



A. MỤC TIÊU:

- Thiết kế được User Control.
- Sử dụng được các User Control đã tạo vào các ứng dụng khác.

B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

1. Tóm tắt lý thuyết

Components

Phân loại: Có 2 loại

- Thành phần xây dựng sẵn của .NET
- Thành phần do người dùng định nghĩa.

1.1. Thành phần xây dựng sẵn của .NET

Tham chiếu đến thành phần xây dựng sẵn:

- Thành phần không nhìn thấy trên thanh công cụ: sử dụng hộp thoại Project/ Add reference
- Thành phần nhìn thấy trên thanh công cụ: kích phải trên tab Toolbox, chọn Choose Items, danh sách các thành phần hiển thị ở Tab COM Components hay .NET Framework Components....

Ví dụ thành phần xây dựng sẵn:

1. Kích phải trên tab Toolbox, chọn Choose Items
2. Chọn các components cần thiết ở tab COM Components hay .NET Framework Components
3. Kích nút OK, biểu tượng Browse Button sẽ xuất hiện trên Toolbox.
4. Kéo điều khiển mong muốn vào Winform, sẽ có các assemblies .NET tương ứng tự động tạo và thêm vào ứng dụng, xem ở mục References trong cửa sổ Solution Explorer.

1.2. Thành phần người dùng định nghĩa

a. Kế thừa từ control đã có

Control thừa kế từ control đã có

1. File/ New Project, Windows Forms Control Library
2. Viết mã kế thừa lớp điều khiển đã có:

3. Ví dụ: `public class NumericTextBox: System.Windows.Forms.TextBox`
4. Viết mã bổ sung hàm, sự kiện, thuộc tính cho lớp
5. Biên dịch để tạo file DLL trong thư mục
<Application Folder>\bin\Debug

b. Custom control không giao diện (tạo thư viện sử dụng)

1. File/ New Project, Class Library (không có giao diện).
2. Viết mã thuộc tính, phương thức, ...
3. Biên dịch ứng dụng để tạo file DLL trong thư mục:
<Application Folder>\bin\Debug

c. Custom control có giao diện

1. File/ New Project, Windows Forms Control Library (có giao diện)
2. Thiết kế giao diện người dùng của user control
3. Viết mã để thêm phương thức, property cho control.
4. Biên dịch ứng dụng, sẽ tạo tập tin DLL trong thư mục:
<Application Folder>\bin\Debug

2. Giới thiệu bài tập mẫu

Bài 1. Tạo Control NumericTextBox chỉ chấp nhận nhập giá trị số.

Hướng dẫn:

1. File/New Project, Windows Forms Control Library, đặt tên **NumericTextBox**
2. Thay mã thừa kế:
`public partial class NumericTextBox : TextBox`
3. Thêm mã sau:
`public partial class NumericTextBox : TextBox`
{
 `public NumericTextBox()`
 {
 `InitializeComponent();`
 }
 //Bổ sung mã sau để yêu cầu người dùng chỉ nhập số
 `protected override void OnKeyPress(KeyPressEventArgs e)`
 {
 //nếu ký tự nhập vào không phải là kiểu số, ký tự xóa trái backspace
 `if (!char.IsNumber(e.KeyChar) && e.KeyChar != '\b')`
 //sự kiện đã được xử lý (bỏ qua thao tác nhập vừa thực hiện)
 `e.Handled = true;`
 }
}

Tạo ứng dụng sử dụng control

1. File/ New Project, Windows Forms Application
2. Kích phải trên All Windows Forms trong Toolbox, chọn Choose Items, ở tab .NET Framework Components, chọn Browse để duyệt đến file NumericTextBox.dll, OK
3. Tạo form, đặt điều khiển NumericTextBox vào form
4. Chạy ứng dụng

Bài 2. Custom control không giao diện (tạo thư viện sử dụng): Kiểm tra độ dài của Tên Card nếu Tên Card có độ dài khác 5 là không hợp lệ.

1. File/ New Project, Class Library
2. Đặt tên Project: CardValidator chứa lớp CardValidator.cs

```
public class CardValidator
{
    public string Name { get; set; }
    public string CardNo { get; set; }
    public CardValidator(string Name, string CardNo)
    {
        this.Name = Name;
        this.CardNo = CardNo;
    }
    public bool Validate()
    {
        int CardNameLength = Name.Length;
        if (CardNameLength == 5)
            return true;
        else
            return false;
    }
}
```

Tạo ứng dụng sử dụng custom control không giao diện

1. File/ New/ Project, chọn Windows Forms Application
2. Để thêm một tham chiếu đến lớp CardValidator, kích phải trên mục Reference trong cửa sổ Solution Explorer, chọn mục Add Reference, chọn Browse để duyệt đến lớp thư viện Card_Validator.dll
3. Thiết kế form chứa 1 button và 1 label
`using CardValidator;`

Tạo một sự kiện click cho button và thực thi lệnh sau:

```
CardValidator Validator = new CardValidator("Name", "1234567");  
if (Validator.Validate())  
    this.lblThongBao.Text = "Valid Card Name: " + Validator.Name;  
else  
    this.lblThongBao.Text = " Invalid Card Name";
```

