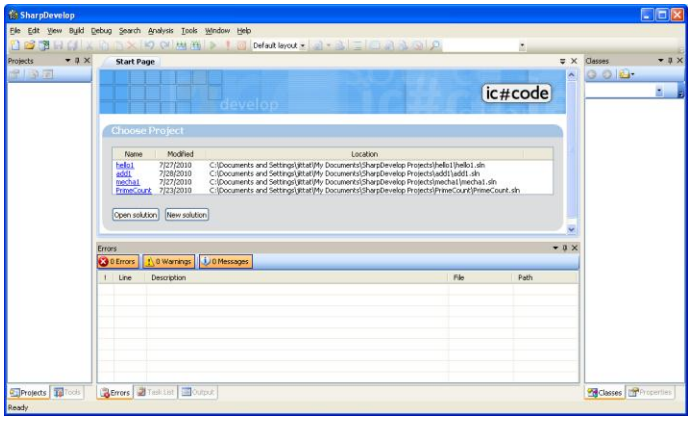
Lab 7 : แนะนำภาษา C# และสภาพแวดล้อม

ในปฏิบัติการนี้เราจะได้ทดลองศึกษาภาษา C# เบื้องต้น สำหรับคนที่เคยเขียนภาษา Python มาก่อน การเริ่มต้นโปรแกรมภาษา C# อาจจะดูมีพิธีรีตรองมากสักนิด แต่สามารถทำความคุ้นเคยได้ไม่ยากนัก สำหรับเครื่องมือที่ใช้ในการเขียนโปรแกรมนี้ได้แก่ **Sharp Develop** (http://www.icsharpcode.net/opensource/sd/)

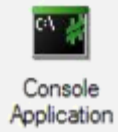
โปรแกรม Sharp Develop

เมื่อเริ่มต้นใช้โปรแกรม Sharp Develop เราจะเห็นหน้าต่างลักษณะดังรูปที่ 1



รูปที่ 1 หน้าต่าง Sharp Develop เมื่อเริ่มต้นทำงาน

ในภาษา C# จะเรียกว่า Solution ในวิชานี้เราจะได้เรียนการพัฒนาโปรแกรมสองรูปแบบคือ



1. แบบ **Console** หรือแบบที่ติดต่อกับผู้ใช้ผ่านทางตัวอักษร

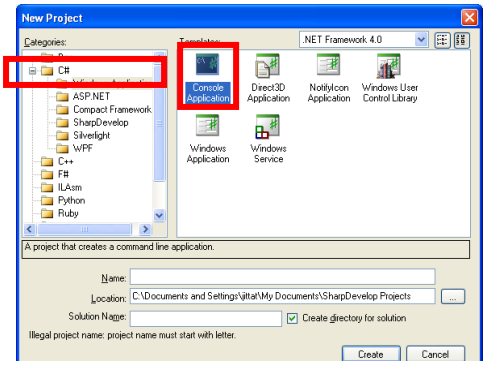
และ

2. แบบ **Windows application** หรือแบบที่ติดต่อกับผู้ใช้ด้วยระบบหน้าต่าง

ในขั้นตอนแรกนี้เราจะเริ่มพัฒนาโปรแกรมแบบ Console ก่อน

1. โปรแกรมแบบ Console

เราจะเริ่มสร้างโปรแกรมโดยเลือกเมนู File > New > Solution… หน้าต่าง New Project จะปรากฏขึ้นดังรูปที่ 2



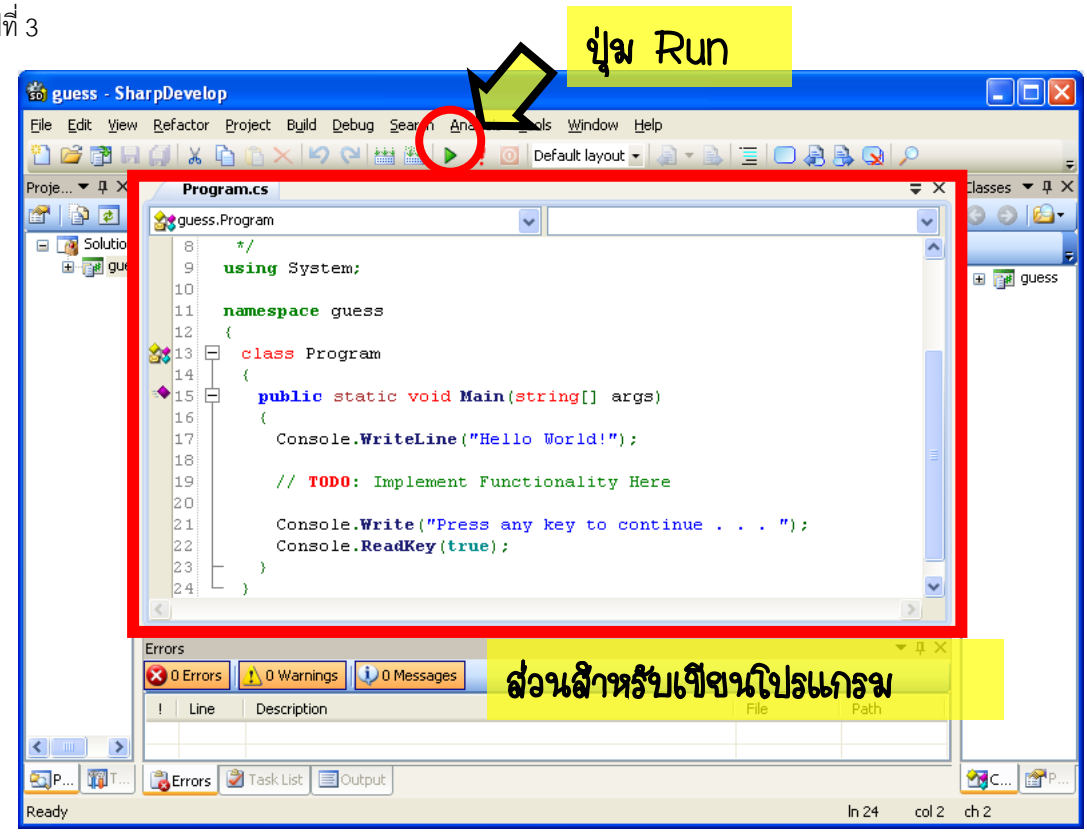
รูปที่ 2 การเลือกประเภทของ Solution

ให้ทำตามขั้นตอนดังนี้

1. ในส่วน Categories ให้คลิ๊กเลือก C# และ Windows Applications
2. ในส่วน Templates เลือก Console Application
3. ให้ตั้งชื่อ (Name) ให้กับ Solution ในที่นี้ให้ใช้ชื่อว่า mech
4. กด Create

