

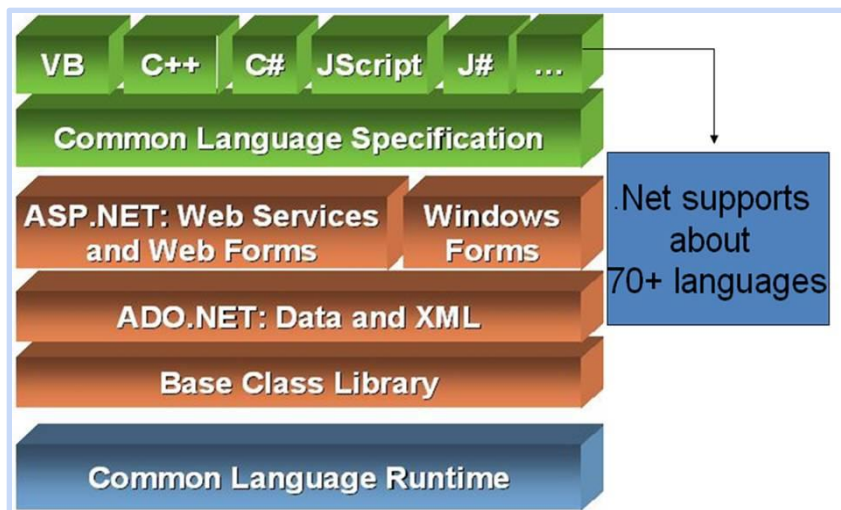
การทดลองที่ 3

การใช้งาน git ร่วมกับ Visual studio IDE

แนะนำ Dot NET Framework

Dot NET Framework เป็น Framework ที่พัฒนามาเพื่อรองรับการสร้างซอฟต์แวร์บน platform ต่างๆ เช่น แอปพลิเคชันบนระบบปฏิบัติการวินโดวส์ (Windows applications) ระบบปฏิบัติการโทรศัพท์เคลื่อนที่ (Mobile applications) โปรแกรมประยุกต์สำหรับเว็บ (Web applications) คอมโพเนนท์ (Components) และ XML Web Services โดย .NET Framework จะอยู่บนสถาปัตยกรรมที่แยกชั้นจาก Kernel ของระบบปฏิบัติการอย่างชัดเจน เป็นผลให้สามารถนำ .NET Framework ไปติดตั้งบนระบบปฏิบัติการได้หลากหลาย

สถาปัตยกรรมของ .NET



รูปที่ 1 สถาปัตยกรรมของ .NET Framework

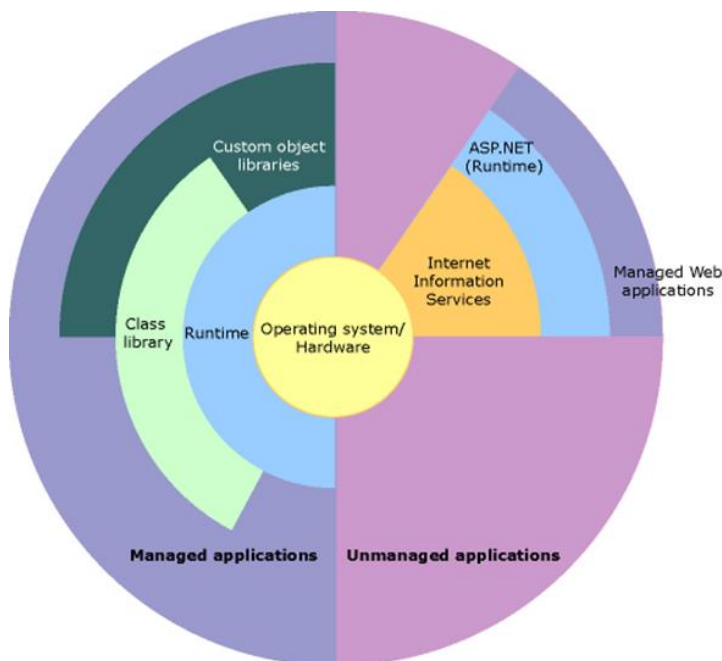
.Net Framework¹ (อ่านว่า dot net framework) เป็น platform หนึ่ง
ที่เตรียมเครื่องมือและเทคโนโลยีที่ช่วยให้เราสามารถพัฒนาแอปพลิเคชัน ที่สามารถทำงานบนระบบปฏิบัติการต่างๆ

¹ "Components of .Net Framework, CLR, CTS, CLS ... - DeveloperIn.Net."
<http://www.developerin.net/a/39-Intro-to-.Net-FrameWork/23-Components-of-.Net-FrameWork>. Accessed 23 Aug. 2017.

(ได้แทบทุกระบบ) ซึ่ง .Net Framework จะประกอบด้วยองค์ประกอบหลักสองอย่างได้แก่ Common Language Runtime (CLR) และ .Net Framework Class Library.

Common Language Runtime (CLR)

