แผนปฏิบัติการสอนสัปดาห์ที่ 4

วิชา 030143361 การโปรแกรมคอมพิวเตอร์สำหรับงานควบคุม **ระดับ:** ปริญญาตรี

เรื่อง Thread และ Timer **เวลา:** บรรยาย 120 นาที

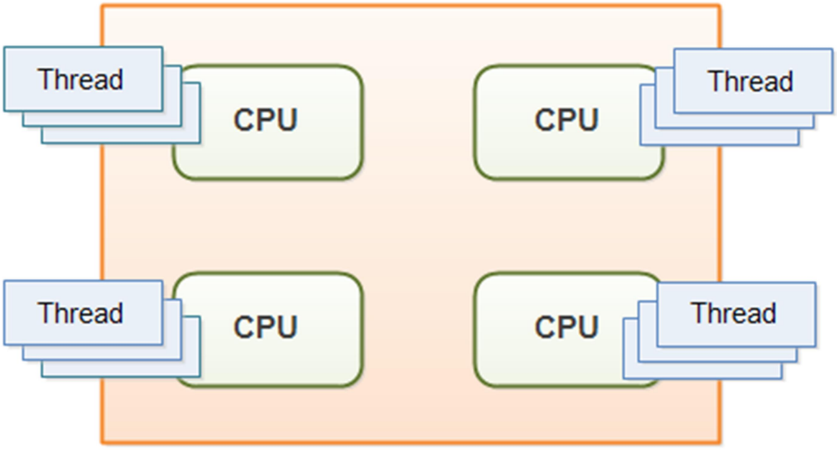
ปฏิบัติ 120 นาที

ก. วัตถุประสงค์การสอน	**รายละเอียดตามที่ระบุไว้ใน**
1. สร้าง Thread	หน้า 47-60
2. เรียกใช้งาน Thread และการทำลายทิ้ง	
3. ใช้ Timer ควบคุมจังหวะเวลา	
4. เปิด-ปิด Timer ตามคำสั่ง	
5. ทำ Debugging	

ข. การจัดการเรียนการสอน

เวลา – นาที		0 60 120 180 240
วัตถุประสงค์		1, 2, 3, 4, 5
การนำเข้าสู่บทเรียน		
ให้เนื้อหา		
สรุปเนื้อหา		
พัก		
ทำแบบฝึกหัด		
ให้เนื้อหา		
สรุปเนื้อหา		
ทดสอบและเก็บคะแนน		
ประเมินผล		พิจารณาจากผลการทดสอบในชั่วโมงสุดท้าย
วิธีการสอน:	บรรยาย	
	ถาม – ตอบ	
	ทำแบบฝึกหัด	
	บรรยาย และแสดงให้ดู	
	ทดสอบ	
สื่อการสอน:	คอมพิวเตอร์	
	Presentations	

Content



http://tutorials.jenkov.com/java-concurrency/index.html

1. สร้าง Thread

Thread คือ Process ย่อยที่สามารถ Run คำสั่ง หรือโปรแกรมที่ต้องการได้ ปัจจุบันใน 1 Process หรือ 1 Program สามารถมีได้หลาย Thread (Multithread) ยกตัวอย่างเช่น ในขณะที่เราพิมพ์งานอยู่บนโปรแกรมพิมพ์งานถ้าเราพิมพ์ผิดก็จะขึ้นตัวขีดเส้นใต้สีแดง โปรแกรมคอยตรวจสอบคำผิดที่ทำงานอยู่เบื้องหลังนี้คืออีก 1 Thread ที่ทำงานแยกออกมาจากโปรแกรมหลัก แต่จะถูกควบคุมและสั่งงานได้จากโปรแกรมหลักรวมถึงถูกทำลายไปพร้อมโปรแกรมหลักด้วย เมื่อสร้าง Thread ระบบ Operating System (OS) จะแบ่งทรัพยากร (Resources) เช่น หน่วยความจำ และ CPU ให้กับ Thread เพื่อให้ Thread สามารถ Run คำสั่งได้เหมือนโปรแกรมหนึ่ง ดังนั้นถ้า Thread มีการวนทำคำสั่งไม่รู้จบหรือ Infinity loop ก็จะไม่ส่งผลต่อโปรแกรมหลัก แต่ถ้าไม่ใช้ Thread ทำงานและมีการเข้าทำงาน Infinity loop ในโปรแกรมหลักจะทำให้โปรแกรมนั้นไม่ตอบสนองไปชั่วขณะจนกว่าจะออกจาก loop นั้นมาทำงานปกติ