อาจจะมีคำเตือนว่ามี Solution ชื่อนี้อยู่แล้วเพราะว่ามีการใช้งานมาก่อน หากเกิดขึ้นให้กด “Overwrite” เพื่อให้ Sharp Develop เขียนทับของเก่าเหล่านั้น

เมื่อเราสร้าง Solution เรียบร้อยแล้ว Sharp Develop จะแสดงหน้าจอที่พร้อมให้เราแก้ไขและทดลองโปรแกรม ดังรูปที่ 3



รูปที่ 3 หน้าจอสำหรับแก้ไขและตรวจสอบโปรแกรมของ Sharp Develop

เมื่อเริ่มต้น ระบบ Sharp Develop จะแสดงโครงของโปรแกรมมาให้โดยอัตโนมัติ เราสามารถเขียนหรือแก้ไขโปรแกรมได้ในช่องตรงกลางหน้าต่าง และสามารถเรียกให้โปรแกรมดังกล่าวทำงานได้โดยกดปุ่ม Run ( )

แบบฝึกหัดที่ 1 : ให้นิสิตลองกดปุ่ม Run จากนั้นบันทึกผลลัพธ์ที่ได้ว่าเกิดอะไรขึ้นอย่างไร ถ้าโปรแกรมทำงานค้างให้กดปุ่มใด ๆ บนแป้นพิมพ์

1.1 โครงสร้างของโปรแกรม

โครงของโปรแกรมที่ Sharp Develop สร้างขึ้นเป็นตัวอย่างของโปรแกรมภาษา C# โดยทั่วไป ดังรูปที่ 4

```
 1: /*
 2:  * หมายเหตุของโปรแกรม  ใช้สำหรับเขียนอธิบายทั่วไป
 3:  */
 4: using System;
 5:
 6: namespace mech
 7: {
 8:   class Program
 9:   {
10:     public static void Main(string[] args)
11:     {
12:       Console.WriteLine("Hello World!");
13:
14:       // TODO: Implement Functionality Here
15:
16:       Console.Write("Press any key to continue . . . ");
17:       Console.ReadKey(true);
18:     }
19:   }
20: }
```

ส่วนของ namespace

ส่วนของ class

ส่วนของเมธอด Main

รูปที่ 4 โครงสร้างโปรแกรมเบื้องต้น

โปรแกรมดังกล่าวแสดงให้เห็นเนมสเปส (namespace) คลาส (class) และเมธอด `Main` (ซึ่งเป็นเมธอดแรกที่จะถูกทำงานในโปรแกรมภาษา C#) ซึ่งในเบื้องต้นเราจะเริ่มต้นเขียนโปรแกรมในส่วนของ class เท่านั้น ซึ่งก็คือเมธอด Main นั้นเอง ในส่วนอื่น ๆ ให้นักเรียนคงไว้ดังเดิมโดยเราจะอธิบายในภายหลัง

แบบฝึกหัดที่ 1.1.1 : Good morning teacher.

จากโปรแกรมในรูปที่ 4 ให้นักเรียนลบคำสั่งในบรรทัดที่ 12 – 17 ออกจากนั้นแทนด้วยคำสั่งเหล่านี้

```
Console.WriteLine("Good morning teacher.");
Console.WriteLine("How are you?");
Console.WriteLine("I'm fine thank you, and you?");
Console.ReadLine();
```

จากนั้นกด Run แล้วบันทึกผลที่ได้จากการทำงานของโปรแกรม

แบบฝึกหัดที่ 1.1.2 : Good morning teacher again.

จากโปรแกรมในแบบฝึกหัด 1.1.1 ให้นักเรียนแก้ไขคำสั่ง `Console.WriteLine` เป็น `Console.Write` ผลที่ได้จากการรันโปรแกรมจะเป็นเช่นไร

จากแบบฝึกหัดที่ 1.1.1 และ 1.1.2 นักเรียนคิดว่าคำสั่ง `Console.Write` และ `Console.WriteLine` ให้ผลการทำงานแตกต่างกันอย่างไร

แบบฝึกหัดที่ 1.1.3 : Console.ReadLine()

จากโปรแกรมในแบบฝึกหัดที่ 1.1.1 และ 1.1.2 นั้นมีคำสั่ง `Console.ReadLine();` อยู่ในบรรทัดสุดท้ายของเมธอด Main ให้นักเรียนลองลบคำสั่ง `Console.ReadLine();` ออกแล้ว แล้วลอง Run โปรแกรม ผลที่ได้จากโปรแกรมเป็นเช่นไร

นักเรียนคิดว่าคำสั่ง `Console.ReadLine();` มีหน้าที่ทำงานเช่นไร

1.2 โปรแกรมราคาสินค้า

ให้นักเรียนพิจารณาโปรแกรมในภาษาไพธอนต่อไปนี้

```
1: name = input("Enter item's name: ")
2: price = int(input("Enter item's price: "))
3: pay = int(input("Enter the amount the customer pay: "))
4: print(name,"price is",price)
5: print("Customer pay", pay, ",change is", pay - price, "baht.")
```

