

National Taipei University of Technology

Windows Programming (Fall 2024)

Homework #3

Deadline: 10/30 (Wed.), before 23:59

壹、注意

請遵守以下規則，否則單次成績以 0 分計算

1. 準時繳交作業，逾時不候
2. 不准抄襲他人作業，請自己完成

貳、主題

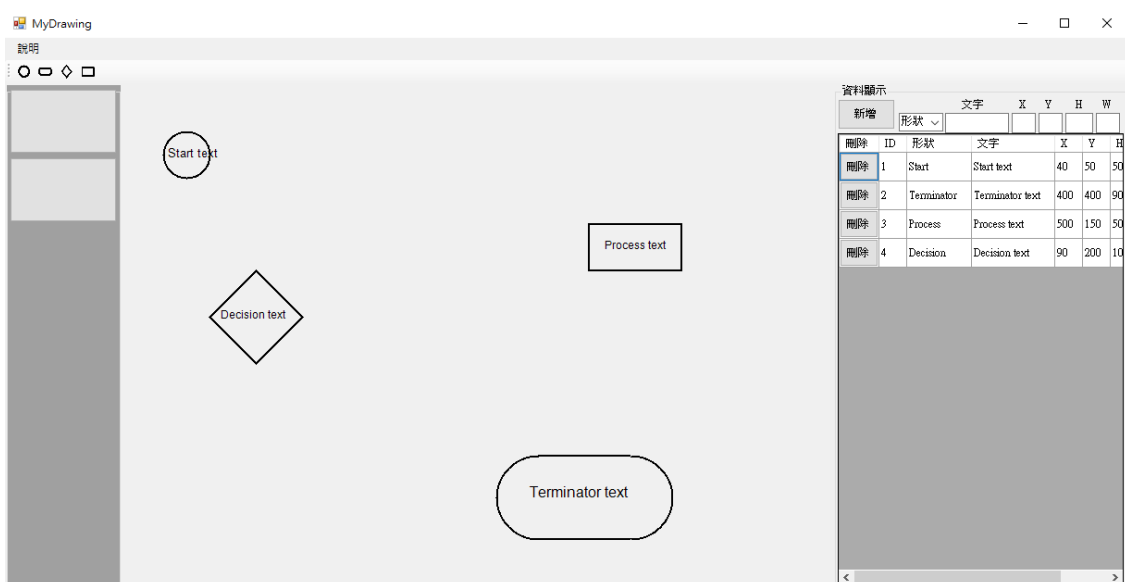
我們要開發一個流程圖繪圖軟體，稱為MyDrawing，一共會分為四個區塊，分別為選單、頁面選擇、繪圖區塊、圖形資料顯示。

本次作業將完成圖形資料顯示區的**圖形資料新增、顯示、刪除**等功能以及 Toolbar，並且**需要實際繪圖至繪圖區**。請注意作業為連續作業，我們需要確保程式的設計是易於維護和擴展的，以便於未來可以方便地增加更多的功能和需求。

參、題目介紹



一、[2 pts] GUI (使用者介面)

請在界面上方新增一**工具列**(如圖一所示)。此 Toolbar (在Windows Forms的控制項稱為ToolStrip)應包含四個按鈕：「**Start**」、「**Terminator**」、「**Process**」、「**Decision**」。