การสร้าง Thread นั้นสามารถทำได้ 2 วิธีคือ In-unit Thread และ Thread Object ในตัวอย่างแรกจะแสดงวิธีการสร้าง Thread ภายใน Unit ที่เรียกใช้

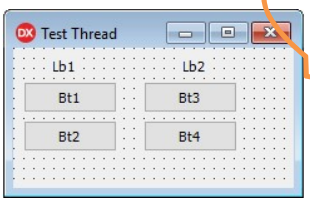
http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/delphivclwin32/Classes_TThread_Create.html

ขั้นตอนการสร้าง In-unit Thread

- *ประกาศชนิดข้อมูลที่สืบทอดมาจาก TThread เช่น TMyThread = class(TThread)*
- *ประกาศตัวแปร Thread เช่น MyThread:TMyThread;*
- *สร้าง Thread เช่น MyThread := TMyThread.create(false);*

- *เขียน Code ที่ต้องการให้ Thread Run อยู่ภายใน procedure TMyThread2.Execute;*
- *ถ้าต้องการ Synchronize กับ Procedure ใช้คำสั่ง Synchronize(ชื่อ Procedure);*
- *ทำลาย Thread เมื่อไม่ได้ใช้ MyThread.Terminate; MyThread.Free;*

Example



วาง Objects

รับค่าเป็น Object

ประกาศ Thread

```
unit UThread1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;
type
// แบบไม่มี Contructor
 TMyThread = class(TThread)
  private
   Count:Integer;
   procedure ResetCount;
  protected
   procedure Execute; override;
  end;

// แบบไม่มี Contructor
 TMyThread2 = class(TThread)
  private
   Count:Integer;
   LabelThread:TLabel;
   procedure ResetCount;
  protected
   procedure Execute; override;
   procedure UpdateCount;
  public
   constructor create(Labelx:TLabel);
   destructor Destroy; override;
  end;

type
  TFrm1 = class(TForm)
    Bt1: TButton;
    Lb1: TLabel;
    Bt2: TButton;
    Lb2: TLabel;
    Bt3: TButton;
    Bt4: TButton;
    procedure Bt1Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Bt2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Bt3Click(Sender: TObject);
    procedure Bt4Click(Sender: TObject);
  private
   MyThread:TMyThread;
   MyThread2:TMyThread2;
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  Frm1: TFrm1;

implementation

{$R *.dfm}

{ TMyThread }

procedure TMyThread.Execute;
begin
  inherited;
  Count:=0;
  repeat
   sleep(1000);
   Inc(Count);
   Frm1.Lb1.Caption:=IntToStr(Count);
  until Terminated;
end;

procedure TMyThread.ResetCount;
begin
 Count:=0
end;

procedure TFrm1.Bt1Click(Sender: TObject);
begin
 if MyThread = nil then
  MyThread := TMyThread.create(false);
MyThread.FreeOnTerminate:=True; // don't need to
cleanup after terminate
 MyThread.Priority:=tpLower;  // set the priority to lower
than normal
end;
```

Thread Run ใหม่

```
procedure TFrm1.Bt2Click(Sender: TObject);
begin
 MyThread.ResetCount;
end;

procedure TFrm1.Bt3Click(Sender: TObject);
begin
if MyThread2 = nil then
  MyThread2:=TMyThread2.create(Lb2);
  //MyThread2:=TMyThread2.create();
end;

procedure TFrm1.Bt4Click(Sender: TObject);
begin
 MyThread2.ResetCount;
end;

procedure TFrm1.FormCreate(Sender: TObject);
begin
 MyThread:=nil;
end;

procedure TFrm1.FormDestroy(Sender: TObject);
begin
  if MyThread <> nil then
  begin
   MyThread.Terminate;
   MyThread.Free;
  end;
end;

{ TMyThread2 }

constructor TMyThread2.create(Labelx: TLabel);
begin
 inherited create(false);
 FreeOnTerminate:=true; // Auto Free
```