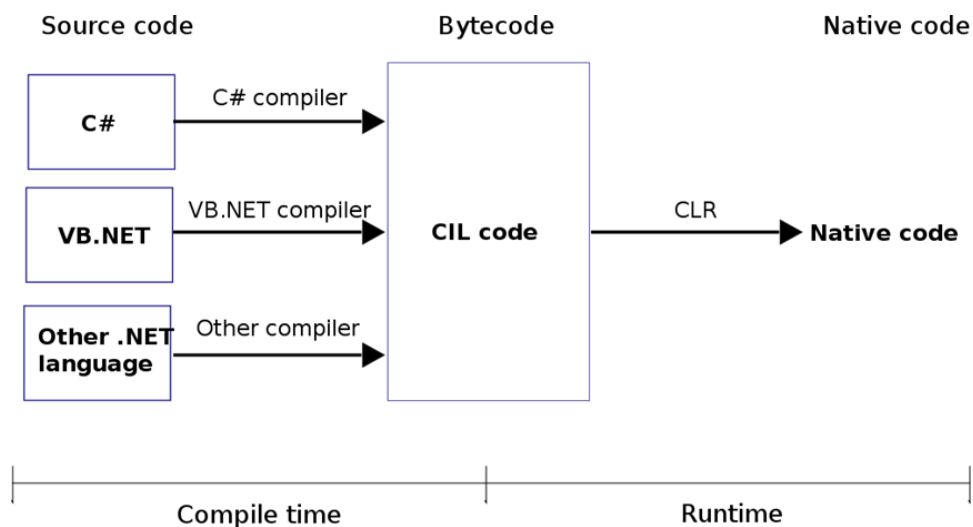
ใน .Net Framework จะมี Common Language Runtime (CLR) ทำหน้าที่จัดการสภาพแวดล้อมสำหรับรันโปรแกรมต่างๆ บน .NET โดย code ที่ทำงานภายใต้ CLR เรียกว่า Managed Code ทำหน้าที่คอยจัดการหน่วยความจำ (memory management) และจัดการเธรด (thread management) ดังนั้นโปรแกรมต่างๆ จึงไม่จำเป็นต้องกังวลเรื่องการจัดการหน่วยความจำและเรื่องการเขียนโปรแกรมแบบ multi threading ใน .NET framework จะมี CLR คอยทำหน้าที่ร้องขอพื้นที่หน่วยความจำของระบบตามที่โปรแกรมที่เราพัฒนาขึ้นมาต้องการจะใช้ หลังจากที่โปรแกรมของเราเลิกทำงานไปแล้ว CLR จะคืนหน่วยความจำที่ขอมานั้นกลับคืนแก่ระบบ หากไม่มี CLR คอยจัดการหน่วยความจำ (รวมทั้งทรัพยากรอื่นๆ เช่น network, ports, files, I/O, ฯลฯ) นักพัฒนาก็ต้องเป็นผู้ดำเนินการเอง และถ้าไม่มีการจัดการอย่างเหมาะสม จะทำให้ระบบสูญเสียทรัพยากร (จากการยืมไปใช้แล้วไม่ส่งคืน) จนระบบมีทรัพยากรเหลือน้อยเกินกว่าที่จะดำรงอยู่ต่อไปได้และหยุดทำงานลงในที่สุด ข้อดีอีกประการหนึ่งของ CLR คือจะเข้ามาทำงานแทนในส่วนที่ทุกๆ โปรแกรมต้องดำเนินการกับทรัพยากรของระบบ ทำให้นักพัฒนาประหยัดเวลาที่จะต้องมาเขียนโปรแกรมในส่วนนี้ซ้ำๆ กันในทุกโครงการซอฟต์แวร์



รูปที่ 2 Common Language Runtime

โปรแกรมที่เราเขียนขึ้น (ไม่ว่าจะเป็นภาษา C#, VB.Net, J# หรืออื่นๆ) จะถูกตัวคอมไพเลอร์แปลเป็นภาษากลางสำหรับ .NET เรียกว่า Microsoft Intermediate Language (MSIL) ซึ่งในที่สุดจะถูกแปลเป็น Native Code โดย CLR เพื่อทำงานบนระบบปฏิบัติการ ดังรูปที่ 3

ในปัจจุบัน มีภาษาที่สามารถแปลเป็น MSIL ได้หลายสิบภาษา ทั้งภาษาที่ถูกสร้างโดย Microsoft เองและโดยบริษัทอื่นๆ ซึ่งสามารถนำมารันภายใต้ CLR ร่วมกันได้อย่างราบรื่น การเปิดโอกาสให้มีการใช้ระบบร่วมกันของ code ในภาษาต่างๆ จะช่วยลดความยากลำบากในการสร้างทีมพัฒนาซอฟต์แวร์ที่จำเป็นต้องใช้ภาษาเดียวกันทั้งหมด เพราะบางภาษาอาจจะมีนักเขียนโปรแกรมเป็นจำนวนน้อยมากแต่อาจจะยังคงมีความจำเป็น เนื่องจากภาษานั้นสามารถให้ executable unit ที่กระชับ ทำงานรวดเร็ว มีประสิทธิภาพ และประหยัดทรัพยากรของระบบ



รูปที่ 3 กระบวนการคอมไพล์โปรแกรมภาษาต่างๆ ใน .NET Framework

.Net Framework Class Library (FCL)

ไลบรารี FCL นี้ บางครั้งจะถูกเรียกว่า Base Class Library ทำหน้าที่เป็นพื้นฐานคลาสสำหรับทุกๆ ชนิดข้อมูล (data type) ใน .NET ทุกภาษาโปรแกรมบน .NET จะสามารถเรียกใช้งาน Class Library นี้ได้ อาทิเช่น โปรแกรมในภาษา VB.NET จะเรียกใช้ FCL เช่นเดียวกับ C# และทุกๆ ภาษาในตระกูล .NET แอปพลิเคชันที่สามารถใช้ .net class library ตัวเดียวกันได้แก่ Windows Application, Console Application, Web Application, XML Web Services และ Windows Services.

