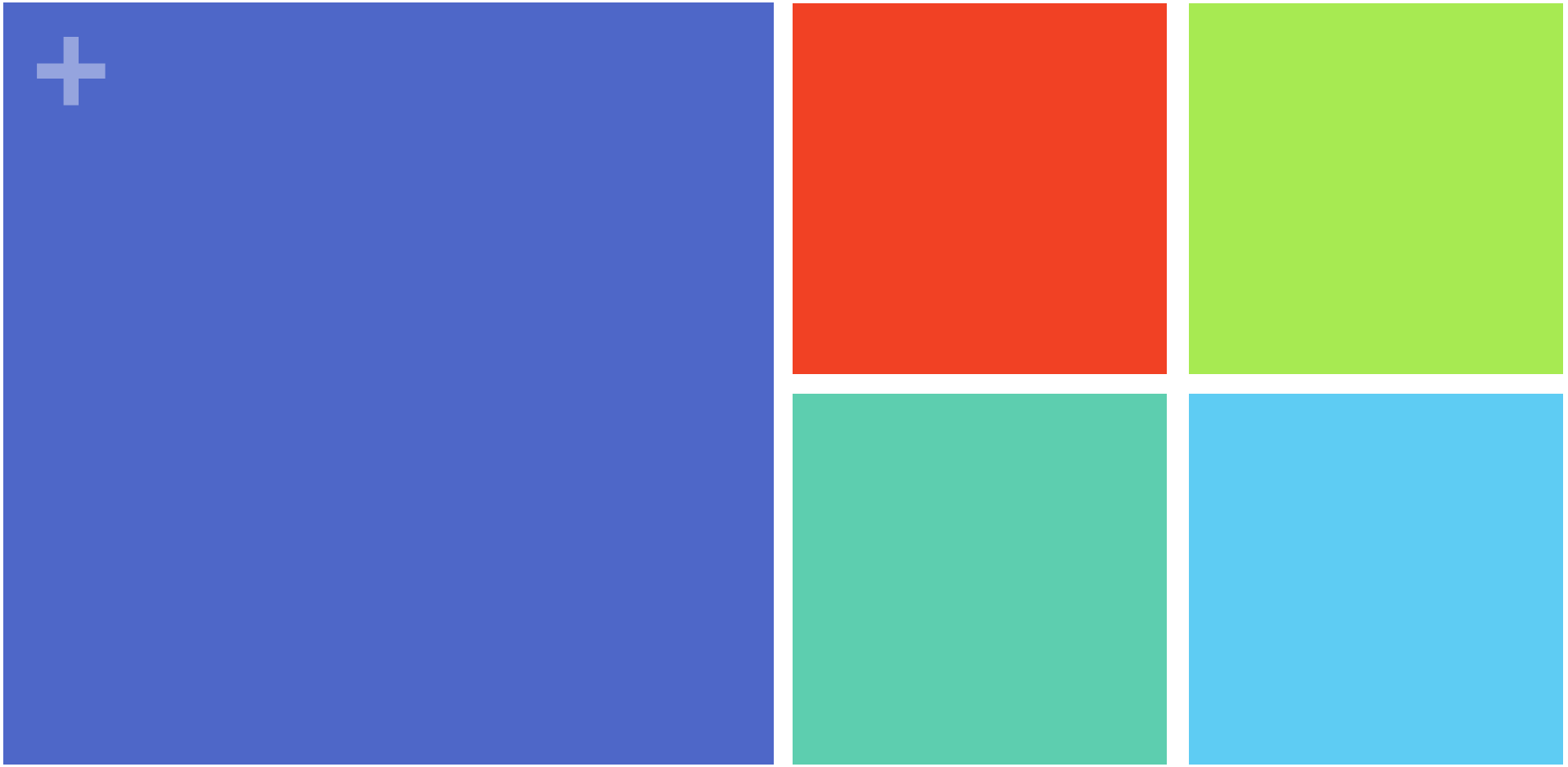


WINDOW FORM

TS. ĐẶNG THÀNH TRUNG

NỘI DUNG CHÍNH

- Xây dựng ứng dụng Windows Form
- Thuộc tính của Windows Form
- Bắt sự kiện (Event Handling)
- Sử dụng namespace System.Drawing

XÂY DỰNG ỨNG DỤNG Windows Form

SỬ DỤNG LỚP System.Windows.Forms.Form

- Một ứng dụng Windows thường có một hoặc nhiều Windows Form.
  - Window Form là một vùng màn hình được sử dụng để thiết kế giao diện.
  - Là nơi để đặt các thành phần giao diện khác như: Textbox, Button,Llist, Grid, Menu…
  - Window Form là một thể hiện cụ thể của lớp Form trong namespace ***System.Windows. Forms***

THIẾT KẾ Windows Form

- Sử dụng Visual Studio .NET để xây dựng Windows Form
- Khi tạo ra một ứng dụng Windows mới, VS.NET tự động bổ sung một form vào ứng dụng.
  - Mỗi form có một file mã lệnh tương ứng (vd:Form1.cs).
    - File này chứa các định nghĩa về Windows Form để ta có thể nhìn thấy khi thực thi chương trình.
- Phân biệt Project và Solution:
  - Solution được sử dụng để nhóm một hoặc nhiều Project.
  - Có thể tạo ra một Solution và sau đó bổ sung các Project vào Solution đó.
  - Nếu trực tiếp tạo ra Project thì VS.NET sẽ tự động tạo ra một Solution để chứa Project này.