สร้าง Thread

Free Thread

```
 LabelThread:=Labelx;     // Instant Point to Label
end;

destructor TMyThread2.Destroy;
begin
 ShowMessage('Thread2 Destroy!');
  inherited;
end;

procedure TMyThread2.Execute;
begin
  inherited;
 try
  Count:=0;
  repeat
   Application.ProcessMessages;
   Synchronize(UpdateCount);
   sleep(1000);
  until Terminated;
 except
  ShowMessage('Thread2 Error!');
 end;
end;

procedure TMyThread2.ResetCount;
begin
 Count:=0;
 LabelThread.Caption:=IntToStr(Count);
end;

procedure TMyThread2.UpdateCount;
begin
 Count:=random(200);
 LabelThread.Caption:=IntToStr(Count);
end;

end.
```

Code\week4\ Ex_Thread_1

Tip

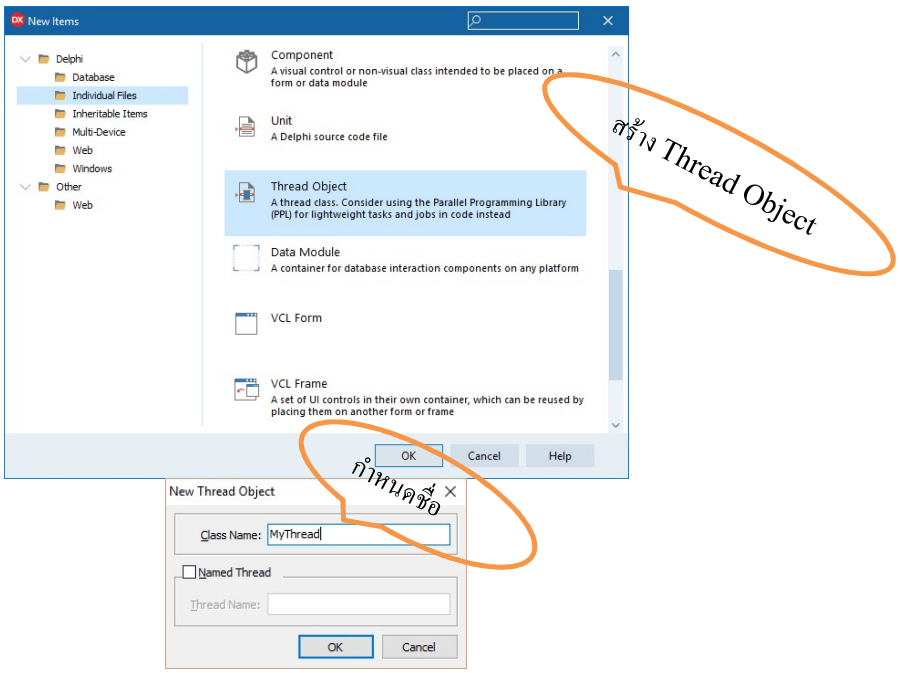
- เมื่อสร้าง Thread และใช้งานเสร็จแล้วต้อง Free ทิ้งเพื่อคืน Resources ให้ OS เสมอ

Example (การสร้าง Thread ให้ Run ด้วยคำสั่ง Resume และ Free ตัวเองกลังจากจบงาน)

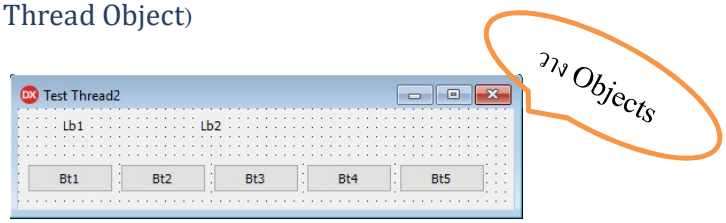
```
MyThread:=TMyThread.Create(True); // create suspended – secondprocess does not run yet
 MyThread.FreeOnTerminate:=True; // don't need to cleanup after terminate
 MyThread.Priority:=tpLower;  // set the priority to lower than normal
 MyThread.Resume(); // now run the thread
```