โปรแกรมดังกล่าวเปรียบเทียบกับโปรแกรมในภาษา C# ได้ดังนี้

```
 1: using System;
 2: namespace mech
 3: {
 4:   class Program
 5:   {
 6:      public static void Main(string[] args)
 7:      {
 8:        string name;
 9:        int price, pay;
10:        Console.Write("Enter item's name: ");
11:        name = Console.ReadLine();
12:        Console.Write("Enter item's price: ");
13:        price = int.Parse(Console.ReadLine());
14:        Console.Write("Enter the amount the customer pay: ");
15:        pay = int.Parse(Console.ReadLine());
16:        Console.WriteLine("{0} price is {1}",name, price);
17:        Console.WriteLine("Customer pay {0},change is {1} baht.",pay,pay-price);
18:        Console.ReadLine();
19:      }
20:   }
21: }
```

โปรแกรมข้างต้นดังกล่าวแสดงให้เห็นความแตกต่างหลายประการระหว่างภาษา C# และภาษา Python เราจะพิจารณาส่วนย่อย ๆ ของโปรแกรมดังกล่าว ดังนี้

บรรทัด	คำสั่ง	คำอธิบาย
8 - 9	`string name;` `int price, pay;`	ในภาษา C# ตัวแปรทุกตัวจะต้องมีการประกาศก่อนการใช้งาน ตัวแปรหนึ่งตัวจะมีแบบชนิด (type) ที่คงที่ นั่นคือจะเก็บข้อมูลได้แบบชนิดเดียวตลอดการทำงาน
10	`Console.Write ("Enter item's name:");`	คำสั่ง Console.Write ทำหน้าที่แสดงข้อความออกไปหน้าจอ
11	`name = Console.ReadLine();`	Console.ReadLine() ทำหน้าที่อ่านค่าจากผู้ใช้แล้วคืนค่านั้นเป็นสตริง ซึ่งในที่นี้เรานำสตริงนั้นเก็บใส่ไว้ในตัวแปร name อีกที
12	`Console.Write ("Enter item's price: ");`	คำสั่ง Console.Write ทำหน้าที่แสดงข้อความออกไปหน้าจอเช่นเดียวกันกับคำสั่งในบรรทัดที่ 10
13	`price = int.Parse (Console.ReadLine());`	คำสั่ง Console.ReadLine() ทำหน้าที่อ่านค่าจากผู้ใช้เป็นสตริง คล้ายกับคำสั่ง input ในภาษา python ดังนั้นหากต้องการค่าเป็นตัวเลขจำนวนเต็มจึงต้องมีคำสั่งแปลงชนิดอีกทีซึ่งในที่นี้ก็คือคำสั่ง int.Parse คล้ายกับคำสั่ง int() ในภาษา pythonจากนั้นเก็บตัวเลขที่ได้ไว้ในตัวแปรชื่อ price
14 - 15	`Console.Write ("Enter customer pay: ");` `pay = int.Parse (Console.ReadLine());`	ทำงานลักษณะเดียวกับกันคำสั่งในบรรทัดที่ 12 - 13
16	`Console.WriteLine ("{0} price is {1}",name, price);`	คำสั่ง Console.WriteLine ทำหน้าที่แสดงข้อความออกทางหน้าจอ โดยจะแทนที่ข้อความในตำแหน่ง {0} และ {1} ด้วยค่าที่เก็บไว้ในตัวแปร name และ price ตามลำดับ
17	`Console.WriteLine ("Customer pay {0},change is {1} baht.",pay ,pay-price);`	คำสั่ง Console.WriteLine ค่าที่ถูกนำไปแทนใน {0} หรือ {1} ไม่จำเป็นต้องเป็นตัวแปรเสมอไป สามารถใช้นิพจน์ทางคณิตศาสตร์ (ตัวอย่างในบรรทัดนี้คือ pay – price) ต่าง ๆ แทนได้เช่นกัน โดยโปรแกรมจะคำนวณนิพจน์ทางคณิตศาสตร์นั้น ๆ จนได้ผลลัพธ์สุดท้าย เพื่อนำผลลัพธ์นั้นไปแทนค่า

แบบฝึกหัดที่ 1.2.1 : แก้ไขโปรแกรม

ให้นักเรียนทำตามแก้โปรแกรมตามข้อกำหนดต่อไปนี้ แล้วจดบันทึกข้อผิดพลาดที่คอมไพเลอร์แจ้งออกมา (ทดสอบแยกกันแต่ละข้อ)

การแก้โปรแกรม	ข้อผิดพลาดที่คอมไพเลอร์แจ้งพร้อมอธิบายสาเหตุของความผิดพลาด
1. ลบ ; ที่ท้ายบรรทัดบางบรรทัด	
2. ลบบรรทัด `string name;` ในบรรทัดที่ 8 ออก	
3. แก้ไขบรรทัดที่ 13 โดยเอาคำว่า int.Parse ออก ให้เหลือเป็น `price = (Console.ReadLine());`	

จงตอบคำถามต่อไปนี้

คำถาม	คำตอบ
1. จากคำสั่งในบรรทัดที่ 16 หากเปลี่ยน {0} และ {1} เป็น {1} และ {0} ตามลำดับ ผลลัพธ์ที่ได้เป็นเช่นไร	
2. จากคำสั่งในบรรทัดที่ 16 หากเปลี่ยนข้อความ name และ price เป็นข้อความ price และ name แทนตามลำดับ ผลลัพธ์ที่ได้เป็นเช่นไร	
3. จากคำสั่งในบรรทัดที่ 16 หากเปลี่ยน {0} และ {1} เป็น {1} และ {2} ตามลำดับ ผลลัพธ์ที่ได้เป็นเช่นไร	