ในทางปฏิบัติแล้ว สิ่งที่นักพัฒนาต้องทำ ก็มีเพียงแค่การนำเข้า (import) BCL ไปยังโปรแกรมโค้ดของตัวเองและเรียกใช้เมธอดที่สามารถทำงานอันซับซ้อนซึ่งอยู่ภายในไลบรารี เช่น การอ่าน-เขียนแฟ้มข้อมูล

การวาดกราฟฟิกส์ การเชื่อมต่อและทำงานกับฐานข้อมูลหรือแฟ้ม XML โดยที่ทุกๆ ภาษาใน .NET จะใช้ได้ด้วยกัน ทำให้ไม่มีปัญหาเรื่องความเร็วของโปรแกรมในขณะรัน แม้จะเขียนด้วยภาษาที่ง่ายๆ ก็ตาม

นอกจากที่กล่าวมาแล้ว ใน .Net framework ยังมีส่วนประกอบที่น่าสนใจอีก 2 อย่างคือ Common Type System (CTS) และ Common Language Specification (CLS)

Common Type System (CTS)

โดยทั่วไป ภาษาโปรแกรมต่างๆ มักจะมีชนิดข้อมูลมาตรฐานของตนเองอยู่แล้ว เมื่อนำภาษาเหล่านั้นมาคอมไพล์เป็นภาษากลางใน Common Language Runtime ก็พบว่ามีความแตกต่างระหว่างชนิดข้อมูลของภาษานั้นๆ กับชนิดข้อมูลกลาง (Common type system) แต่ยังคงมีความเข้ากันได้ข้อมูลขนาดต่างๆ เช่น 1, 2, 4, 8 ไบต์ เป็นต้น จึงต้องมีการเทียบ (mapping) ชนิดข้อมูลของแต่ละภาษาไปยังชนิดข้อมูลกลางของ .NET framework เพื่อให้โปรแกรมที่ถูกแปลมาจากภาษาเหล่านั้นสามารถใช้งานข้อมูลร่วมกันได้อย่างราบรื่น ไม่มีการสูญเสียส่วนสำคัญของข้อมูล เพื่อให้เข้าใจการ mapping ให้ดูตัวอย่างจากลิงค์ต่อไปนี้ ประกอบ

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/built-in-types-table>

แบบฝึกหัดเสริมความเข้าใจ

ให้นักศึกษา ค้นคว้าชนิดข้อมูลในภาษาอื่นๆ ที่สามารถทำงานบน .NET ได้ เช่น VB.NET ,J#, F#, ฯลฯ พร้อมทั้งตารางสำหรับเทียบกับ common type ใน .NET

ระบบชนิดข้อมูลกลาง จะรองรับการใช้งานชนิดข้อมูลพื้นฐานสองอย่างคือ Value types และ Reference types

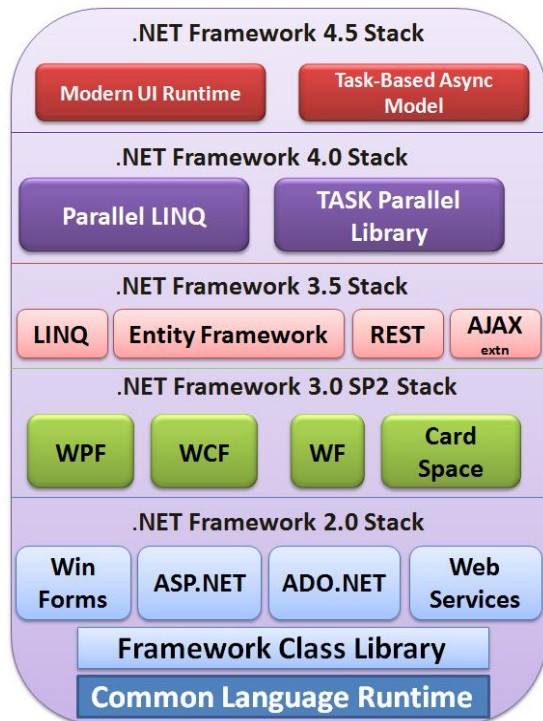
Value types:

Value types เป็นชนิดข้อมูลที่บรรจุค่าต่างๆ ไว้ในตัวเอง (เช่น ธนบัตรหรือเหรียญกษาปณ์ ซึ่งมีมูลค่าตามตัวเลขบนธนบัตรหรือเหรียญนั้น ๆ) โปรแกรมจะเก็บวัตถุชนิด Value types นี้ไว้ใน stack ของโปรแกรมหรือไม่ก็เก็บไว้ใน structure ขนาดเล็ก ๆ โดยที่ value types สามารถเป็นได้ทั้งแบบ built-in (กำหนดมาพร้อมกับการสร้างภาษา), แบบผู้ใช้กำหนดเอง (user-defined) หรือแบบ enumerations

Reference types:

Reference types เก็บค่าตำแหน่งที่อยู่ของวัตถุที่อ้างถึง โดยวัตถุนั้นต้องถูกสร้างขึ้นโดยวิธีการจองหน่วยความจำจากระบบปฏิบัติการ เช่นคำสั่ง new เป็นผลให้วัตถุนั้นถูกเก็บอยู่ใน heap แล้วนำมาผูกไว้กับตัวแปรอีกทอดหนึ่ง โดย Reference types นี้สามารถเป็นได้ทั้งชนิด self-describing types, pointer types หรือ interface types ทั้งนี้ self-describing types อาจหมายถึง arrays หรือ class types ก็ได้

Common Language Specification (CLS)



รูปที่ 4 สถาปัตยกรรม .NET ตั้งแต่รุ่น 2.0 ถึง 4.5

CLS จะเป็นสมาชิกของ Common Type System และเป็นเหมือนกฎเกณฑ์ที่คอยกำหนดให้ทุกๆ ภาษาโปรแกรมใน .NET ต้องปฏิบัติตาม เพื่อให้ใช้งานข้ามภาษาได้ทั้งที่เป็นแบบ cross language inheritance และ cross language debugging

[รายละเอียดของ .NET Framework](#)

[สามารถหาอ่านเพิ่มเติมได้จากแหล่งอ้างอิง](#)