ขั้นตอนการสร้าง *Thread Object*

Thread Object คือการสร้าง Thread ใน Unit ที่แยกออกไปเป็นอิสระจาก Unit แม่ ทำให้สามารถเรียกใช้จาก Unit ไหนก็ได้ง่ายขึ้นอีกทั้งยังสามารถนำไป uses ใช้งานที่ Unit อื่นๆได้อย่างสะดวกสบาย การสร้าง Thread Object ทำได้โดยการเข้าที่เมนู File/New/Other../Delphi/Individual Files/Thread Object



Example (ตัวอย่างการใช้งาน Thread Object)



Tip

- ก่อนเรียกใช้งาน Unit อื่นจะต้องประกาศชื่อ Unit นั้นในส่วน uses ก่อนเสมอ

```
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, UThread_1, UThread_2;
```

Code

```
unit UMain;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
```

```
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, UThread_1,
UThread_2, Vcl.StdCtrls;

type
  TFrm1 = class(TForm)
    Bt1: TButton;
    Bt2: TButton;
    Bt3: TButton;
```

ประกาศใช้ Thread Unit

```
    Bt4: TButton;
    Bt5: TButton;
    Lb1: TLabel;
    Lb2: TLabel;
    procedure Bt1Click(Sender: TObject);
    procedure Bt2Click(Sender: TObject);
    procedure Bt3Click(Sender: TObject);
    procedure Bt4Click(Sender: TObject);
    procedure Bt5Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
     MyThread1:TMyThread1;
     MyThread2:TMyThread2;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Frm1: TFrm1;

implementation

{$R *.dfm}

procedure TFrm1.Bt1Click(Sender: TObject);
begin
 if MyThread1 = nil then
  MyThread1:=TMyThread1.create(Lb1);
end;

procedure TFrm1.Bt2Click(Sender: TObject);
begin
 if MyThread2 = nil then
  begin
   MyThread2:=TMyThread2.create(Frm1);
  end;
end;

procedure TFrm1.Bt3Click(Sender: TObject);
begin
 if MyThread2 <> nil then
  begin
    MyThread2.Terminate;
    MyThread2.Free;
    MyThread2:=nil;
  end;
end;

procedure TFrm1.Bt4Click(Sender: TObject);
var
 x:TLabel;
begin
 x:=FindComponent('lb1') as TLabel;
 if Assigned(x) then
  x.Caption:='123';
end;

procedure TFrm1.Bt5Click(Sender: TObject);
var
  p:TLabel;
begin
 if not Assigned(FindComponent('Lb3') as TLabel) then
  begin
   p := TLabel.Create(Self); // create a TLabel at run-
time
   p.Name := 'lb3'; // set a unique name
   p.Caption:='New';
   p.Top:=20;
   p.Left:=20;
   p.Visible:=True;
   p.Parent := Self;
  end;
end;
```