MÃ LỆNH TỰ ĐỘNG

- VS.NET nhóm mã lệnh thành các block (gọi là kỹ thuật code outlining).
  - Ta có thể mở hoặc đóng các block này.
  - Directive: #region và #endregion đánh dấu điểm bắt đầu và kết thúc của một khối lệnh.
    - Có thể đặt tên cho block, để khi đóng block ta vẫn thấy tên của nó.
    - Khi dịch chương trình directive này sẽ bị bỏ qua.

MÃ LỆNH TỰ ĐỘNG

- Sử dụng using để xác định những namespace cần sử dụng.
- Khi tạo ra một Windows Form, VS.NET sẽ định nghĩa một lớp kế thừa lớp Form trong namespace System.Windows.Forms.

```
public class Form1 : System.Windows.Forms.Form {
        // …
}
```

- Có thể thay đổi trực tiếp tên lớp trên code hoặc thông qua thuộc tính Name.

MÃ LỆNH TỰ ĐỘNG

- VS.NET tự động bổ sung hàm tạo mặc định Form1, hàm này gọi đến phương thức InitializeComponent().

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

  - Phương thức InitializeComponent() lưu những thuộc tính đã thay đổi của form khi thiết kế.

```
private void InitializeComponent()
{
        this.components = new System.ComponentModel.Container();
        this.Size = new System.Drawing.Size(300,300);
        this.Text = "Form1";
}
```

MÃ LỆNH TỰ ĐỘNG

- Phương thức Dispose() được sử dụng để dọn dẹp vùng nhớ khi ta đã kết thúc công việc với lớp.
- CLR có khả năng dọn dẹp rác tự động (garbage collection). Tất cả những vùng nhớ không còn sử dụng nữa sẽ tự động được dọn dẹp.

```
protected override void Dispose( bool disposing )
{
        if( disposing )
        {
                if (components != null)
                {
                        components.Dispose();
                }
        }
        base.Dispose( disposing );
}
```

MÃ LỆNH TỰ ĐỘNG

- Một ứng dụng phải có điểm bắt đầu thực thi - được định nghĩa trong phương thức Main().
- Một Project chỉ có một Form chứa hàm Main().

```
[STAThread]
static void Main() {
        Application.Run(new Form1());
}
```

  - Dòng lệnh đầu tiên là thuộc tính được gán cho phương thức Main() để xác định ứng xử trong lúc thực thi.
  - Thuộc tính STAThread cho phép ứng dụng tham gia vào môi trường COM (Component Object Model) hoặc làm bất kỳ việc gì sử dụng Object Linking and Embedding (như thao tác kéo thả hoặc sử dụng Clipboard).
  - Câu lệnh viện dẫn tới phương thức Run() của lớp Application.
  - Sử dụng Form1() trong lệnh Application.Run sẽ gọi tới hàm tạo của Form1, để trả về một thể hiện mới của lớp Form1 và sẽ được hiển thị trên màn hình khi ứng dụng bắt đầu

LỚP Application

- Lớp Application chịu trách nhiệm quản lý ứng dụng Window.
- Các phương thức và thuộc tính của lớp Application đều là static.
  - Không thể tạo ra một thể hiện cụ thể của lớp Application
  - Sử dụng các thuộc tính và phương thức trực tiếp thông qua tên lớp.
- Ví dụ: Sử dụng các thuộc tính của lớp Application

MỘT SỐ THUỘC TÍNH VÀ PHƯƠNG THỨC LỚP Application

CompanyName	xác định tên của tổ chức có liên quan tới ứng dụng
CurrentCulture	xác định thông tin về Thread hiện tại
CurrentInputLanguage	xác định ngôn ngữ đầu vào hiện tại đã khởi đầu ứng dụng.
ExecutablePath	xác định đường dẫn của file exe khởi đầu ứng dụng.
ProductName	xác định tên của sản phẩm
ProductVersion	xác định phiên bản của sản phẩm.
DoEvents()	xử lý tất cả các thông điệp Windows trong hàng đợi thông điệp.
Exit()	thoát ra khỏi vòng lặp thông điệp hiện tại và đóng tất cả các cửa sổ trong Thread.
Run()	bắt đầu một vòng lặp thông điệp ứng dụng chuẩn trong Thread hiện tại. Phương thức Run() khởi tạo ứng dụng bằng cách tạo ra một Form trên màn hình và gửi ứng dụng tới một vòng lặp thông điệp. Ứng dụng sẽ nằm trong vòng lặp và đáp ứng các thông điệp của người sử dụng cho đến khi vòng lặp thông điệp được đóng lại do Form bị đóng.

