format	Định dạng kiểu hiển thị của ngày tháng. Lưu ý: Thường sử dụng giá trị kiểu Short
Value	Trả về giá trị hiện thời của điều khiển DateTimePicker
Value.Date	Trả về ngày tháng năm
Value.Day	Trả về ngày của tháng
Value.Month	Trả về tháng
Value.Year	Trả về năm
Value.DateOfWeek	Trả về ngày của tuần (0 là chủ nhật, 1 là thứ 2, 2 là thứ 3, ... 6 là thứ 7)
Value.DateOfYear	Trả về ngày thứ bao nhiêu của năm
CustomFormat	Cho phép lập trình viên tạo ra một định dạng khác về ngày tháng. Lưu ý: Định dạng ngày tháng năm như kiểu Việt Nam thì kiểu định dạng phải là dd/MM/yyyy. Khi đó thuộc tính format phải thiết lập là Cusom.

<i>MaxDate</i>	Thiết lập ngày lớn nhất cho phép người dùng chọn trên điều khiển <i>DateTimePicker</i>
<i>MinDate</i>	Thiết lập ngày nhỏ nhất cho phép người dùng chọn trên điều khiển <i>DateTimePicker</i>
<i>Text</i>	Trả về ngày hiển thị

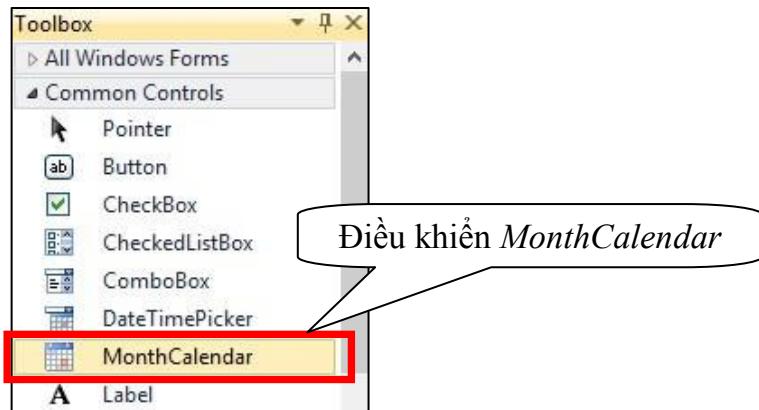
- Một số sự kiện thường dùng của *DateTimePicker*:

Bảng 4.22: Bảng mô tả các sự kiện của DateTimePicker

Sự kiện	Mô tả
<i>ValueChanged</i>	Phát sinh khi người dùng chọn giá trị khác với giá trị trước đó trên điều khiển <i>DateTimePicker</i>
<i>CloseUp</i>	Phát sinh người dùng kết thúc việc chọn ngày trên điều khiển <i>DateTimePicker</i>

4.5.2. Điều khiển MonthCalendar

MonthCalendar là điều khiển hiển thị lịch dưới dạng một lịch biểu cho phép người dùng chọn ngày tháng. Nhưng khác biệt là *MonthCalendar* cho phép người dùng có thể chọn một tập các ngày hay nói cách khác là một tập các đối tượng thuộc lớp *DateTime*. Điều khiển *MonthCalendar* được đặt trong nhóm Common Controls của cửa sổ Toolbox như hình 4.24.



Hình 4.25: Điều khiển MonthCalendar trong cửa sổ Toolbox

- Một số thuộc tính thường dùng của MonthCalendar:

Bảng 4.23: Bảng mô tả các thuộc tính của MonthCalendar

Thuộc tính	Mô tả
<i>MaxDate</i>	Thiết lập ngày lớn nhất cho phép người dùng chọn trên điều khiển MonthCalendar
<i>MinDate</i>	Thiết lập ngày nhỏ nhất cho phép người dùng chọn trên điều khiển MonthCalendar
<i>SelectionRange</i>	Trả về một dãy các ngày liên tục được chọn bởi người dùng
<i>SelectionStart</i>	Trả về ngày đầu tiên trong dãy tại thuộc tính <i>SelectionRange</i>
<i>SelectionEnd</i>	Trả về ngày cuối cùng trong dãy tại thuộc tính <i>SelectionRange</i>
<i>AnnuallyBoldedDates</i>	Chứa một mảng các ngày. Trong mỗi năm, các ngày trong mảng sẽ được bôi đen MonthCalendar
<i>BoldedDates</i>	Chứa mảng các ngày. Các ngày này sẽ được bôi đen trên điều khiển MonthCalendar tại những năm chỉ định.
<i>MaxSelectCount</i>	Thiết lập số lượng ngày tối đa mà người dùng có thể chọn