ค้นหา Object ตามชื่อ

สร้าง Object ตอน Run time

ส่งทั้ง Form ไปให้ Thread

```
procedure TFrm1.FormCreate(Sender: TObject);
begin
 MyThread1:=nil;

 if MyThread2<>nil then
```

```
unit UThread_1;

interface

uses
  System.Classes, Vcl.StdCtrls, System.SysUtils;

type
  TMyThread1 = class(TThread)
  private
   MyLabel:TLabel;
   procedure RandomData();
    { Private declarations }
  protected
    procedure Execute; override;
  public
    constructor create(Labelx:TLabel); //Overload;
  end;

implementation

{ MyThread }
```

```
unit UThread_2;

interface

uses
```

```
  MyThread2:=nil;

end;

end.
```

```
constructor TMyThread1.create(Labelx: TLabel);
begin
 inherited create(false);
 FreeOnTerminate:=true;
 MyLabel:=Labelx;
end;

procedure TMyThread1.Execute;
begin
 repeat
  Synchronize(RandomData);
  sleep(1000);          // every 1 seconds
 until Terminated;
end;

procedure TMyThread1.RandomData;
begin
 MyLabel.Caption:=IntToStr(random(200));
end;

end.
```

```
  System.Classes, Vcl.Forms, Vcl.StdCtrls, System.SysUtils,
Vcl.Dialogs;

type
  TMyThread2 = class(TThread)
  private
```

```
  Data1:Integer;
  FForm:TForm;
  FLabel:TLabel;

  procedure OnLabelClick(Sender: TObject);
  procedure Do_xxx;
   { Private declarations }
 protected
   procedure Execute; override;
 public
  constructor Create(MainFrom:TForm);   // Main From is
Parameter
  destructor Destroy; override;
 end;

implementation

{ MyThread2 }

constructor TMyThread2.Create(MainFrom: TForm);
begin
 inherited Create(false);
 FForm:=MainFrom;
 FLabel:=FForm.FindComponent('Lb2') as TLabel;
 FLabel.OnClick:=OnLabelClick;
end;

destructor TMyThread2.Destroy;
begin

  inherited;
end;
```

ประกาศ Event ใหม่

Map Event ใหม่

```
procedure TMyThread2.Do_xxx;
var
 x:TLabel;
begin
 FForm.Caption:=IntToStr(random(200));
 x:=FForm.FindComponent('Lb2') as TLabel;
 if Assigned(x) then
  begin
   x.Caption:=IntToStr(random(200));
  end;
end;

procedure TMyThread2.Execute;
begin
 try
  repeat
   Application.ProcessMessages;
   Synchronize(Do_xxx);
   sleep(1000);
  until Terminated;
 except
  ;
 end;
end;

procedure TMyThread2.OnLabelClick(Sender: TObject);
begin
 TLabel(Sender).Caption:='Click';
end;

end.
```

ค้นหา Object ใน Form หลัก

เปลี่ยน Property

แปลงชนิด TObject เป็น Tlabel

Code\week4\ Ex_Thread_2

Code ตัวอย่างเรื่อง Thread ด้านบนเป็นพื้นฐานในการศึกษาเรื่องการเขียนโปรแกรมแบบ OOP ได้อย่างดี สิ่งสำคัญที่น่าสังเกตุคือเราสามารถส่งค่า Parameter ไปเป็น Object ให้กับอีก Class ใช้งานได้ นั่นหมายถึงการส่ง Object ทุกตัวบน Form ไปให้ Class อื่นจัดการ Update ได้

2. เรียกใช้งาน Thread และการทำลายทิ้ง

เมื่อสร้าง Thread ด้วยคำสั่ง MyThread := TMyThread.create(false); Parameter false หมายถึงให้ Run Procedure Execute ทันที ถ้าใช้ true หมายถึงจะต้องเรียกใช้คำสั่ง Resume ถึงจะเริ่ม Run และ Thread ได้รับ Resources เสร็จแล้ว Thread จะเข้าใป Run คำสั่งภายใต้ procedure TMyThread.Execute; ถ้าต้องการให้ Thread ทำงานอะไรก็เพียงแค่ใส่คำสั่งไว้ภายใต้ procedure TMyThread2.Execute; นี้เท่านั้น การทำงานจะเป็นการทำรอบเดียวจบไม่มีการวนซ้ำ ยกเว้นมีคำสั่งให้วนซ้ำในโปแกรมที่ใส่ให้กับ Thread ในกรณีที่ต้องการให้ Thread ไปทำงาน Run Procedure ข้างนอกก็สามารถใช้คำสั่ง Synchronize(ชื่อ Procedure); ได้ หลังจากใช้งานเสร็จก็ต้องทำลายทิ้งหรือคืน Resources ให้กับ OS ด้วยคำสั่ง MyThread.Terminate; MyThread.Free; ถ้าเราสร้าง Thread ให้ทำลายตัวเองเมื่อจบการทำงาน MyThread.FreeOnTerminate:=True; มันจะถูกทำลายไปเองโดยอัตโนมัติ ตัวอย่างการใช้งานให้ดูจาก Code ด้านบน