圖一

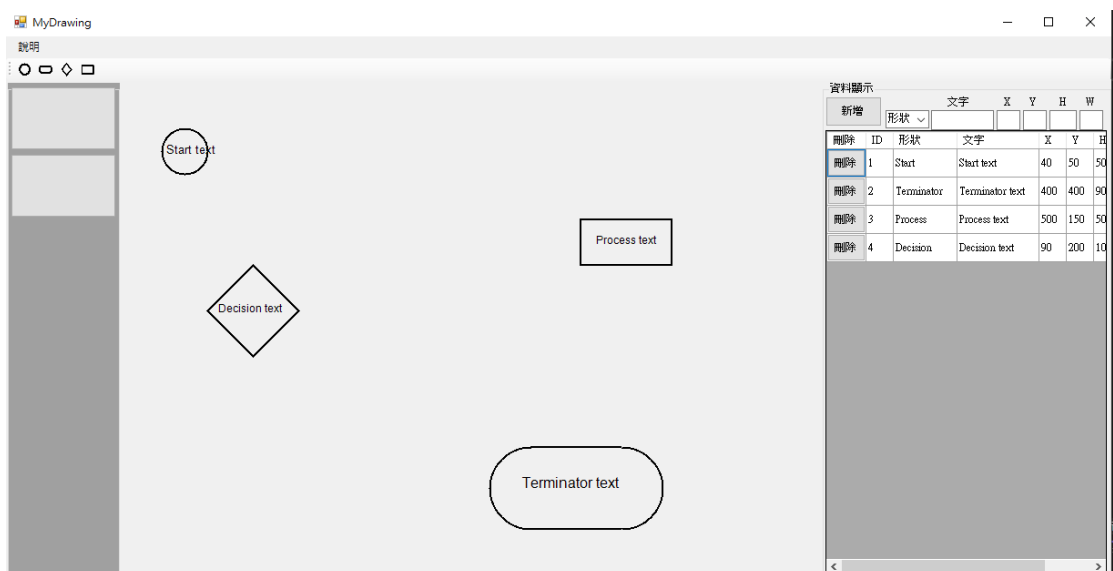
二、[2 pts] 點擊Toolbar 按鈕與按鈕之 Checked 屬性

當程式啟動時，四個流程圖元件按鈕(Start, Terminator, Process, Decision)的 Checked 屬性均應為 false，當使用者自 Toolbar 點擊其中一個按鈕後，此按鈕的 Checked 屬性應改為 true，而其他按鈕的 Checked 屬性應改為 false，且當游標進入繪圖區時，游標應自動變更為『十字形』(參考power point的繪圖方式)，若離開繪圖區則游標恢復原狀。註：(1) 當 Toolbar 按鈕的 Checked 屬性為 true 時，按鈕會有一個外框(); 為 false 時則無外框(); (2) 有關Checked邏輯的實作應該採用Presentation Model，但是本次作業暫時不要求使用之。

三、[5 pts] 圖形繪製

當使用者點擊 Toolbar 上的圖形按鈕(選擇圖形)後，即可在繪圖區使用滑鼠左鍵由左上角拖至右下角畫出一個使用者選擇的圖形，在拖曳過程中，游標保持十字形，被繪出的圖形會因游標位置之變更而變更大小，且在放開左鍵後，該圖形會被固定顯示在繪圖區(如圖二所示)，而游標則恢復為正常的指標形狀、圖形按鈕的 Checked 屬性變成 false。同時請注意，當使用者畫出一個圖形後(放開左鍵後)，**該圖形的資料應該同步被顯示到資料顯示區，也就是說，繪圖區與資料顯示區是同一個 Model 的兩個 View。**

拖曳出來的圖形被同步顯示到資料顯示區，資料顯示區會記錄圖形的座標、長寬，為簡化問題，圖形內的文字請暫時使用亂數隨機產生長度介於3~10個字的數字與英文組合，並將文字顯示在圖形內(參考圖一)，另外，此次作業不要求文字的位置置於圖形正中，只要不差太多，隨意即可。後續作業會增加修改文字及調整文字位置的功能。



圖二(此為影片，請點[這裡播放](#))

四、[3 pts] 使用 DataGridView 新增流程圖元件

除了使用滑鼠繪圖，**使用者也能夠直接使用資料顯示區進行流程圖元件的繪製以及刪除**，當使用者按下新增時，建立一個新圖形，並同時顯示該圖形的資訊在資料顯示區，以及在繪圖區畫出該圖形。

五、[3 pts] Factory Pattern、繼承與多型

在此作業中，有四種流程圖元件（Start, Terminator, Process, Decision）。

當新增元件時，請使用簡單工廠（simple factory）模式來創建不同元件的 Instance。

當繼承架構下新增新的元件時，Model 底下只有 Factory class 需要更改，其他的 class 都不用變動。

在此作業中，請創建一個基礎類 Shape，並讓 Start、Terminator、Process 和 Decision 四個子類別繼承它並分別覆寫基礎類別中的方法以實現多型。