1.3 เกมทายเลข: เมธอด

ในส่วนนี้เราจะทดลองเกมทายเลข ให้สร้าง solution ของโปรแกรมแบบ Console ขึ้นมาใหม่ จากนั้นเราจะเริ่มต้นโดยการเพิ่มเมธอด `RandInt` ที่เป็นเมธอดสำหรับสร้างเลขสุ่มเข้าไปในโปรแกรม เมธอดดังกล่าวจะใช้ตัวแปร `randGen` ที่เป็นตัวแปรแบบ global ในการสุ่มตัวเลข ข้อสังเกตก็คือทั้งเมธอดและตัวแปรขึ้นต้นการประกาศด้วยคำว่า `static` เราจะได้ศึกษารายละเอียดของการเขียนเมธอดและเรื่องของตัวแปร global อย่างละเอียดต่อไป สำหรับตอนนี้ให้ใช้งานไปก่อน

ให้เพิ่มการประกาศตัวแปรและ เมธอดดังกล่าวลงในคลาส `Program` แต่อยู่ภายนอกเมธอด `Main` เมธอดแสดงเป็นแถบสีเทาในรูปที่ 6

```
class Program
{

  static Random randGen = new Random();
  static int RandInt(int fr, int to)
  {
    return fr + (randGen.Next() % (to - fr + 1));
  }

  public static void Main(string[] args)
  {
    // ……
  }
}
```

รูปที่ 6 แสดงการประกาศตัวแปร randGen และเมธอด RandInt

เราจะทดลองเมธอดดังกล่าว ให้แก้เมธอด `Main` ให้เป็นดังด้านล่าง

```
  public static void Main(string[] args)
  {
    Console.WriteLine(RandInt(1,100));
    Console.WriteLine(RandInt(1,100));
    Console.WriteLine(RandInt(1,100));
    Console.ReadLine();
  }
```

ให้ทดลองเรียกให้โปรแกรมทำงาน เราจะพบว่าโปรแกรมจะพิมพ์ตัวเลขออกทาง Console ซึ่งตัวเลขเหล่านี้น่าจะแตกต่างกัน จากนั้นทดลองเปลี่ยนตัวเลขแทนที่เลข 1 และเลข 100 โดยให้ตัวเลขตัวหน้ามีค่าน้อยกว่าเสมอ แล้วตอบคำถามว่า “เมธอด RandInt() ทำหน้าที่อะไร”

1.4 เกมทายเลข: โปรแกรมหลัก

เราจะเขียนโปรแกรมทายเลขแบบง่ายในภาษา C# โปรแกรมดังกล่าวจะมีลักษณะการทำงานเช่นเดียวกับโปรแกรมที่เราเคยเขียนในภาษา Python แก้โปรแกรมหลักเป็นดังด้านล่าง จากนั้นให้ทดลองสั่งให้โปรแกรมทำงาน

```
  public static void Main(string[] args)
  {
    int s = RandInt(1,100);
    int g = -1;

    while(g != s) {
      Console.Write("Please guess: ");
      g = int.Parse(Console.ReadLine());
      if(g > s)
        Console.WriteLine("Your guess is too high.");
      if(g < s)
        Console.WriteLine("Your guess is too low.");
    }
    Console.WriteLine("You guessed correctly.");
    Console.ReadLine();
  }
```

แบบฝึกหัดที่ 1.4.1 ให้ตอบคำถามต่อไปนี้

1. เลขที่สุ่มได้มีขอบเขตเป็นเท่าใด?	
2. ให้อธิบายการทำงานของคำสั่ง while ในโปรแกรม เงื่อนไขใดที่ทำให้โปรแกรมเลิกทำงานซ้ำ (คำใบ้: คำสั่งดังกล่าวมีการทำงานเหมือนกับในภาษา Python)	
3. สมมติว่าค่าที่คืนจากเมธอด `RandInt` คือ 45 และผู้ใช้ทายว่า 70 โปรแกรมจะพิมพ์ประโยคใดออกมา	

แบบฝึกหัดที่ 1.4.2 จำนวนครั้งของการทาย

แก้โปรแกรมให้พิมพ์จำนวนครั้งของการทายออกมาเมื่อโปรแกรมทำงานเสร็จด้วย โดยเติมคำสั่งที่ต้องการลงในโปรแกรมด้านล่าง บรรทัดที่เกี่ยวข้องที่ถูกเพิ่มในโปรแกรมแสดงเป็นแถบสีเทา

```
    public static void Main(string[] args)
    {
      int s = RandInt(1,100);
      int g = -1;

      int count = 0;

      while(g != s) {

        ____________________________________________________
        Console.Write("Please guess: ");
        g = int.Parse(Console.ReadLine());
        if(g > s)
          Console.WriteLine("Your guess is too high.");
        if(g < s)
          Console.WriteLine("Your guess is too low.");
      }
      Console.WriteLine("You guessed correctly.");
      Console.WriteLine("You guessed for {0} times.”,_________________);
      Console.ReadLine();
    }
```

1.5 เกมบวกเลข

เราจะเขียนเกมฝึกบวกเลข โดยตัวอย่างของการเล่นเกมดังกล่าวสองครั้งแสดงดังรูปด้านล่าง

```
42 + 72 = ? 114
Good.  You're correct.

56 + 75 = ? 7
Sorry.  The correct answer is 131.
```

ในการเขียน ให้ตัดส่วนของเมธอด `RandInt` จากรูปที่ 6 มาใส่ในคลาสด้วย เพื่อใช้ในการสุ่มตัวเลข

แบบฝึกหัดที่ 1.5.1 ส่วนสุ่มคำถาม

ให้เติมโปรแกรมในเมธอด `Main` ที่สุ่มคำถามให้สมบูรณ์ โปรแกรมดังกล่าวจะสุ่มจำนวนเต็มระหว่าง 1 ถึง 100 มาสองจำนวน แล้วพิมพ์คำถาม ถ้าโปรแกรมถูกต้องแล้วเมื่อเรียกให้โปรแกรมทำงาน โปรแกรมจะแสดงคำถามที่สุ่มได้แล้วรอผู้ใช้กดปุ่ม Enter ก่อนจบการทำงาน

```
public static void Main(string[] args)
{

  int x = ____________________________________;

  ____________________________________________;

  Console.Write("{0} + {1} = ? ",____, y);
  Console.ReadLine();
}
```