3. ใช้ Timer ควบคุมจังหวะเวลา

การเขียนโปรแกรมให้ทำงานแบบมีจังหวะหรือมีคาบการทำงานในช่วงเวลาที่กำหนดสามารถทำได้ 2 วิธีคือ 1.ใช้การวนรอบการทำงานและหน่วงเวลารอตามเวลาที่ต้องการ 2.ใช้ Timer นับเวลา เมื่อถึงเวลาที่กำหนดก็ให้มาทำงานใน Event ที่ต้องการ

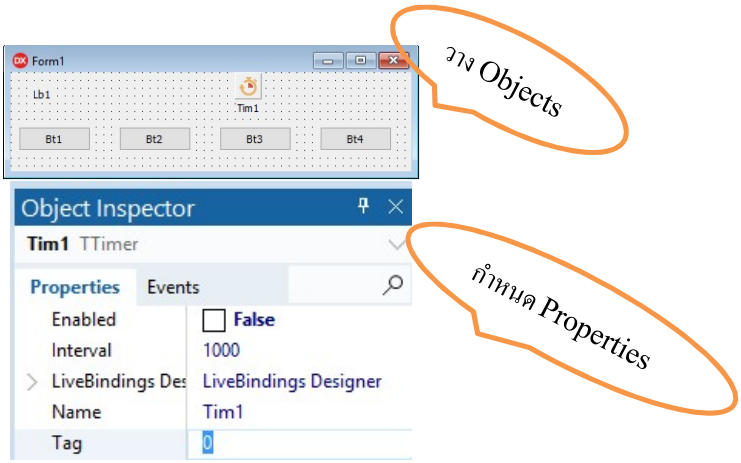
วิธีแรกไม่เป็นที่นิยมเนื่องจากการใช้คำสั่งให้หยุดรอ Sleep(เวลารอ) เป็นการทำให้โปรแกรมหยุดรอจนกว่าจะหมดเวลารอถึงจะทำคำสั่งถัดไปได้ การทำงานจึงช้าและดูเหมือนว่าโปรแกรมได้หยุดทำงานในช่วงเวลารอดังกล่าว ส่วนวิธีที่ 2. การใช้ Timer นับเวลารอเป็นที่นิยมมากกว่าเพราะในขณะที่ Timer นับเวลาโปรแกรมยังคงทำงานอื่นได้ปกติเมื่อถึงเวลาที่ตั้งไว้ก็จะเกิด Event OnTimer ผู้เขียนโปรแกรมเพียงแค่วาง Code ที่ต้องการให้ทำงานตามช่วงเวลาที่กำหนดไว้ใน procedure TForm1.Timer1Timer(Sender: TObject); เท่านั้น

<u>Tip</u>

- กรณีที่โปรแกรมติดอยู่ใน Loop ที่ต้องทำงานนานๆ ควรใส่คำสั่ง Application.ProcessMessages; ไว้เพื่อให้โปรแกรมยังคงตอบสนองต่อ Event อื่นๆได้ด้วย

4. เปิด-ปิด Timer ตามคำสั่ง

Example



Property Enable ใช้เปิด-ปิดการทำงานของ Timer ส่วน Interval เป็นช่วงเวลาให้เกิด Event มีหน่วยเป็น mS.

```
unit UTimer1;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls;

type
  TForm1 = class(TForm)
    Tim1: TTimer;
    Bt1: TButton;
    Bt2: TButton;
    Bt3: TButton;
    Lb1: TLabel;
    Bt4: TButton;
    procedure Tim1Timer(Sender: TObject);
    procedure Bt1Click(Sender: TObject);
    procedure Bt2Click(Sender: TObject);
    procedure Bt3Click(Sender: TObject);
    procedure Bt4Click(Sender: TObject);
  private
   //Count:Integer = 0;    // Can not initial.
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Count:Integer = 0;    // Only global variables may be
initialised.

implementation

{$R *.dfm}

procedure TForm1.Bt1Click(Sender: TObject);
begin
 Tim1.Enabled:=True;
end;
```