แนะนำ IDE

IDE ย่อมาจาก Integrated Development Environment เป็นซอฟต์แวร์ประยุกต์ที่รวบรวมเครื่องมือและทรัพยากรต่าง ๆ เพื่อช่วยในการพัฒนาซอฟต์แวร์ เริ่มตั้งแต่การ coding, debugging และ deploying ซอฟต์แวร์ IDE ส่วนมากจะช่วยงานง่าย ๆ นับตั้งแต่ การตรวจสอบคำผิด เติมเต็มคำอัตโนมัติ เพื่อช่วยลดเวลาในการพิมพ์ code, ช่วยในการค้นหาและแก้ไข code ไปจนถึงการคอมไพล์ ดีบั๊ก หรือช่วยออกแบบส่วน user interface เป็นต้น (IDE จะต่างจาก text editor ตรงที่ IDE จะรวมเอาชุด compiler สำหรับการแปลภาษา และ debugger สำหรับการค้นหาและแก้ไขข้อผิดพลาดของโปรแกรมอีกด้วย)

IDE ที่ใช้งานในการทดลองนี้ เป็น IDE ที่ชื่อว่า Microsoft Visual Studio 2017 รุ่น Community edition เป็น IDE ที่ครอบคลุมการแก้ไข code ไปจนช่วยงานออกแบบ UI ตามที่เราต้องการ

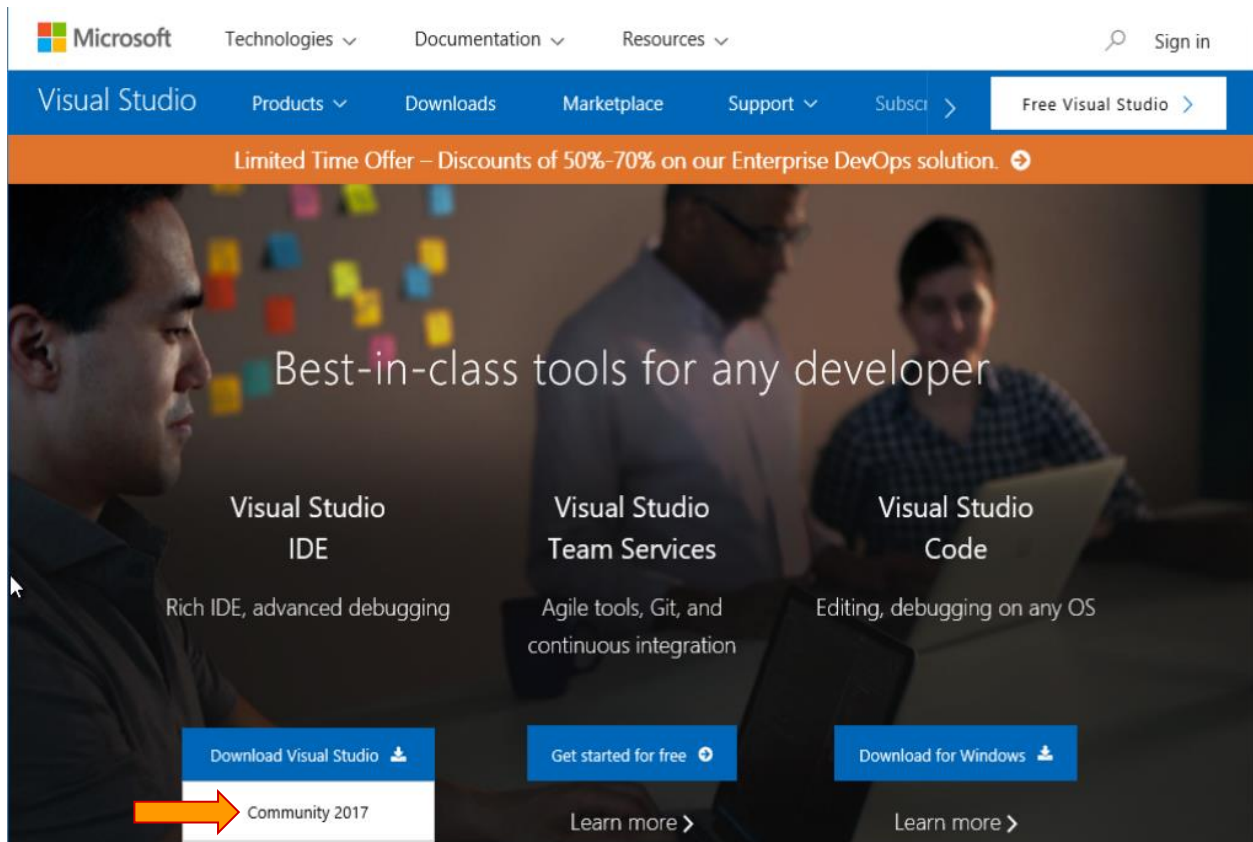
นอกจาก IDE ที่ใช้ในวิชาเรียนนี้แล้ว ในการพัฒนาโปรแกรม เราสามารถใช้ text editor ที่มีความสามารถสูง เช่น visual studio code, sublime หรืออื่นๆ นักศึกษาสามารถศึกษาเพิ่มเติมได้จากแหล่งค้นคว้าที่ระบุในเอกสารอ้างอิง