แบบฝึกหัดที่ 1.5.2 ส่วนตรวจคำตอบ

ในส่วนตรวจคำตอบนั้น เราจะใช้คำสั่งควบคุม `if` เพื่อตรวจสอบเงื่อนไข คำสั่ง if มีรูปแบบดังนี้

```
if(  เงื่อนไข  )
    statement;
```

สิ่งที่แตกต่างจากในภาษา Python คือ คำสั่ง `if` จะควบคุมคำสั่งที่ตามมาแค่คำสั่งเดียวเท่านั้น ถ้าต้องการให้ควบคุมหลายคำสั่ง จะต้องรวมคำสั่งหลาย ๆ อันเป็นกลุ่มคำสั่งเดียว โดยใช้เครื่องหมายปีกกา ดังรูปแบบด้านล่าง นิสิตจะได้เรียนคำสั่งนี้อย่างละเอียดอีกครั้งหนึ่ง

```
if(  เงื่อนไข  ) {
    statement1;
    statement2;
        …
    statement3;
}
```

การเขียนเงื่อนไข หรือนิพจน์ตรรกศาสตร์ในภาษา C# โดยทั่วไปแล้วไม่แตกต่างจากในภาษา Python มากนัก กล่าวคือ ถ้าเราต้องการทดสอบว่า เท่ากับ, มากกว่า, หรือน้อยกว่า เราจะใช้ตัวดำเนินการ "==", ">", หรือ "<" ตามลำดับ สิ่งที่แตกต่างคือในภาษา C# ไม่มีตัวดำเนินการ `and`, `or`, หรือ `not` แต่จะใช้สัญลักษณ์แทน

ในภาษา C# การย่อหน้าไม่มีความหมายใด ๆ ต่อการคอมไพล์หรือนำโปรแกรมไปทำงาน อย่างไรก็ตาม เราควรจะย่อหน้าในลักษณะเดียวกับใน Python เพื่อทำให้โปรแกรมอ่านง่าย

ให้เขียนโปรแกรมส่วนตรวจคำตอบ โดยเติมโปรแกรมด้านล่างให้สมบูรณ์ ไม่ต้องเขียนส่วนสุ่มและแสดงคำถาม

```
public static void Main(string[] args)
{
  // ===== ส่วนสุ่มและแสดงคำถามถูกละไว้ ==========
  int ans = ____________________________

  if(_______________________)
    Console.WriteLine("Good.  You're correct.");
  else

    Console.WriteLine(____________________________________________);
}
```

2. โปรแกรมติดต่อกับผู้ใช้แบบกราฟิกส์

2.1 แนวคิดเบื้องต้น

ในการพัฒนาโปรแกรมที่ติดต่อกับผู้ใช้แบบกราฟิกส์บนระบบวินโดวส์ (หรือเรียกว่า GUI) มีรูปแบบที่แตกต่างไปจากการเขียนโปรแกรมที่เราเคยศึกษามาบ้าง โดยมากโปรแกรมที่เราศึกษามานั้นจะมีจุดเริ่มต้น และจุดสิ้นสุดเดียว นอกจากนี้ในการทำงาน ผู้ใช้จะติดต่อกับโปรแกรมในรูปแบบที่ค่อนข้างตายตัว ผิดกับในโปรแกรมที่ทำงานแบบกราฟิกส์ ที่ผู้ใช้สามารถเลือกกระทำการกับหลาย ๆ ส่วนของหน้าต่างได้อย่างเป็นอิสระมากกว่า

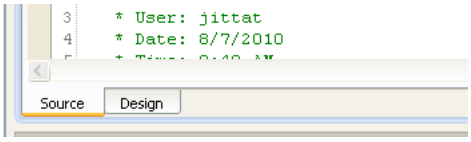
ดังนั้นรูปแบบการเขียนโปรแกรมติดต่อกับผู้ใช้แบบกราฟิกส์มักนิยมใช้การโปรแกรมแบบตอบสนองเหตุการณ์ (event-driven) ยกตัวอย่างเช่น ในการเขียนโปรแกรมกับหน้าต่างที่มีปุ่มอยู่หลายปุ่ม แทนที่เราจะเขียนโปรแกรมหลักโปรแกรมเดียว เราจะเขียนเมธอดย่อย ๆ หลาย ๆ เมธอดเพื่อทำงานกับเหตุการณ์การกดปุ่มหลาย ๆ ปุ่มเหล่านั้นแยกกัน

2.2 ทดลองสร้างโปรแกรมแรก

ในส่วนนี้เราจะทดลองโปรแกรมที่ติดต่อกับผู้ใช้ผ่านทางวินโดวส์ โปรแกรมดังกล่าวจะแสดงปุ่ม เมื่อเรากดปุ่มดังกล่าวข้อความบนหน้าจอจะแสดงคำว่า `Hello, world`

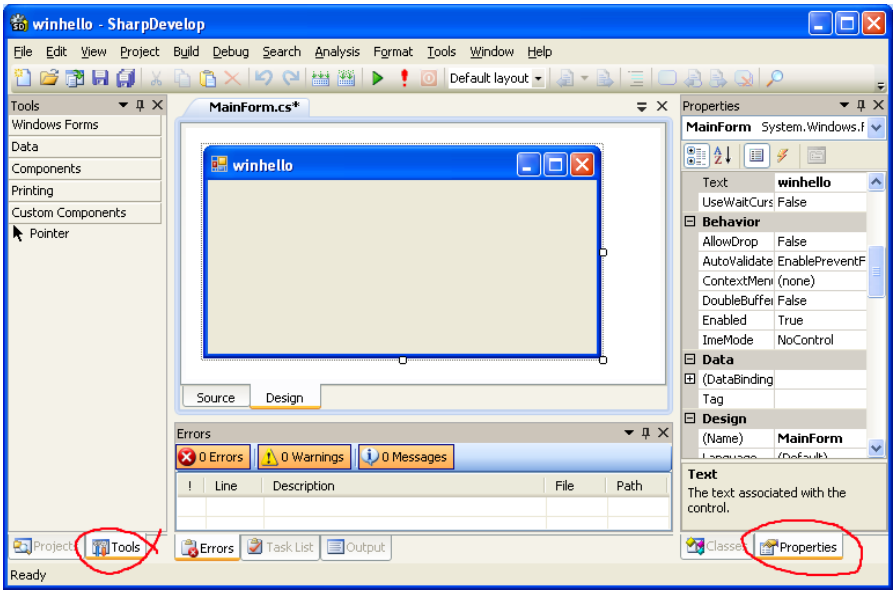
ให้การสร้าง Solution ดังกล่าว ใน Sharp Develop ให้สั่งสร้าง Solution โดยในส่วนของ Templates ให้เลือก **"Windows application"** เมื่อต้องระบุ templates สำหรับส่วนนี้ให้ใส่ชื่อ solution เป็น winhello ถ้ามี solution นี้อยู่ในเครื่องอยู่แล้วให้กด overwrite เพื่อเขียนทับ