Global Var with Initial value

Start Timer

```
procedure TForm1.Bt2Click(Sender: TObject);
begin
 Count:=0;
 Lb1.Caption:=IntToStr(Count);
end;

procedure TForm1.Bt3Click(Sender: TObject);
begin
 Tim1.Enabled:=False;
 Tim1.Enabled:=True;
end;
```

Reset Counter

Reset Timer

```
procedure TForm1.Bt4Click(Sender: TObject);
begin
 Tim1.Enabled:=False;
end;

procedure TForm1.Tim1Timer(Sender: TObject);
begin
 Inc(Count);
 Lb1.Caption:=IntToStr(Count);
end;

end.
```

Stop Timer

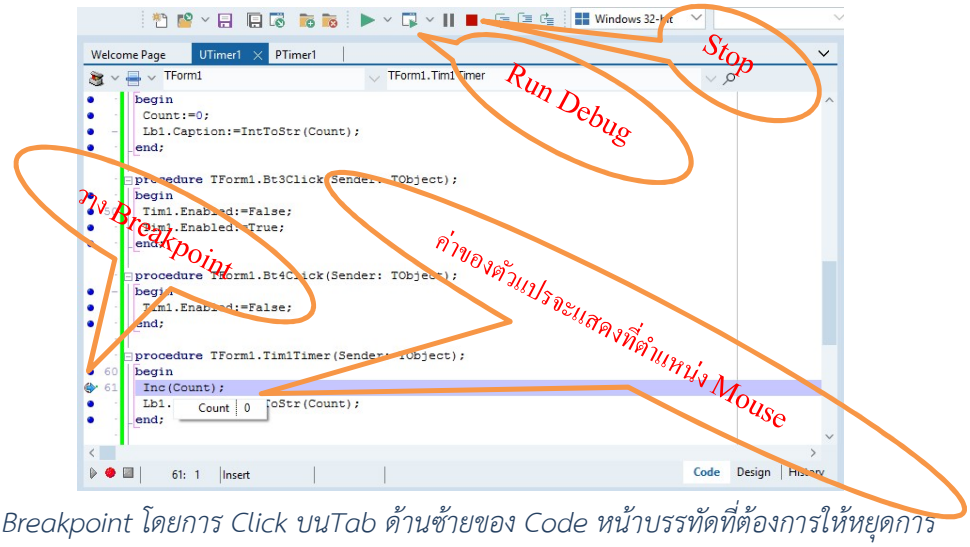
Event OnTimer

Code\week4\ Ex_Timer_1

ตัวอย่างด้านบนแสดงการใช้งาน เปิด-ปิด รีเซ็ต Timer และค่าที่ถูกนับใน Timer

5. ทำ Debugging

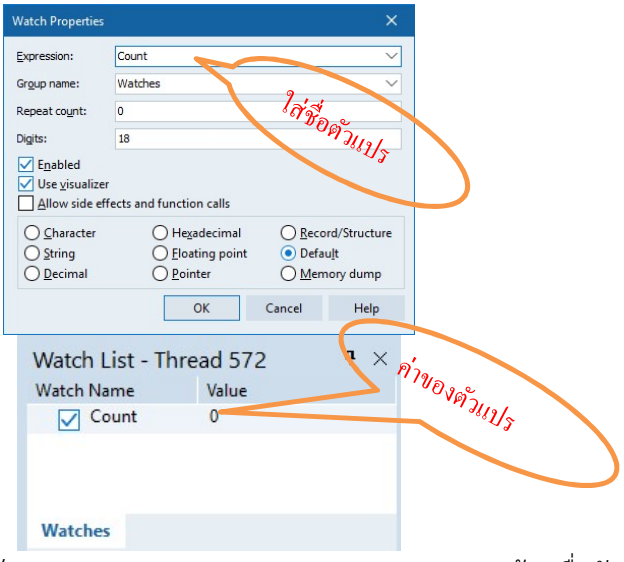
เมื่อต้องการทดสอบโปรแกรมว่ามีข้อผิดพลาดไหม หรืออยากตรวจเช็คค่าที่อยู่ในตัวแปรแต่ละตัวในขณะ Runtime เพื่อการแก้ไข ปรับปรุงหรือพัฒนาโปรแกรมสามารถทำได้โดยการ Debugging ที่มีขั้นตอนดังนี้