การเตรียมการทดลอง

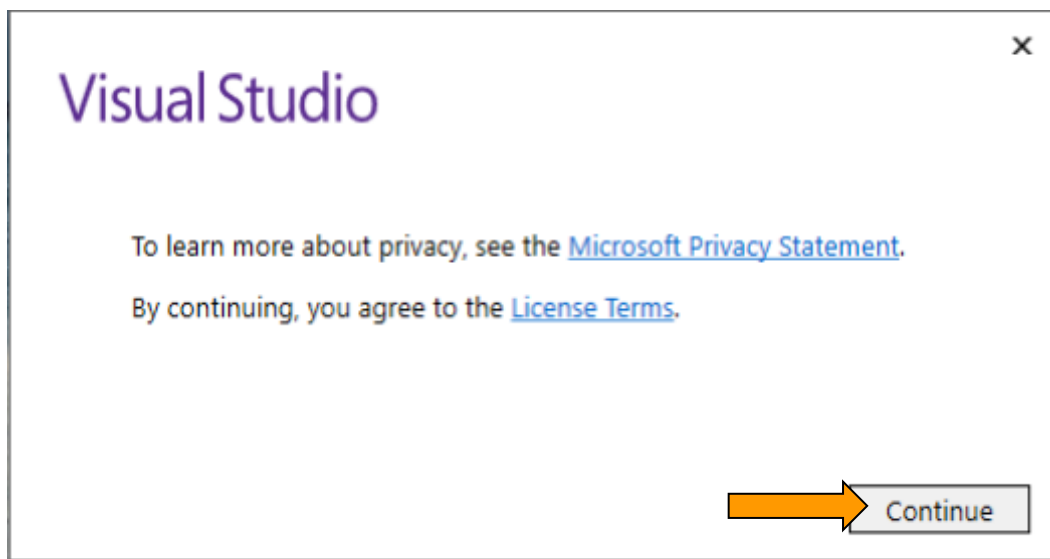
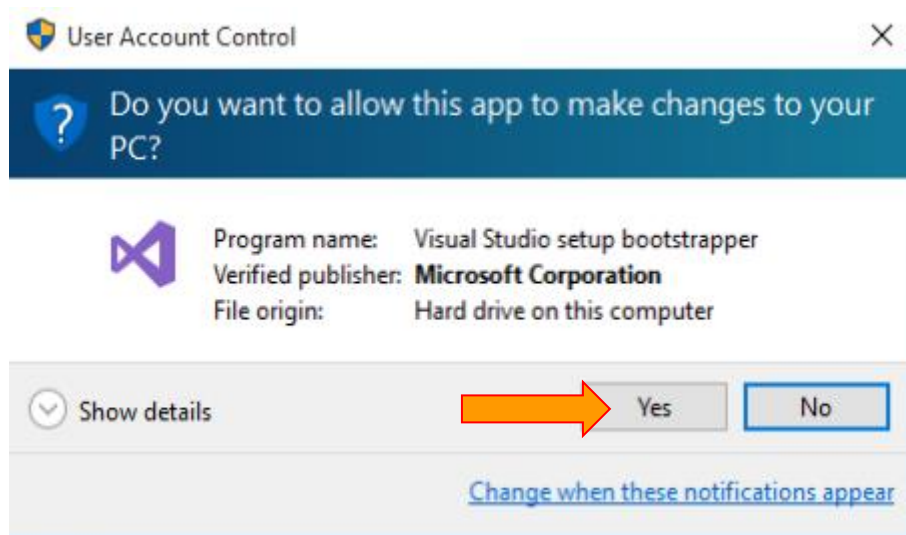
ขั้นตอนการติดตั้ง IDE (Microsoft Visual Studio 