<i>MonthlyBoldedDates</i>	Chứa mảng các ngày. Trong mỗi tháng, các ngày trong mảng sẽ được bôi đen trên <i>MonthCalendar</i>
---------------------------	--

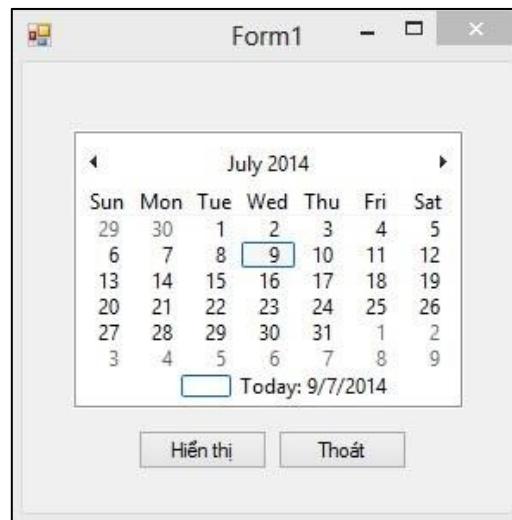
Người dùng có thể chọn một ngày nào đó trên *MonthCalendar* bằng cách nhấp chuột chọn ngày đó hoặc chọn một dãy nhiều ngày liên tiếp bằng cách nhấp chuột chọn ngày đầu tiên và giữ phím shift đồng thời chọn ngày cuối cùng của dãy. Số lượng các ngày được chọn trong dãy phải nhỏ hơn giá trị thiết lập trong thuộc tính *MaxSelectCount*

- Một số sự kiện thường dùng của *MonthCalendar*:

Bảng 4.24: Bảng mô tả các sự kiện của *MonthCalendar*

Sự kiện	Mô tả
<i>DateChanged</i>	Được phát sinh một ngày mới hoặc một dãy các ngày mới được chọn

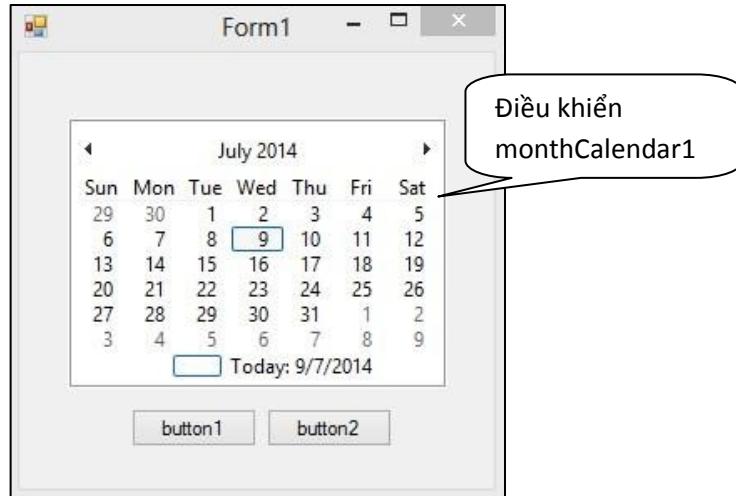
Ví dụ 4.9: Viết chương trình minh họa việc hiển thị lịch, thiết kế giao diện form như hình 4.26. Yêu cầu: khi nhấp nút hiển thị thì các ngày được chọn sẽ hiển thị trên MessageBox



Hình 4.26: Giao diện chương trình hiển thị lịch ví dụ 4.9

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 4.27



Hình 4.27: Giao diện ban đầu ví dụ 4.9

Bước 2: Thiết lập giá trị cho thuộc tính trong cửa sổ Properties

- button1:

Thuộc tính Text: “Hiển thị”

Thuộc tính Name: btnHienThi

- button2:

Thuộc tính Text: “Thoát”

Thuộc tính Name: btnThoat

- monthCalendar1:

Thuộc tính Name: MyMCalendar

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện Click của nút btnHienThi:

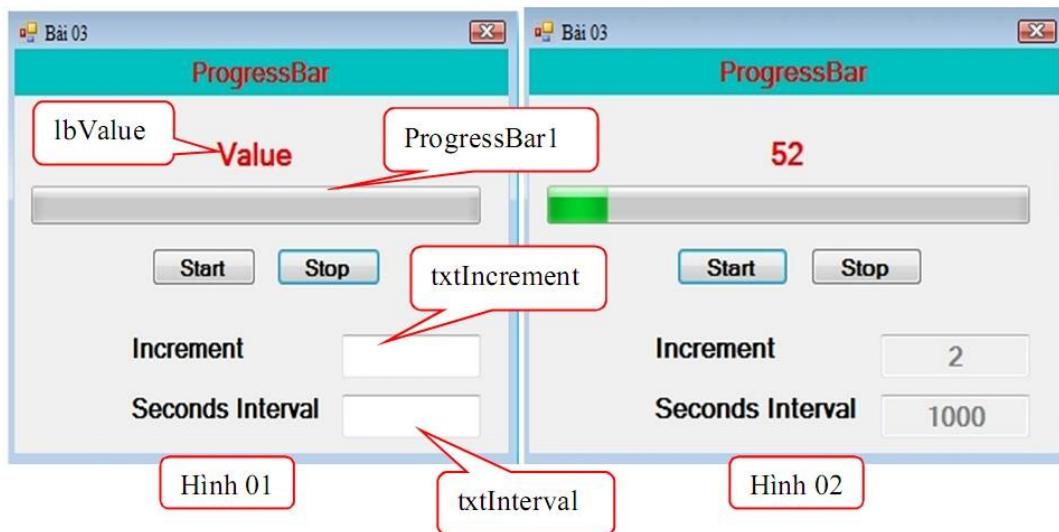
```
private void btnHienThi_Click(object sender, EventArgs e)
{
    string strngay="";
    DateTime i=new DateTime();
    for ( i = MyMCalendar.SelectionStart;
          i<= MyMCalendar.SelectionEnd; i=i.AddDays(1.0))
    {
        strngay += i.ToString() + "\n";
    }
    MessageBox.Show(strngay);
}
```

- Sự kiện *Click* của nút btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

4.6. Bài tập cuối chương

Câu 1: Thiết kế chương trình có giao diện như hình 4.28



Hình 4.28: Giao diện chương trình ProgressBar

Yêu cầu:

- Người dùng nhập giá trị trong TextBox txtIncrement; nhập giá trị trong TextBox txtInterval.
- Khi người dùng nhấn nút Start,ProgressBar1 sẽ tăng giá trị thuộc tính Value lên (trong khoảng từ Minimum đến Maximum) đồng thời thuộc tính Value được gán cho Label lbValue, mỗi lần tăng với giá trị bằng giá trị đã nhập trong txtIncrement, cứ sau mỗi thời gian được nhập trong txtInterval (/1000 giây) sẽ tăng giá trị thuộc tính Value lên. Khi thuộc tính Value có giá trị \geq Maximum thì thuộc tính Value sẽ được gán bằng Minimum trở lại. Ngoài ra khi nhấn nút Start nếu các TextBox đã có giá trị hợp lệ sẽ chuyển sang trạng thái chỉ đọc không cho người dùng chỉnh sửa (Hình 02).
- Khi người dùng nhấn nút Stop sẽ trả về trạng thái ban đầu.

Câu 2: Thiết kế chương trình có giao diện như hình 4.29.

Hình 01

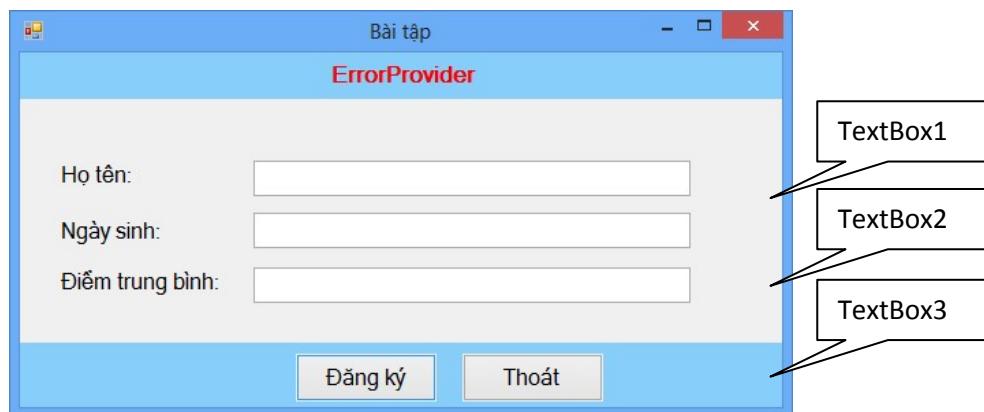
Hình 02

Hình 4.29: Giao diện chương trình chuẩn hóa chuỗi

Yêu cầu:

- Khi bắt đầu chạy chương trình hiển thị một MessageBox với nội dung là “Bạn có muốn tiếp tục Load Form?”, chuỗi hiển thị trên titlebar MessageBox là Họ tên - số máy của sinh viên dự thi.
 - o Nếu người dùng nhấn chọn No thì dừng việc Load Form, ngược lại,
 - o Nếu người dùng nhấn chọn Yes thì hiện Form với giao diện như Hình 01.
- Người dùng nhập một chuỗi bất kỳ vào TextBox “txtChuoi”. Khi người dùng nhấn Button “Chuẩn Hóa” sẽ định dạng lại chuỗi nhập theo yêu cầu sau:
 - o Loại bỏ các khoảng trắng thừa (loại bỏ các khoảng trắng ở đầu và cuối chuỗi, làm sao giữa các từ cách nhau một khoảng trắng.)
 - o Đầu câu nằm sát từ đứng kè trước và cách từ đứng kè sau một khoảng trắng. (Xử lý các dấu câu sau: dấu chấm (“.”), dấu phẩy (“,”), dấu hai chấm (“::”), dấu chấm than (“!”), dấu hỏi (“?”), dấu chấm phẩy (“;”))
- Chuỗi sau khi chuẩn hóa sẽ được hiển thị ở TextBox “txtChuoi”. (Xem Hình 02)
- Khi người dùng nhấn Button “Thống Kê” sẽ thống kê tần số xuất hiện của các từ đơn trong chuỗi ra ListView “lvThongKe” (không phân biệt chữ hoa, chữ thường khi thống kê). (Xem Hình 02)
- Khi người dùng nhấn Button “Lưu” thì sẽ hiện MessageBox thông báo “Đã lưu thành công” và thoát khỏi chương trình.

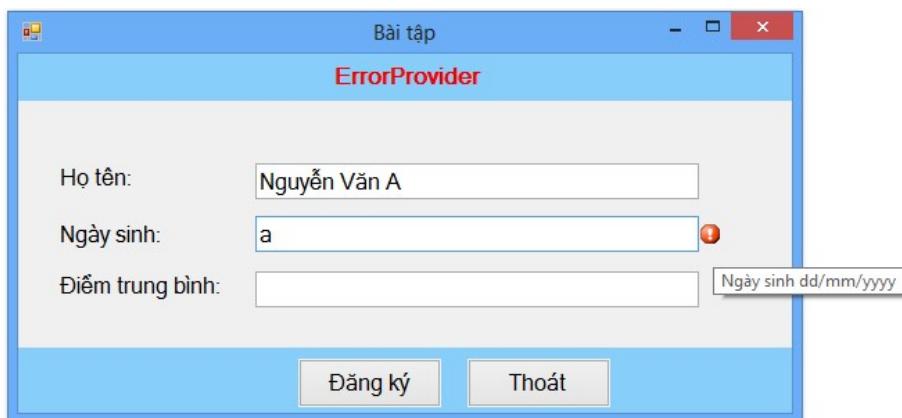
Câu 3: Thiết kế chương trình có giao diện như hình 4.30.



Hình 4.30: Giao diện chương trình nhập thông tin sinh viên

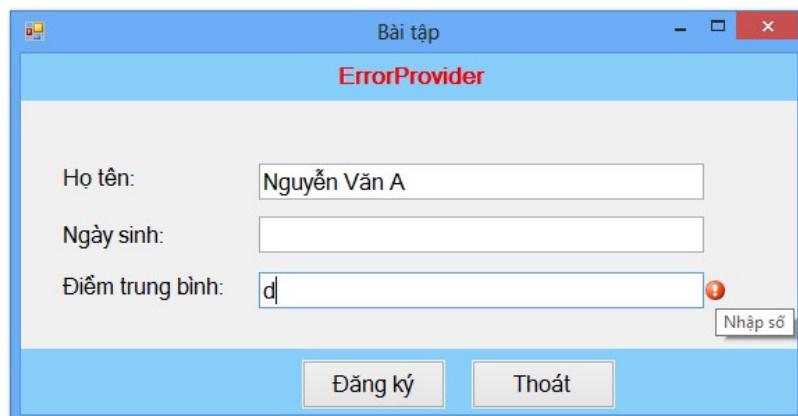
Yêu cầu:

- TextBox2 phải nhập ngày sinh theo định dạng dd/mm/yyyy. Nếu nhập sai thì sẽ hiện thông báo lỗi “Ngày sinh dd/mm/yyyy” như hình 4.31.