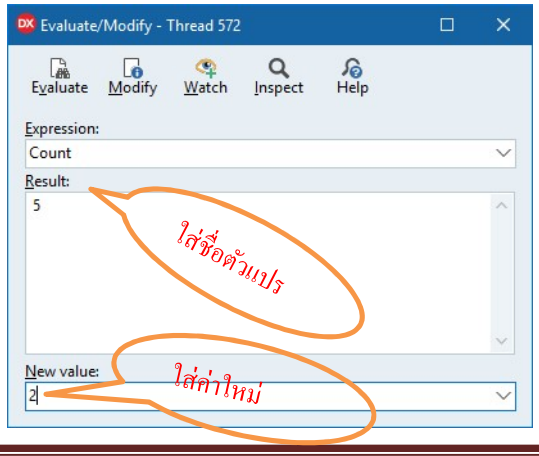
- *Add Breakpoint โดยการ Click บนTab ด้านซ้ายของ Code หน้าบรรทัดที่ต้องการให้หยุดการทำงาน*

- *กด F9 และสั่งให้ Event ที่ใส่ Break Point ไว้ทำงาน โปรแกรมจะหยุดตรงตำแหน่งที่วาง Breakpoint ไว้*
- *เอา Mouse ไปชี้ที่ตัวแปรที่ต้องการรู้ค่า ค่าของตัวแปรจะแสดงออกมา*
- *กด F9 เพื่อ Run ต่อ หรือ F8 เพื่อ Run ในบรรทัดถัดไป หรือ F7 เพื่อ Run เข้าไปใน Function ย่อย(ถ้ามี) หรือ F4 Run to Cursor*
- *กด ctrl+F2 เพื่อหยุด Run*

นอกจากจะดูค่าตัวแปรด้วยการวาง Mouse เหนือตัวแปรที่ต้องการดูค่าแล้วยังสามารถกำหนดตัวแปรที่ต้องการดูค่าได้อีกด้วยโดยเข้าไปที่เมนู Run Add Watch (ctrl+F5)



ถ้าต้องการแก้ค่าในตัวแปรให้เข้าเมนู Run/Evaluate/Modify...(ctrl+F7) ป้อนชื่อตัวแปรในช่อง Expression แล้วกด Evaluate เพื่อดูค่าปัจจุบัน ถ้าต้องการแก้ไขค่าให้ป้อนในช่อง New value แล้วกด Modify ค่าของตัวแปรนั้นจะเปลี่ยนตามค่าใหม่



Exercise

- *อธิบายเหตุผลว่าทำไมต้องใช้ Thread พร้อมยกตัวอย่าง*
- *เขียนโปรแกรมเปลี่ยนสี Form สลับสีเขียว แดง ทุกๆ 1 วินาที โดยใช้ Thread*
- *เขียนโปรแกรมเปลี่ยนสี Form สลับสีเขียว แดง ทุกๆ 1 วินาที โดยใช้ Timer*
- *เขียนโปรแกรมเปลี่ยนสี Form สลับสีเขียว แดง ทุกๆ 1 วินาที โดยใช้ Repeat Loop กับ Sleep()*

Assignment

- *เขียนโปรแกรมตรวจสอบการพิมพ์ในช่องของ TEdit โดยใช้ Thread Object เมื่อมีการกดเลข 1 ให้แสดง "Type 1" ถ้าเป็น Key อื่นให้แสดงค่า Key ที่พิมพ์ที่ Tlabel บน Form หลัก*

Answer Sheet

Code\week4\ Ans_Thread_1

Code\week4\ Ans_Timer_1

Code\week4\ Ans_Sleep_1

Code\week4\ Ans_Ass_1