2017 Community Edition)

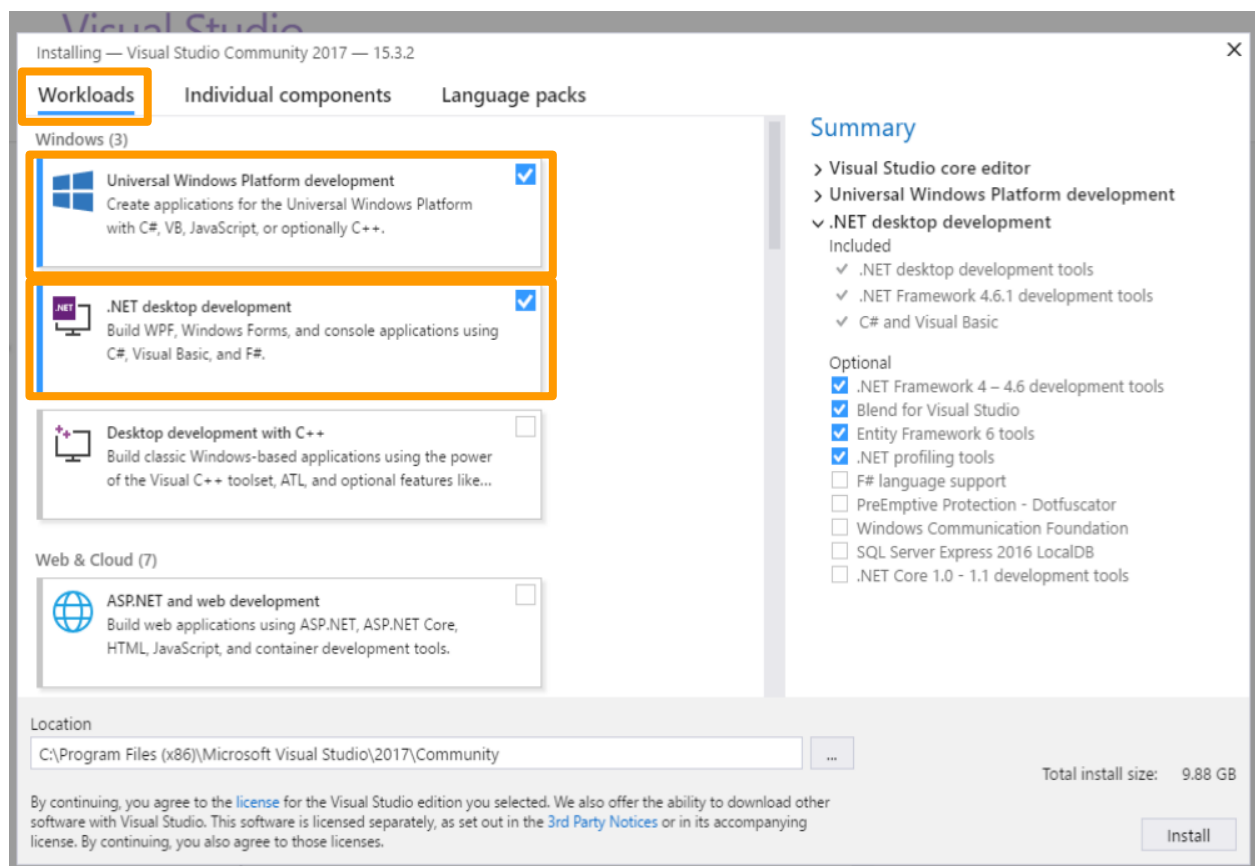
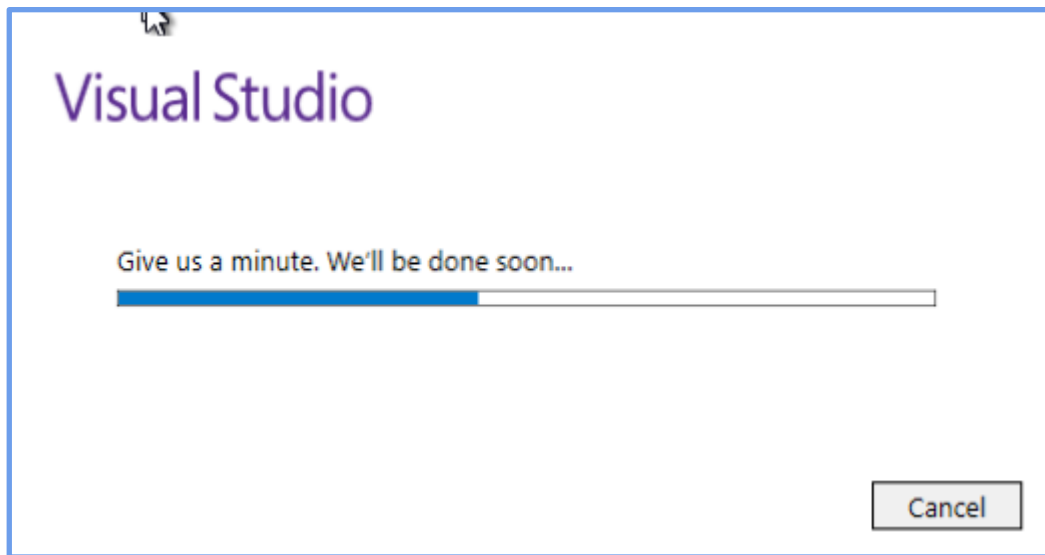
1. ดาวน์โหลดโปรแกรมสำหรับติดตั้ง ได้จาก