LỚP MessageBox

- Thuộc vào Namespace System.Windows.Forms và thừa kế từ lớp Object.
- Cung cấp phương thức Show(), có thể sử dụng theo 12 cách khác nhau để hiển thị thông điệp.
  - Tham số MessageBoxButtons – xác định kiểu thông báo
  - Tham số MessageBoxIcons – xác định biểu tượng trên thông báo

THUỘC TÍNH CỦA Windows Form

THUỘC TÍNH Windows Form

- Tất cả các thuộc tính được truy nhập dễ dàng thông qua cửa sổ Properties của VS.NET.
- Ví dụ: Sử dụng Visual Designer để thay đổi thuộc tính của Windows Form

MỘT SỐ THUỘC TÍNH CỦA FORM

BackColor:	xác định màu nền của form
BackgroundImage:	xác định ảnh nền của form
ClientSize:	xác định kích thước của vùng chính trong form
ControlBox	chỉ ra rằng liệu hộp điều khiển có cần hiển thị trên form
DesktopLocation	xác định vị trí của form trên màn hình
Enabled	chỉ ra rằng liệu điều khiển có thể đáp ứng được tương tác của người sử dụng
FormBoderStyle	xác định kiểu đường biên của Form
HelpButton:	liệu nút lệnh Help có cần hiển thị không
Icon:	xác định biểu tượng của form
MaximizeBox:	xác định nút lệnh Max có được hiển thị không
MaximumSize:	xác định kích thước lớn nhất có thể thay đổi.
MinimizeBox:	xác định nút lệnh Min có được hiển thị không
MiniimumSize:	xác định kích thước nhỏ nhất có thể thay đổi.
Modal:	form có được hiển thị bình thường không
ShowInTaskbar	liệu form có được hiển thị trên Window taskbar không
Size	kích thước của form
StartPosition	vị trí bắt đầu hiển thị của form
TopMost	liệu form có được hiển thị ở vị trí cao nhất trong ứng dụng.

THUỘC TÍNH Windows Form

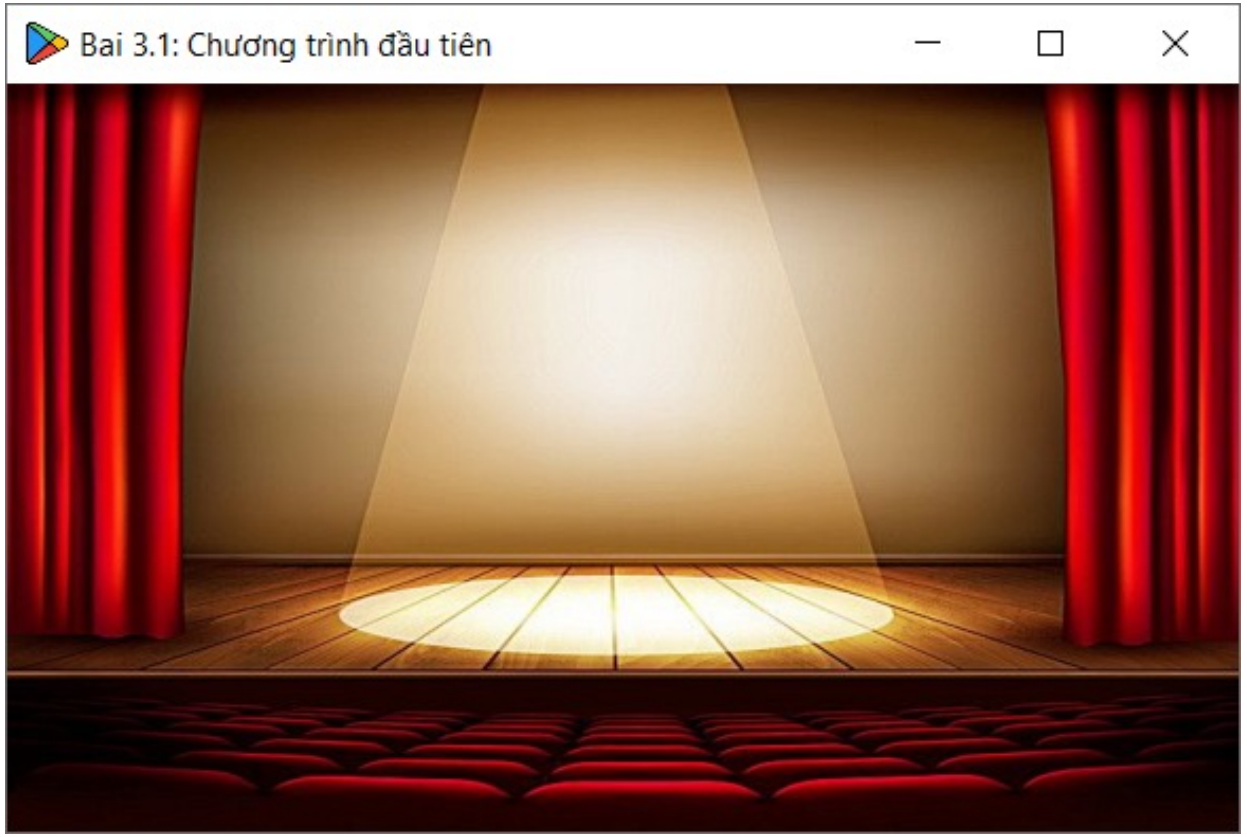
- Visual Designer chỉ cho phép gán các thuộc tính tại thời điểm thiết kế.
  - Mã lệnh tương ứng với sự thay đổi thuộc tính của Form được lưu trong phương thức InitalizeComponent().
- Muốn thay đổi thuộc tính của Form vào lúc thực thi thì phải viết mã lệnh.
- Ví dụ: Thay đổi thuộc tính của Windows Form thông qua mã lệnh