เมื่อสร้าง solution เสร็จแล้ว หน้าจอของ Sharp Develop จะมีลักษณะไม่ต่างจากรูปที่ 3 ที่เราใช้เขียนโปรแกรมบน Console เท่าใดนัก อย่างไรก็ตามที่ด้านล่างของส่วนสำหรับแก้ไขโปรแกรม (Editor) จะมีแท็บ Design ปรากฏขึ้น ดังแสดงในรูปที่ 5 ถ้านิสิตไม่พบแท็บดังกล่าว อาจเป็นเพราะนิสิตไม่ได้สร้าง Solution ถูกต้อง (เช่น ไม่ได้เลือกประเภท Windows application) ให้กลับไปสร้าง Solution ใหม่อีกครั้ง

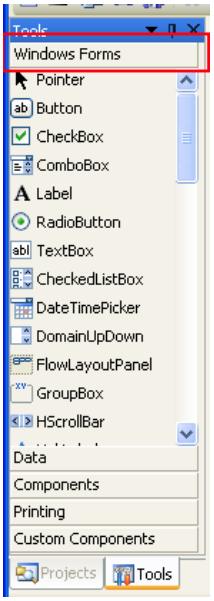


รูปที่ 5 (a) แท็บ Design, (b) หน้าจอสำหรับออกแบบ

เมื่อเรากดที่แท็บดังกล่าว เราจะเห็นหน้าต่างเปล่า ๆ ในส่วนที่เคยเป็นโปรแกรม หน้าจอนี้เป็นหน้าจอสำหรับออกแบบ เพื่อให้การทำงานสะดวก ให้กดเลือกแท็บ Tools ที่ส่วนหน้าจอข้างซ้าย และแท็บ Properties ที่ส่วนหน้าจอข้างขวาขึ้นมาด้วย ตำแหน่งของแท็บทั้งสองและหน้าจอผลลัพธ์แสดงในรูปที่ 6 (a)



(a)



(b)

รูปที่ 6 (a) แสดงหน้าจอสำหรับออกแบบหน้าต่าง วงกลมด้านซ้ายและด้านขวาแสดงตำแหน่งของแท็บ Tools และ Properties (b) แสดงแท็บ Tools หลังจากเลือก Windows Forms แล้ว จะเห็นว่ามีวัตถุที่เราสามารถเลือกใช้ได้หลายแบบ

ในส่วนตรงกลางหน้าจอ จะแสดงหน้าต่างของโปรแกรมของเรา เพื่อป้องกันความสับสนเราจะเรียกหน้าต่างนั้นว่าหน้าต่าง `winhello`

<u>แบบฝึกหัดที่ 2.2 ให้สั่งให้โปรแกรมทำงาน ผลลัพธ์ที่ได้คืออะไร</u>

ให้กดปุ่ม X ที่มุมขวาของหน้าต่างที่เราสร้างขึ้น เพื่อกลับไปยังหน้าจอเขียนโปรแกรมของ Sharp Develop

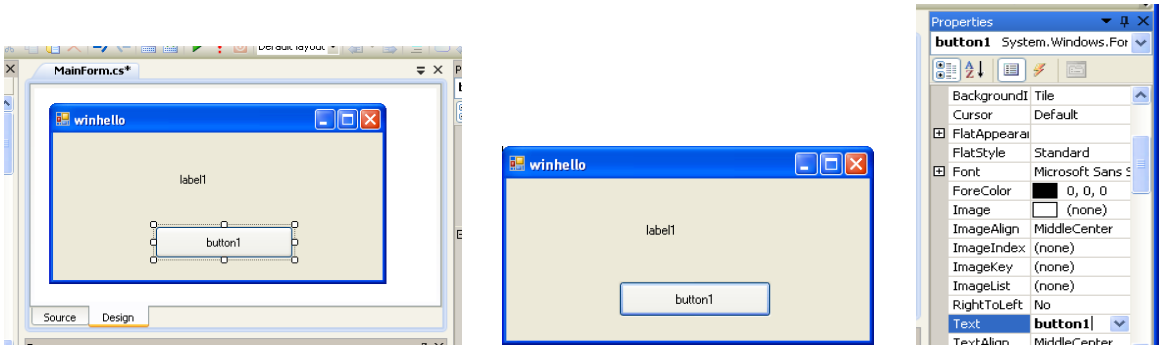
2.2.1 หน้าต่างและวัตถุ

ในหน้า Design ส่วนแท็บ Tools ที่ด้านซ้ายจะเป็นส่วนที่เราใช้เลือกวัตถุเช่น ปุ่ม ป้ายข้อความ หรือ กล่องข้อความ เพื่อนำมาวางในหน้าต่างที่เรากำลังจะออกแบบ

ในแท็บ Tools เราจะเห็นกลุ่มของวัตถุหลาย ๆ ประเภท ให้กดเลือก “Windows Forms” เราจะเห็นวัตถุพื้นฐานจำนวนมาก ดังแสดงให้รูปที่ 6 (b) เราสามารถเลือกวัตถุดังกล่าว จากนั้นนำมาวางในหน้าต่างของเราได้