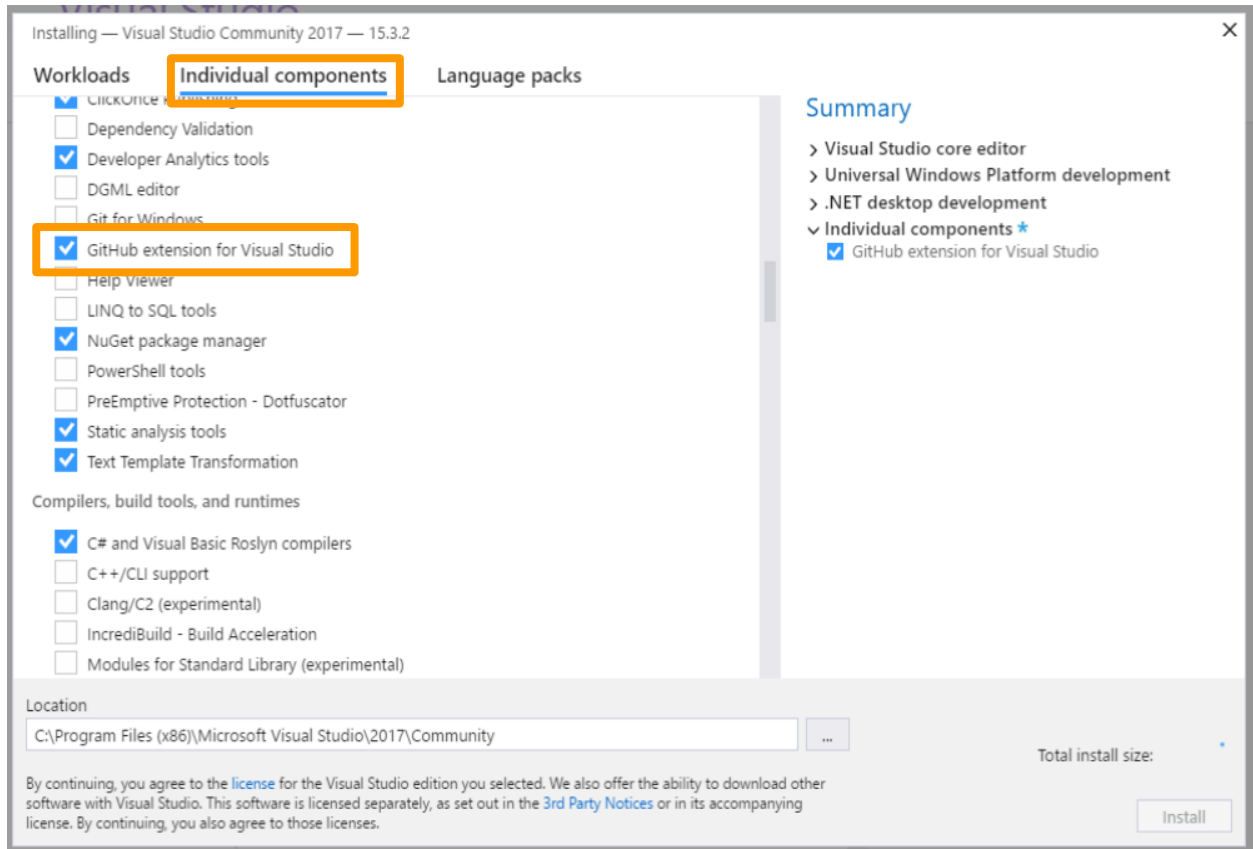
<https://www.visualstudio.com/>

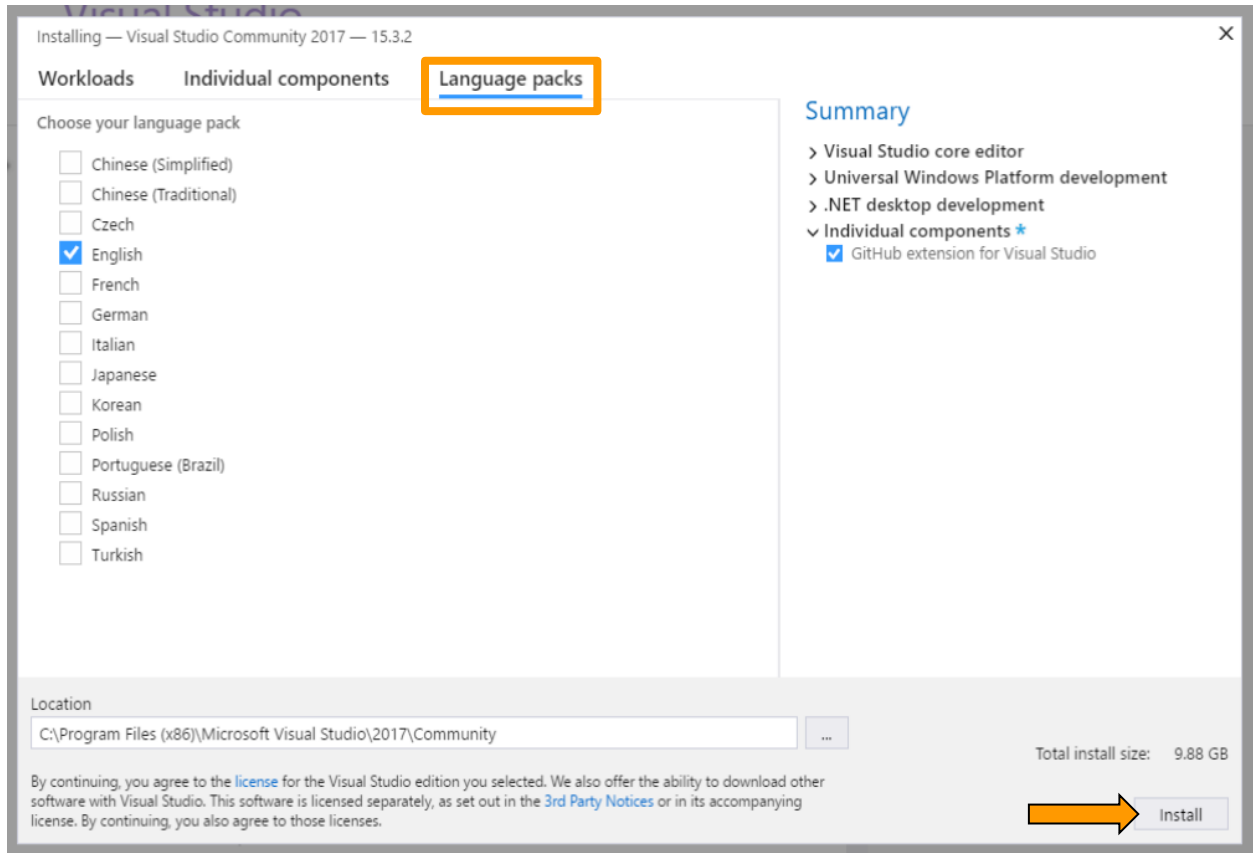


2. การติดตั้ง









Visual Studio



Products

Installed



Visual Studio Community 2017

Starting operation

0%

Starting operation

0%

Cancel

Available



Visual Studio Enterprise 2017

15.3.2

Microsoft DevOps solution for productivity and coordination across teams of any size

[License terms](#) | [Release notes](#)

Install



Visual Studio Professional 2017

15.3.2

Professional developer tools and services for small teams

Welcome!

We invite you to go online to hone your skills and find additional tools to support your development workflow.



Learn

Whether you're new to development or an experienced developer, we have you covered with our tutorials, videos, and sample code.



Marketplace

Use Visual Studio extensions to add support for new technologies, integrate with other products and services, and fine-tune your experience.

Need some help?

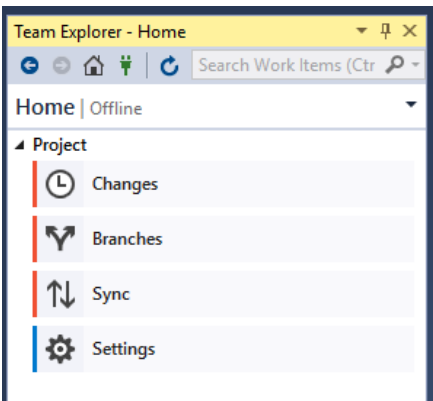
Check out the [Microsoft Developer Community](#) where developers provide feedback and answers to many common problems.

Get help from Microsoft at [Visual Studio Support](#).