BÀI TẬP THỰC HÀNH

- **Bài 3.1:** Tạo một dự án Window form, chỉnh sửa và định dạng form như sau:
  - Tiêu đề: "Bài 3.1: Chương trình đầu tiên"
  - Màu nền: OrangeRed
  - Kích thước: 600x400
  - Vị trí: Giữa màn hình

BÀI TẬP THỰC HÀNH

- **Bài 3.1:** Thêm icon và hình nền cho Form



BẮT SỰ KiỆN
(Event Handling)

SỰ KIỆN VÀ BẮT SỰ KIỆN

- Khi ta thực hiện một hành động trên đối tượng, đối tượng sẽ tạo ra các sự kiện trong ứng dụng.
- Bắt sự kiện là một phương thức được thực hiện để đáp ứng cho một sự kiện.
- Phân loại sự kiện
  - Sự kiện được kích hoạt bởi người sử dụng.
  - Sự kiện được kích hoạt bởi sự thay đổi của môi trường như gửi/nhận thư, chỉnh sửa file, thay đổi thời gian …
  - Có thể tự định nghĩa các sự kiện hoặc viết mã lệnh để bắt các sự kiện.

BẮT SỰ KIỆN BỞI Delegate

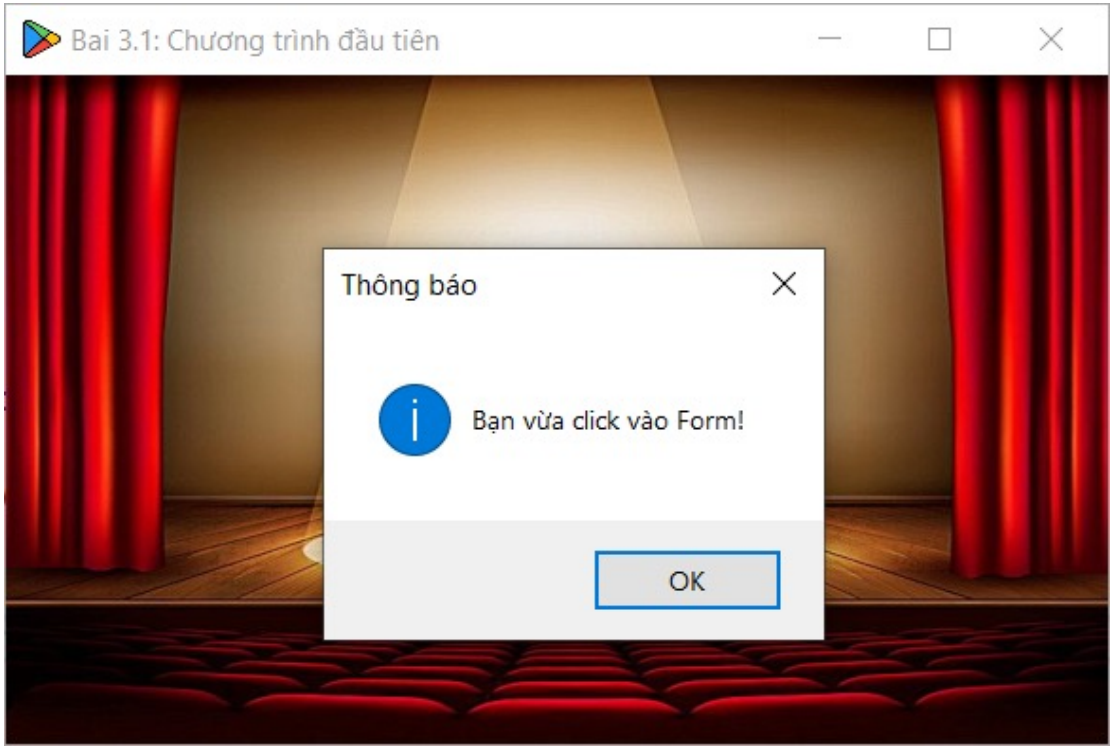
- Lớp Form có một tập các sự kiện đã được định nghĩa và có thể truy nhập thông qua cửa sổ Properties.
- Nếu chương trình phải thực hiện hành động khi một sự kiện được kích hoạt, thì phải định nghĩa bộ bắt sự kiện thích hợp.
  - Bộ bắt sự kiện phải được đăng ký với nguồn của sự kiện
  - Khi sự kiện xảy ra, bộ bắt sự kiện tương ứng sẽ được viện dẫn.
  - Có thể có nhiều bộ bắt sự kiện đáp ứng một sự kiện hoặc một bộ bắt sự kiện đáp ứng nhiều sự kiện.