六、[5 pts] Observer pattern

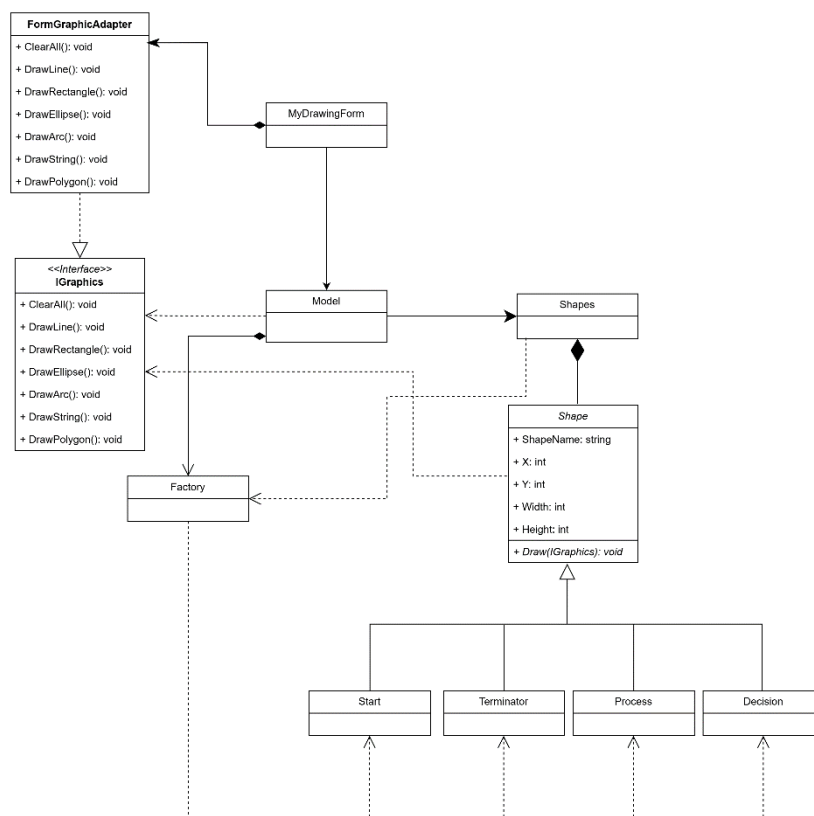
當 Model (Subject)發生變更時，應該引用 Observer pattern，以”事件”通知 View (Observer)，使得 View（繪圖區及圖形資料顯示區）能更新其畫面，並且維持 View 與 Model 的單向關係。

七、[3 pts] Adapter pattern

在本作業中，您必須對繪圖介面作抽象化，使 Model 在操作繪圖時採用標準的中性繪圖介面，而不是直接依賴 View 的繪圖物件，以避免 Model 對 View 的耦合。

八、[5 pts] Model View Controller (MVC) Pattern

你的設計將根據 MVC 架構評分，請讓 UI 盡可能薄，並必須將 UI 以及 Model 切開，強制實行單向依賴，建議遵循「圖三」的 Class Diagram 設計程式。



圖三 Class Diagram

九、[3 pts] 程式排版

請使用 Visual Studio 的自動排版，若發現有任何不整齊排版、亂空格、亂空行、亂命名，或有各種上課講到的 Smell 即扣分。

Visual Studio 2022 預設按下 ctrl + k，再按下 ctrl + d 即可自動排版。

十、[1 pts] Summary

每一次寫完作業時都必須完成 homework summary，必須填寫本次作業花費時間，請點此[連結](#)下載範本檔。

Additional Information

The following information may be useful for your homework. If you need more details, please visit MSDN website.

- 1 Cursor – 你可以使用下列程式將某個控制項的游標(Cursor)變更為十字形，或其他形狀。

Cursor = Cursors.Cross;

- 2 Checked – 你可以使用下列程式更改 Toolbar 按鈕的 Checked 屬性。

toolStripButton1.Checked = true;

- 3 流程圖元件繪製參考

3.1 Start

```
// Start
Graphics g = e.Graphics;
Pen pen = new Pen(Color.Black, 2);
g.DrawEllipse(pen, 40, 50, 50, 50);
```

3.2 Terminator

```
Graphics g = drawingPanel.CreateGraphics();
Pen pen = new Pen(Color.Black, 2);
// 繪製左半圓
g.DrawArc(pen, x, y, height, height, 90, 180);
// 繪製右半圓
g.DrawArc(pen, x + width, y, height, height, 270, 180);
// 繪製上方直線
g.DrawLine(pen, x + height / 2, y, x + width + height / 2, y);
// 繪製下方直線
g.DrawLine(pen, x + height / 2, y + height, x + width + height / 2, y + height);
```

3.3 Process

```
//process
Graphics g = e.Graphics;
Pen pen = new Pen(Color.Black, 2);
g.DrawRectangle(pen, 500, 150, 100, 50);
```

3.4 Decision

```
//decision
Graphics g = e.Graphics;
Pen pen = new Pen(Color.Black, 2);
x = 90;
y = 200;
width = 100;
height = 100;

Point[] points = new Point[4];
points[0] = new Point(x + width / 2, y); // 頂點
points[1] = new Point(x + width, y + height / 2); // 右點
points[2] = new Point(x + width / 2, y + height); // 底點
points[3] = new Point(x, y + height / 2); // 左點
g.DrawPolygon(pen, points);
```

3.5 流程圖元件中的文字

```
g.DrawString("Start text", new Font("Arial", 7), Brushes.Black, 43, 65);
```