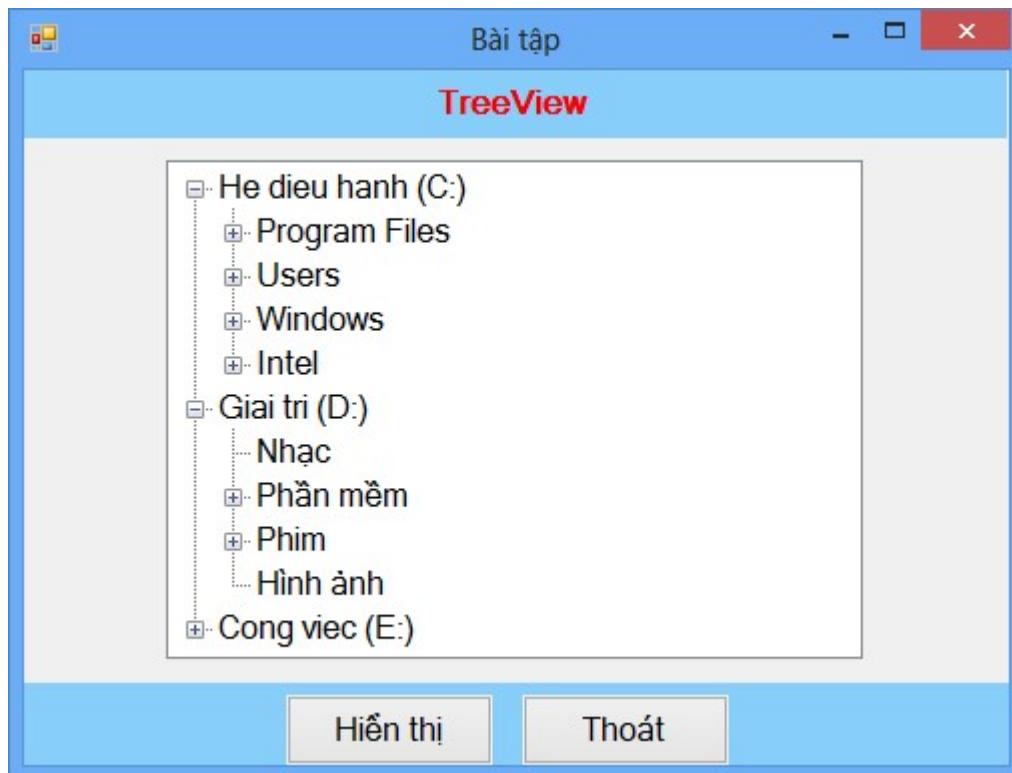
Hình 4.31: Nhập sai định dạng ngày tháng

- TextBox3 phải nhập điểm trung bình là số. Nếu nhập sai thì sẽ hiện thông báo lỗi “Nhập số” như hình 4.32.



Hình 4.42: Nhập sai định dạng số

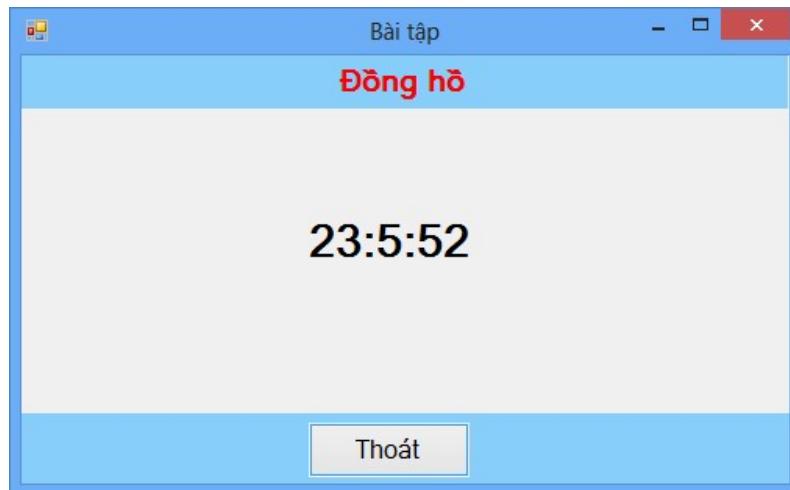
Câu 4: Viết chương trình hiển thị các thư mục trong máy tính. Giao diện chương trình như hình 4.43.



Hình 4.43: Giao diện chương trình hiển thị cây thư mục

- Khi nhấn Button “Hiển thị” thì sẽ hiển thị các thư mục có trong máy tính trên TreeView.
- Khi nhấn Button “Thoát”: đóng chương trình.

Câu 5: Viết chương trình tạo đồng hồ điện tử như hình 4.44 (sử dụng điều khiển Timer)



Hình 4.44: Giao diện chương trình đồng hồ điện tử