VÍ DỤ BẮT SỰ KIỆN

- Bắt sự kiện click chuột
- Bộ bắt sự kiện có kiểu trả về là void và có 2 tham số.
  - Tham số thứ nhất là đối tượng mà sự kiện xảy ra trên nó
  - Tham số thứ hai là kiểu EventArgs hoặc một kiểu thừa kế từ EventArgs có chứa dữ liệu liên quan đến sự kiện.

BÀI TẬP THỰC HÀNH

- **Bài 3.1:** Thiết lập sự kiện Click cho Form, khi nhấn chuột trái vào Form sẽ hiển thị hộp thoại thông báo



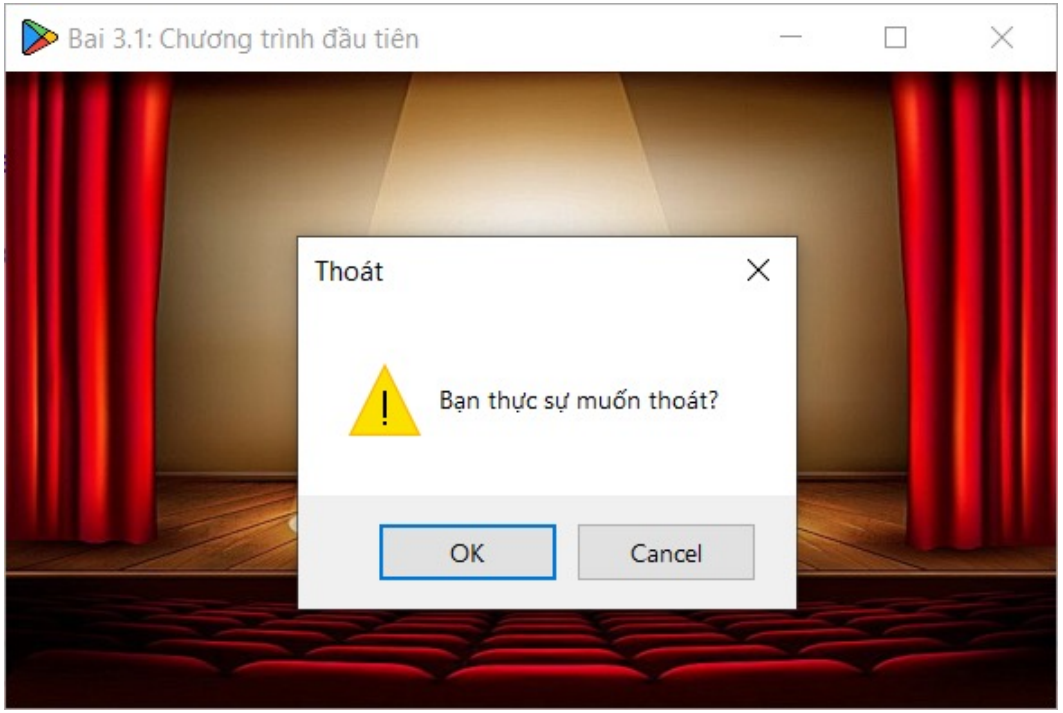
BÀI TẬP THỰC HÀNH

- **Bài 3.1:** Thiết lập sự kiện Click cho Form, khi nhấn chuột trái vào Form sẽ hiển thị hộp thoại thông báo

```
1 reference
private void Bai3_1_MouseClick(object sender, MouseEventArgs e)
{
    MessageBox.Show("Bạn vừa click vào Form!",
        "Thông báo",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
```

BÀI TẬP THỰC HÀNH

- **Bài 3.1:** Nếu người dùng nhấn nút X để đóng ứng dụng, hiển thị hộp thoại hỏi người dùng có muốn tắt ứng dụng hay không.
  - Chọn Ok thì ứng dụng sẽ tắt.