Bài 3. Custom control có giao diện: Xây dựng Control mới kết hợp Label và Timer

1. File/ New Project, Windows Form Control Library, tên Project là DigitalClock
2. Đặt tên User Control: DigitalClock, thiết kế User Control như sau:
 - Bổ sung điều khiển Label
 - Bổ sung điều khiển Timer với thuộc tính:
 - Interval: 1000
 - Enabled: True
3. Kích đúp vào điều khiển Timer để mở hàm sự kiện Tick, bổ sung mã sau:
label1.Text = DateTime.Now.ToString();
4. Biên dịch ứng dụng: Build/Build Solution

Tạo ứng dụng sử dụng custom control có giao diện

1. File/ New Project, Windows Forms Application
2. Kích phải trên tab All Windows Forms trên Toolbox, Choose Items, tab .NET Framework Components, Browse, duyệt đến DigitalClock.dll
3. Tạo form, bổ sung Control DigitalClock vào form
4. Chạy ứng dụng

Bài 4. Xây dựng User control Login thực hiện đăng nhập vào hệ thống

Yêu cầu:

1. Tạo Project Windows Form Control Library, đặt tên MyLoginControl
2. Đổi tên mặc định UserControl1 thành LoginControl
3. Thiết kế giao diện cho LoginControl như sau:

- Textbox Mật khẩu có thuộc tính PasswordChar là “*”
- Khi người dùng chọn checkbox Hiện thị mật khẩu sẽ cho phép hiển thị giá trị mật khẩu đã nhập và sẽ ẩn đi nếu không chọn vào mục này.
- Button đăng nhập sẽ thực hiện kiểm tra người dùng đã nhập tên đăng nhập hay mật khẩu hay chưa? Nếu chưa hiện MessageBox thông báo.
- Button Hủy: thực hiện thao tác đóng form.

Hướng dẫn:

1. Viết sự kiện cho checkbox Hiện thị mật khẩu

```
private void ckcHienThiPass_CheckedChanged(object sender, EventArgs e)
{
    if (ckcHienThiPass.Checked)
        txtPassword.PasswordChar = (char)0;
    else
        txtPassword.PasswordChar = '*';
}
```

2. Viết sự kiện click cho button Đăng nhập:

```
private void btnLogin_Click(object sender, EventArgs e)
{
    //kiểm tra đã nhập user name chưa
    if (txtUser.Text.Length == 0)
        MessageBox.Show("Hãy nhập tên đăng nhập!"); return;
    //kiểm tra đã nhập password chưa
    if (txtPassword.Text.Length == 0)
        MessageBox.Show("Hãy nhập mật khẩu!"); return;
    /*tạo sự kiện Click để gọi hàm xử lý sự kiện Click do người lập
    trình ứng dụng viết*/
    OnSubmitClicked(sender, e);
}
```

3. Định nghĩa delegate phục vụ cho sự kiện, khai báo sự kiện OnSubmitClicked của button Đăng nhập và định nghĩa hàm xử lý sự kiện:

```
//định nghĩa delegate phục vụ cho event
public delegate void SubmitClickedHandler(object sender, EventArgs e);
//định nghĩa event SubmitClicked cho control
public event SubmitClickedHandler SubmitClicked;
//định nghĩa hàm xử lý sự kiện SubmitClicked
protected virtual void OnSubmitClicked(object sender, EventArgs e)
{
    //kiểm tra xem có hàm xử lý sự kiện SubmitClicked? nếu có thì gọi
    if (SubmitClicked != null)
        SubmitClicked(sender, e); // Notify Subscribers
}
```

4. Thực hiện tương tự cho button Hủy.

5. Viết property (get/set) để gán và trả về giá trị cho 2 textbox tên đăng nhập và mật khẩu:

```
public string UserName
{
    get { return txtUser.Text; }
    set { txtUser.Text = value; }
}
public string Password
{
    get { return txtPassword.Text; }
    set { txtPassword.Text = value; }
}
```

6. Biên dịch ứng dụng tạo tập tin DLL. Nếu có lỗi thì sửa lỗi và biên dịch lại.
7. Tạo ứng dụng sử dụng LoginControl, add LoginControl vào ToolBox và kéo thả vào form (có tên là **loginControl1**)

8. Viết sự kiện **SubmitClicked** cho **loginControl1**

```
private void loginControl1_SubmitClicked(object sender, EventArgs e)
{
    //MessageBox.Show("Đã đăng ký tài khoản : " + loginControl1.UserName);
    if (loginControl1.UserName == "admin" && loginControl1.Password == "123456")
        MessageBox.Show("Đăng nhập thành công! Xin chào: " + loginControl1.UserName);
    else
        MessageBox.Show("Tên đăng nhập hoặc mật khẩu chưa hợp lệ!");
}
```

9. Tương tự viết sự kiện cho loginControl1_CancelClicked.

3. Bài tập tự làm

Kế thừa từ control

Bài 5. Tạo control UpperTextBox chỉ chấp nhận chữ hoa.