เราจะเริ่มออกแบบหน้าจอแบบง่าย ๆ โดยกดเลือกวัตถุประเภท `Button` จากแท็บ Tools แล้วคลิ๊กที่ภายในหน้าต่าง `winhello` เราจะเห็นปุ่ม ปรากฏขึ้น บนปุ่มดังกล่าวจะเขียนว่า `button1` จากนั้นให้เลือกวัตถุประเภท `Label` แล้วทำเช่นเดียวกัน เราจะพบว่ามีป้ายข้อความว่า `label1` ปรากฏขึ้นที่บนหน้าต่าง `winhello` เราสามารถกดเลือกทั้งปุ่มหรือป้ายข้อความได้ เมื่อเลือกแล้วเราจะเห็นกรอบเป็นเส้นประรอบวัตถุดังกล่าว เราสามารถเลื่อนตำแหน่งของวัตถุที่เราเลือกไว้ให้เรียงตัวสวยงามได้ ตัวอย่างของหน้าต่าง `winhello` หลังขั้นตอนนี้แสดงดังรูปที่ 7 (a)

เมื่อออกแบบหน้าจอเสร็จแล้ว เราสามารถสั่งให้โปรแกรมทำงานได้ทันที โดยกดปุ่ม Run (สามเหลี่ยม play สีเขียว) รูปที่ 7 (b) แสดงหน้าต่างเมื่อโปรแกรมทำงานแล้ว ก่อนจะทำงานในขั้นตอนถัดไป ให้ปิดโปรแกรมที่ทำงานอยู่เสียก่อน ไม่เช่นนั้นเราจะไม่สามารถแก้ไขโปรแกรมของเราได้



(a) (b) (c)

รูปที่ 7 (a) แสดงหน้าต่าง `winhello` ใน Sharp Develop สังเกตว่าในรูปปุ่ม `button1` ถูกเลือกอยู่
(b) ตัวอย่างหน้าจอ `winhello` เมื่อเราสั่งให้โปรแกรมทำงาน (c) คุณสมบัติของปุ่ม `button1`

2.2.2 **คุณสมบัติ** (Property)

วัตถุต่าง ๆ บนหน้าต่าง เช่น ป้ายข้อความ ปุ่ม หรือกล่องข้อความ ล้วนมีคุณสมบัติที่เราสามารถปรับเปลี่ยนได้ เช่น ตำแหน่ง, สี, ข้อความที่แสดง เราสามารถปรับแต่งวัตถุบนหน้าจอ `winhello` ที่เราออกแบบอยู่ได้ โดยเปลี่ยนแปลงคุณสมบัติ (property) บางอย่างของวัตถุเหล่านั้น เมื่อเราเลือกวัตถุหนึ่ง ๆ รายการคุณสมบัติของวัตถุนั้นจะถูกแสดงบนแท็บ Properties

ให้คลิ๊กเลือก `button1` คุณสมบัติของ `button1` จะถูกแสดงดังรูปที่ 7 (c) ให้ทดลองเปลี่ยน คุณสมบัติ `Text` ให้เป็น `Hello` เราจะเห็นว่าข้อความ `button1` บนปุ่มจะถูกเปลี่ยนเป็น `Hello` ตามที่เรากำหนดลงไป เราสามารถปรับแต่งรายละเอียดอื่น ๆ ของวัตถุได้อีก เช่น เปลี่ยนสี หรือเปลี่ยนรูปแบบฟอนต์ ให้ทดลองปรับแต่งปุ่มให้มีขนาดใหญ่ขึ้นและมีสีตามต้องการ **ระวังอย่าปรับเปลี่ยนคุณสมบัติ Name** (แสดงในแท็บ Properties อยู่ในส่วน Design เป็น (Name))

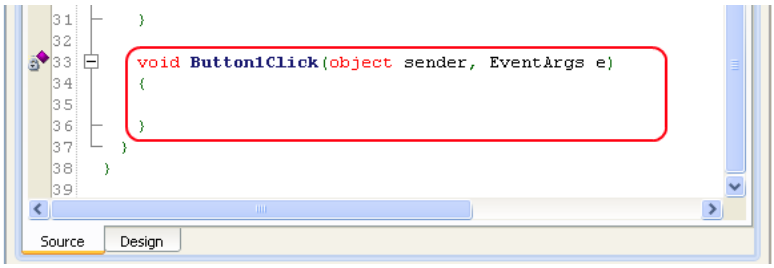
<u>แบบฝึกหัดที่ 2.2.2 ทดลองสั่งให้โปรแกรมทำงาน บันทึกผลที่ได้</u>

สังเกตว่าหน้าต่างที่แสดงจะมีลักษณะที่เปลี่ยนไปตามที่เราออกแบบ อย่างไรก็ตามยังไม่มีอะไรเกิดขึ้นเมื่อเรากดปุ่มบนหน้าต่าง ทั้งนี้เนื่องจากเรายังไม่ได้เขียนโปรแกรมเพื่อตอบสนองเหตุการณ์เหล่านั้น เราจะเขียนส่วนดังกล่าวในขั้นตอนถัดไป อย่าลืมปิดโปรแกรมที่ทำงานอยู่นี้ก่อนด้วย

2.2.3 โปรแกรมสวัสดี

เราจะเพิ่มโปรแกรมเพื่อตอบสนองการกดปุ่ม `button1` เราสามารถเริ่มเขียนโปรแกรมกับเหตุการณ์ดังกล่าวได้โดยกดดับเบิลคลิ๊กที่ `button1` ในแท็บ Design เมื่อกดแล้ว หน้าจอของ Sharp Develop จะเปลี่ยนมาแสดงผลเป็นส่วนแก้ไขโปรแกรม (Editor) เหมือนตอนเราเขียนโปรแกรมกับ Console สังเกตว่าแท็บด้านล่าง ถูกเลือกอยู่ที่ Source ไม่ใช่ Design เราจะเลือกแท็บนี้เมื่อต้องการแก้ไขโปรแกรม