BÀI TẬP THỰC HÀNH

- **Bài 3.1:** Nếu người dùng nhấn nút X để đóng ứng dụng, hiển thị hộp thoại hỏi người dùng có muốn tắt ứng dụng hay không.
  - Chọn Ok thì ứng dụng sẽ tắt.

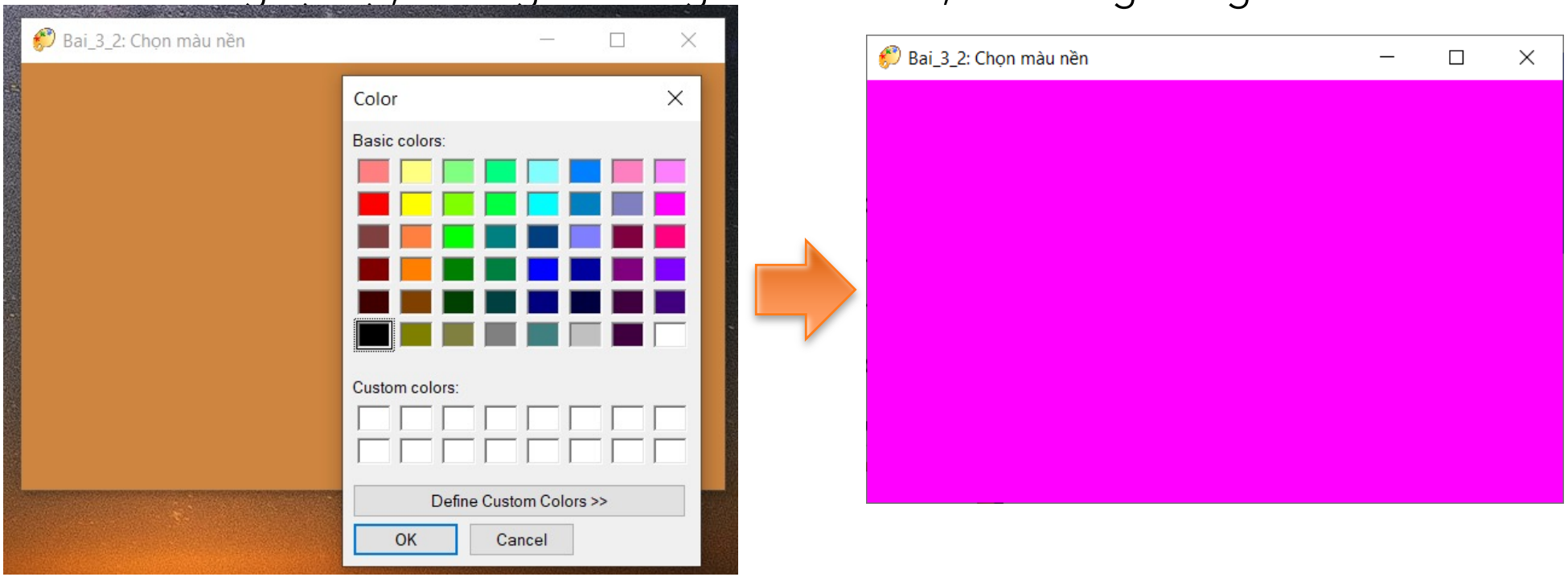
```
1 reference
private void Bai3_1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult result = MessageBox.Show(
        "Bạn thực sự muốn thoát?",
        "Thoát",
        MessageBoxButtons.OKCancel,
        MessageBoxIcon.Warning);
    if (result == DialogResult.OK)
    {
        Application.ExitThread();
    } else
    {
        e.Cancel = true;
    }
}
```

BÀI TẬP THỰC HÀNH

- **Bài 3.2:** Tạo một dự án Window form, chỉnh sửa và định dạng Form như sau:
  - Tiêu đề: "Bài 3.2: Chọn màu nền"
  - Màu nền: Peru
  - Kích thước: 600x400
  - Vị trí: Giữa màn hình
  - Có Icon

BÀI TẬP THỰC HÀNH

- **Bài 3.2:** Khi người dùng click chuột trái vào nền form thì hiển thị một hộp màu sắc để chọn
  - Nếu người dùng bấm OK, thì đổi màu nền Form thành màu vừa chọn
  - Ngược lại, nếu người dùng bấm Cancel, thì không làm gì.