Bài 6. Tạo control MailTextBox chỉ chấp nhận khi có ký tự '@' và '.com'. Nếu không tồn tại 2 điều kiện trên thì thông báo lỗi qua ErrorProvider.

Bài 7. Tạo control TextBox không chấp nhận ký tự đặc biệt. Nếu tồn tại ký tự đặc biệt thì thông báo lỗi qua ErrorProvider.

Bài 8. Tạo control PassTextBox chỉ chấp nhận khi có ký tự đặc biệt. Nếu không tồn tại ký tự đặc biệt hoặc ít hơn 6 ký tự thì thông báo lỗi qua ErrorProvider.

Bài 9. Tạo control DataGridView có dòng chẵn 1 màu dòng lẻ 1 màu.

Bài 10. Xây dựng Button sao cho khi Hover chuột qua thì Button đổi màu.

Custom control không giao diện

Bài 11. Tạo thư viện **SqlClass** với các yêu cầu sau:

1. Thành phần dữ liệu: `SqlConnection` connection;
2. Phương thức tạo đối tượng `SqlConnection` với khai báo như sau:

```
public void CreateConnection(string pConnectionString)
{
}
```

3. Phương thức TestConnection()

```
public bool TestConnection()
{
}
```

4. Phương thức thực thi 1 câu truy vấn trả về danh sách kết quả

```
public DataTable ExcuteQuery(string pQuery)
{
}
```

5. Phương thức thực thi thêm, xóa, sửa

```
public bool Insert(string pQuery)
{
}

public bool Update(string pQuery)
{
}

public bool Delete(string pQuery)
{
}
```

Custom control giao diện

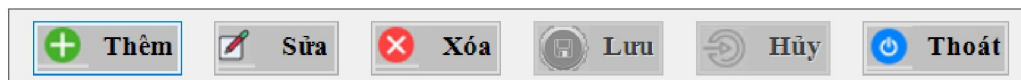
Bài 12. Xây dựng User control tạo các nút (button) Thêm (Add), Sửa (Edit), Xóa (Delete), Lưu (Save), Hủy (Skip), Thoát (Exit) với các yêu cầu:

- Form Load: status (trạng thái) => view

- btnAdd: status => edit
- btnEdit: status => edit
- btnDelete: status => view
- btnSave: status => view
- btnSkip: status => view

Status (trạng thái) : [view, edit]

- edit:
 - enabled = true các button: btnSave, btnSkip
 - enabled = false các button: btnAdd, btnEdit, btnDelete, btnClose
- view:
 - enabled = false các button: btnSave, btnSkip
 - enabled = true các button: btnAdd, btnEdit, btnDelete, btnClose



Bài 13. Xây dựng ứng dụng cập nhật dữ liệu cho table Khoa trong CSDL QLSinhVien như sau (Sử dụng thư viện SQLClass trong bài 11 và User control trong bài 12 đã tạo)

CSDL QLSinhVien:

Khoa (MaKhoa, TenKhoa)

Lop (MaLop, TenLop, *MaKhoa*)

SinhVien (MaSinhVien, HoTen, NgaySinh, *MaLop*)

MonHoc (MaMonHoc, TenMonHoc)

Diem (*MaSinhVien*, *MaMonHoc*, Diem)

Giao diện ứng dụng:

Mã Khoa	Tên Khoa
CNTP	Công nghệ Thực phẩm
CNTT	Công nghệ Thông tin
CT	Chính trị
HH	Hóa học
KT	Kê toán
*	

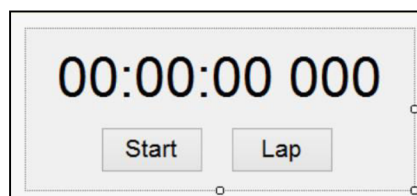
Yêu cầu: Thực hiện load dữ liệu table Khoa lên datagridview và cập nhật dữ liệu table Khoa từ textbox Mã khoa và Tên khoa.

4. Bài tập về nhà

Bài 14. Thiết kế User Control sau với: MSSV truyền từ biến ngoài. Thông tin sinh viên được lưu trên file txt

Mã sinh viên:	3951140202067	Ngày sinh:	29/06/1999
Tên sinh viên:	Huỳnh Ngọc Bích	Giới tính:	Nữ
Mã lớp:	04DHTH1	CMND:	

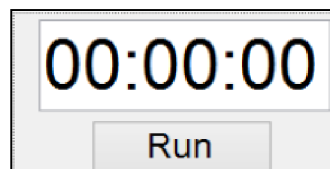
Bài 15. Đồng hồ bấm giờ



Yêu cầu:

- Bấm Start thì cho đồng hồ chạy đồng thời đổi text thành Stop.
- Bấm Stop thì cho đồng hồ dừng và đổi text thành Start.
- Bấm Lap (ghi giờ) thực hiện ghi xuống file số lần cho trước.

Bài 16. Đồng hồ đếm ngược



Yêu cầu:

- Thiết kế và thực hiện coding đồng hồ đếm ngược cho phép người dùng nhập giá trị vào khi bấm “**Run**” thì tiến hành đếm ngược.