เมื่อเราดับเบิลคลิ๊กที่ปุ่ม Sharp Develop จะสร้างเมธอดใหม่ ชื่อ `Button1Click` เพื่อให้เราเขียนโปรแกรมเพื่อตอบสนองกับเหตุการณ์มาตรฐานของปุ่ม (ซึ่งคือการถูกคลิ๊ก) ให้กับเรา โครงของเมธอดดังกล่าวแสดงในรูปที่ 8



รูปที่ 8 แสดงโครงของเมธอด Button1Click ที่ระบบสร้างให้

เมื่อมีการกดปุ่ม เราจะสั่งให้ label1 เปลี่ยนข้อความที่แสดงเป็น Hello, world ดังนั้นเราจะแก้เมธอด Button1Click ให้เป็นดังด้านล่าง

```
void Button1Click(object sender, EventArgs e)
{
  label1.Text = "Hello, world";
}
```

ให้ทดลองสั่งให้โปรแกรมทำงาน ทดลองกดปุ่มแล้วดูผลลัพธ์

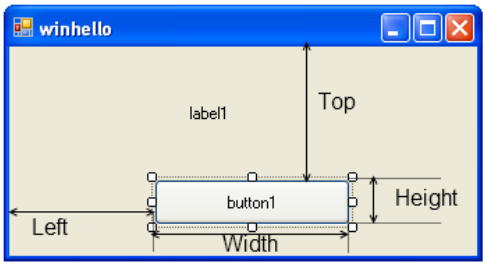
<u>แบบฝึกหัดที่ 2.2.3 ถ้าต้องการให้โปรแกรมแสดงข้อความว่า Good-bye จะต้องแก้โปรแกรมที่ส่วนใด? อย่างไร?</u>

2.3 เกมกดปุ่ม

ในส่วนนี้เราจะเขียนเกมกดปุ่ม โดยปุ่มจะย้ายที่หนีแบบสุ่มหลังจากเรากดแล้ว เป้าหมายคือกดให้ได้จำนวนครั้งมากที่สุด

2.3.1 ตำแหน่งบนหน้าจอ

วัตถุทุกชิ้นในหน้าต่างที่เราออกแบบ จะมีคุณสมบัติที่ใช้กำหนดตำแหน่งและขนาด คือ `Top`, `Left`, `Width`, และ `Height` ดังมีความหมายแสดงตามรูปด้านล่าง



รูปที่ 9 แสดงคุณสมบัติที่กำหนดตำแหน่งและขนาดของวัตถุบนหน้าต่าง

เราจะเริ่มต้นทดลองโดยการ แก้เมธอด `Button1Click` ให้เป็นดังด้านล่าง

```
void Button1Click(object sender, EventArgs e)
{
  button1.Left += 10;
}
```

<u>แบบฝึกหัด 2.3.1 ให้ทดลองโปรแกรมดังกล่าว จากนั้นตอบคำถามเหล่านี้ก่อนทำงานต่อไป</u>

1. โปรแกรมข้างต้นทำงานอะไร?	
2. ถ้าเราต้องการให้ปุ่มเลื่อนขึ้น จะต้องแก้คำสั่งเป็นอย่างไร?	

2.3.2 ปุ่มย้ายที่

ให้คัดลอกเมธอด `RandInt` และส่วนประกาศตัวแปร `randGen` จากรูปที่ 6 มาเพิ่มในคลาส `MainForm` (นำไปวางไว้ก่อน เมธอด `Button1Click`) จากนั้นให้แก้เมธอด `Button1Click` ให้สุ่มเปลี่ยนตำแหน่งของปุ่ม ทั้งในแนวแกน x และ y ให้สุ่มให้ค่า `Top` และ `Left` อยู่ระหว่าง 0 ถึง 100 (คำใบ้ อย่าลืมเมธอด `RandInt`)

<u>แบบฝึกหัด 2.3.2 ให้เขียนเมธอด Button1Click ลงที่นี่</u>

```
void Button1Click(object sender, EventArgs e)
{




}
```

2.3.3 ตัวนับ

เราจะนับจำนวนครั้งที่ผู้ใช้คลิ๊กถูกต้อง ก่อนอื่นให้ประกาศตัวแปร `clickCount` ที่ใช้เก็บจำนวนครั้งที่เรากดปุ่มได้ก่อน โดยให้ประกาศไว้ภายนอกเมธอด `Button1Click` แต่อยู่ภายในคลาส `MainForm` ดังแสดงในรูปที่ 10 สังเกตว่าเรากำหนดให้ค่าเริ่มต้นของตัวแปรดังกล่าวเป็นศูนย์

```
namespace winhello
{
  public partial class MainForm : Form
  {
    //……
    //……

    static int clickCount = 0;

    void Button1Click(object sender, EventArgs e)
    {
      // ……
    }
  }
}
```

รูปที่ 10 การประกาศตัวแปร clickCount

เพิ่มบรรทัดด้านล่างที่แสดงค่าของตัวแปร clickCount บนปุ่ม button1 ลงไปท้ายเมธอด Button1Click

```
      button1.Text = clickCount.ToString() + " click(s)";
```

จากนั้นทดลองเรียกให้โปรแกรมทำงาน เราจะเห็นข้อความว่า 0 click(s) แสดงบนปุ่มเมื่อเราคลิกโดนปุ่ม

<u>แบบฝึกหัดที่ 2.3.3 ให้ตอบคำถามต่อไปนี้</u>

1. ให้ทดลองลบคำว่า ".ToString()" ออกจากบรรทัดข้างต้น จากนั้นสั่งให้โปรแกรมทำงาน ข้อผิดพลาดที่พบคืออะไร? เพราะเหตุใด?	
2. ถ้าเราต้องการเพิ่มจำนวนครั้งที่คลิ๊กโดนปุ่ม เราจะต้องเพิ่มคำสั่งใดเข้าไปในเมธอด Button1Click	