BÀI TẬP THỰC HÀNH

- **Bài 3.2:** Khi người dùng click chuột phải vào nền form thì hiển thị một hộp màu sắc để chọn
  - Nếu người dùng bấm OK, thì đổi màu nền Form thành màu vừa chọn
  - Ngược lại, nếu người dùng bấm Cancel, thì không làm gì.

```
1 reference
private void Bai_3_2_MouseClick(object sender, MouseEventArgs e)
{
    ColorDialog colorDialog = new ColorDialog();
    if (colorDialog.ShowDialog() == DialogResult.OK)
    {
        this.BackColor = colorDialog.Color;
    }
}
```

using namespace System.Drawing

HỆ TỌA ĐỘ CỦA Windows Form

- Window Form là đối tượng có hệ toạ độ 2 chiều.
- Vị trí của control đặt trên Form được xác định bởi một tập hợp điểm.
  - Điểm là một cặp số (x,y) thể hiện toạ độ tính từ gốc của Form.
  - Gốc của form là góc trên bên trái của vùng client.
  - Vùng client là vùng bên trong form bỏ qua phần tiêu đề, biên, menu,…
    - Điểm gốc có toạ độ là (0,0).
    - Giá trị của toạ độ x tăng dần về phía bên phải Form
    - Giá trị của toạ độ y tăng dần về phía dưới Form.

LỚP Graphics

- FCL cung cấp một tập hợp các cài đặt có sẵn của giao diện thiết kế đồ hoạ Windows (GDI+).
- Lớp Graphics cung cấp các phương thức để thực hiện các thao tác trên đối tượng đồ hoạ.
  - Là lớp sealed.
  - Không có hàm tạo. Có 4 cách để nhận được một đối tượng Graphics:
    - Thông qua thuộc tính Graphics của tham số PainEventArgs được truyền tới bộ bắt sự kiện Paint của Control hoặc Form.
    - Gọi phương thức CreateGraphics() của Control hoặc Form
    - Gọi phương thức Graphics.FromHwnd().
    - Gọi phương thức static Graphics.FromImage().

MỘt SỐ CẤU TRÚC TRONG namespace System.Drawing

CharacterRange	Xác định một dãy các vị trí của ký tự trên string.
Color	Xác định cấu trúc màu
Point	Xác định toạ độ x,y (kiểu int)
PointF	Xác định toạ độ x,y (kiểu float).
Rectangle	Lưu vị trí và kích thước của một vùng hình chữ nhật. Có thể tạo ra cấu trúc Rectangle bằng cách sử dụng cấu trúc Point và cấu trúc Size. Point biểu diễn vị trí góc trên bên trái của hình chữ nhật. Size xác định kích thước rộng và cao của hình chữ nhật.
RectangleF	Tương tự như Rectangle, nhưng kiểu dữ liệu là float.
Size	Biểu diễn kích thước của vùng hình chữ nhật, có cặp dữ liệu là chiều rộng và chiều cao.
SizeF	Tương tự như Size nhưng kiểu dữ liệu là float.

MỘT SỐ PHƯƠNG THỨC VẼ CỦA LỚP Graphics

DrawArc()	vẽ cung
DrawClosedCurve()	vẽ đường cong khép kín dựa trên một mảng điểm
DrawCurve()	vẽ đường cong dựa trên một mảng điểm
DrawEllipse()	vẽ đường elip dựa trên hình chữ nhật được xác định bởi 1 cặp toạ độ và các kích thước cao, rộng
DrawIcon()	vẽ ảnh được xác định bởi một đối tương Icon tại vị trí toạ độ cho trước.
DrawImage()	vẽ đối tượng Image tại vị trí xác định, giữ nguyên kích thước ban đầu của ảnh.
DrawLine()	vẽ đường thẳng nối 2 điểm
DrawLines()	vẽ một tập hợp các đường thẳng nối một mảng điểm
DrawPath()	vẽ đối tượng GraphicsPath
DrawPie()	vẽ pie dựa trên elip và 2 bán kính
DrawPolygon()	vẽ đa giác được định nghĩa bởi một mảng điểm
DrawRectangle()	vẽ hình chữ nhật được xác định bởi 1 điểm, chiều rộng và chiều cao
DrawRectangles()	vẽ một tập các hình chữ nhật
DrawString()	vẽ string tại một vị trí xác định, được xác định bởi các đối tượng Brush và Font

PHƯƠNG THỨC DrawString

- Lớp Graphics cung cấp phương thức DrawString() để vẽ một string.

```
private void onFormClick(object sender, EventArgs e)
{
    this.BackColor = Color.Blue;
    Graphics grp = this.CreateGraphics();
    Font font = new Font("Arial",14);
    SolidBrush brush = new SolidBrush(Color.White);
    grp.DrawString("Hello", Font, brush, 50, 50);
}
```

```
private void onPaint(object sender, PaintEventArgs e)
{

    Graphics grp = e.Graphics;
    Font font = new Font("Arial", 14);
    SolidBrush brush = new SolidBrush(Color.White);
    grp.DrawString("Hello", Font, brush, 50, 100);

}
```

- Phương thức Paint:
  - Event handler của sự kiện Paint cung cấp truy nhập tới đối tượng Graphics.
  - Sự kiện Paint được kích hoạt bất kỳ khi nào form được vẽ lại

SỰ KIỆN PAINT

- Khi thay đổi kích thước của Form, sự kiện Paint không được kích hoạt, mà kích hoạt sự kiện Resize().
  - Phương thức Invalidate() tạo ra một thông điệp vẽ gửi tới Form.
    - Invalidate() không có tham số sẽ gọi tới sự kiện Paint trên toàn Form.
    - Invalidate() có tham số kiểu Rectangle sẽ vẽ lại một phần của Form.
  - Thuộc tính ResizeRedraw của lớp Form được gán bằng true thì sẽ chỉ thị cho Control hoặc Form vẽ lại khi Form bị thay đổi kích thước.

MỘT SỐ PHƯƠNG THỨC TÔ MÀU

FillClosedCurve()	tô đậm bên trong của đường cong khép kín được xác định dựa trên một mảng các điểm.
FillEllipse()	tô đậm bên trong của elip nằm trong một hình chữ nhật.
FillPath()	tô bên trong của đối tượng GraphicsPath
FillPie()	tô phần bên trong của phần pie được xác định bởi 1 elip và 2 trục.
FillPolygon()	tô phần bên trong của hình đa giác được xác định bởi một mảng các điểm
FillRectangle()	tô phần bên trong của hình chữ nhật được xác định bởi một điểm, chiều rộng và chiều cao
FillRectangles()	tô phần bên trong của một tập hình chữ nhật
FillRegion()	tô phần bên trong của đối tượng Region

PHÂN BIỆT PHƯƠNG THỨC Draw và Fill

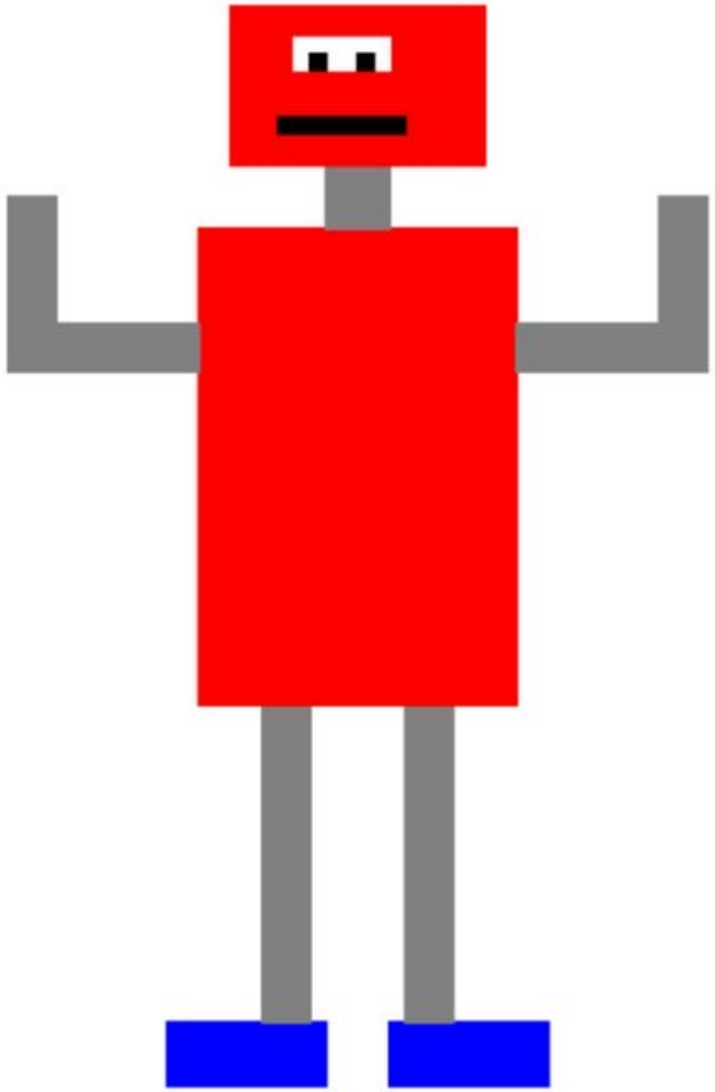
- Cú pháp của phương thức Fill cũng tương tự như phương thức Draw.
- Để tô màu đối tượng cần vẽ.
  - Phương thức Fill sử dụng đối tượng Brush
  - Phương thức Draw sử dụng đối tượng Pen

LỚP Image

- Lớp System.Drawing.Image cung cấp các chức năng cơ bản để làm việc với ảnh.
- Lớp Image là trừu tượng => Phải sử dụng thông qua các lớp sau:
  - Bitmap: được sử dụng để làm việc với các file đồ hoạ được lưu theo các pixel như: BMP, GIF, JPEG.
  - Icon: tạo ra các ảnh bitmap nhỏ, là các icon trong Window.
  - MetaFile: kết hợp bitmap với các chuỗi bản ghi nhị phân để biểu diễn các thao tác đồ hoạ như vẽ đường thẳng.

BÀI TẬP THỰC HÀNH

Bài 3.3: Vẽ hình sau



BÀI TẬP THỰC HÀNH

Bài 3.4: Vẽ hình sau