1.11.33287.817

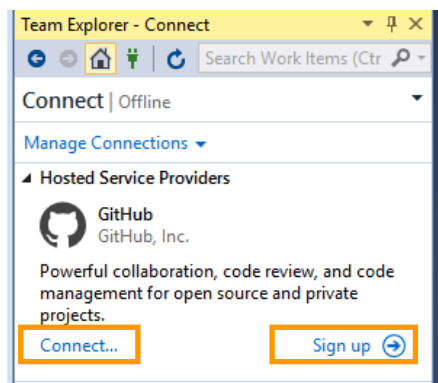
3. การทดสอบใช้งาน

การใช้งาน IDE ร่วมกับ repository (Github)

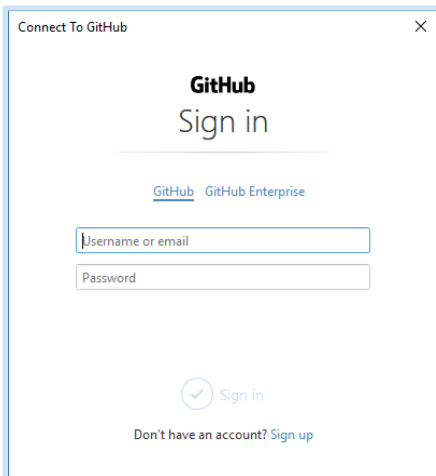


หลังจากติดตั้ง IDE และ Github extension เรียบร้อยแล้ว ให้คลิกที่เมนู View -> Team Explorer เพื่อเรียกหน้าต่าง Team Explorer ขึ้นมา จะได้หน้าต่างดังรูปขวามือ

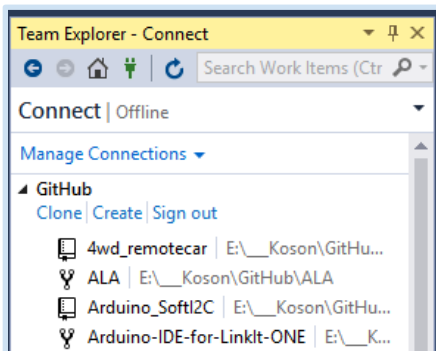
ปุ่มกดสำหรับหน้าต่าง Team Explorer มีหน้าต่างดังรูปต่อไปนี้  จากซ้ายไปขวา คือปุ่ม Backward, Forward, Home, Manage connection และ Refresh ตามลำดับ ให้คลิกที่ปุ่ม manage connection ปุ่มที่ 4



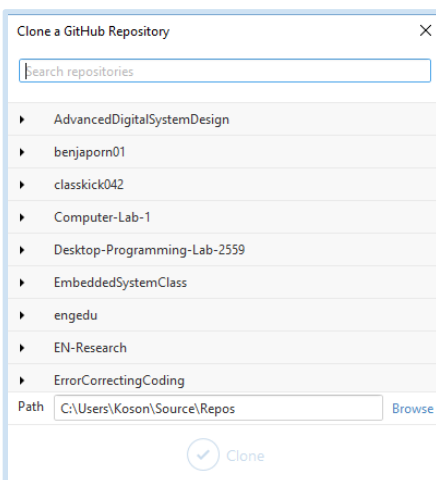
เมื่อกดปุ่ม manage connection จะปรากฏ Host Service Providers พร้อมทั้งชื่อ Host คือ GitHub ให้คลิกที่ Connect... เพื่อเชื่อมต่อไปยัง Github
(ในกรณีที่ยังไม่มีบัญชีผู้ใช้ ให้กด Sign up เพื่อลงทะเบียนกับ Github ซึ่งมีทั้งชนิดฟรีและเสียค่าใช้จ่าย)



เมื่อคลิก Connect จะปรากฏหน้าต่าง Sign in ไปยัง Github.com ให้ใส่ User name หรือ email ที่ลงทะเบียนไว้กับ Github.com พร้อมทั้ง password แล้วกดปุ่ม



หลังจาก Sign in สำเร็จ หน้าต่าง Team Explorer จะแสดง repository ทั้งหมดที่เรามีบน Localhost (อยู่บน harddisk) นอกจากนี้จะมีตัวเลือก Clone | Create | Sign out ถ้าเราคลิกที่ Clone หรือ Create ก็จะมีการแสดงหน้าจอ popup ขึ้นมา เพื่อให้ clone หรือ สร้าง repo ใหม่ตามต้องการ

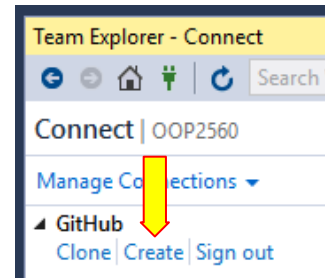


ในกรณีที่เลือก clone จะปรากฏหน้าต่าง Clone a GitHub Repository และแสดง repository ทั้งหมดที่มีอยู่บน Github

- ★ หากมี repo จำนวนมากก็สามารถใช้วิธีการ search ได้
- ★ เมื่อเลือก repo ที่ต้องการได้แล้ว ก็ต้องเลือก local path ที่จะ clone มาเก็บไว้
- ★ จากนั้นกดปุ่ม Clone

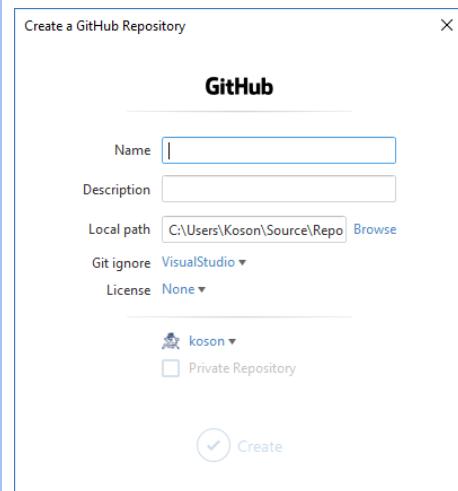
การทดลอง 3.1 การสร้าง repository บน GitHub ด้วย Visual Studio

1. สร้าง repository โดยการคลิกปุ่ม Create



2. ระบุรายละเอียดของ repo ใหม่ที่จะสร้าง

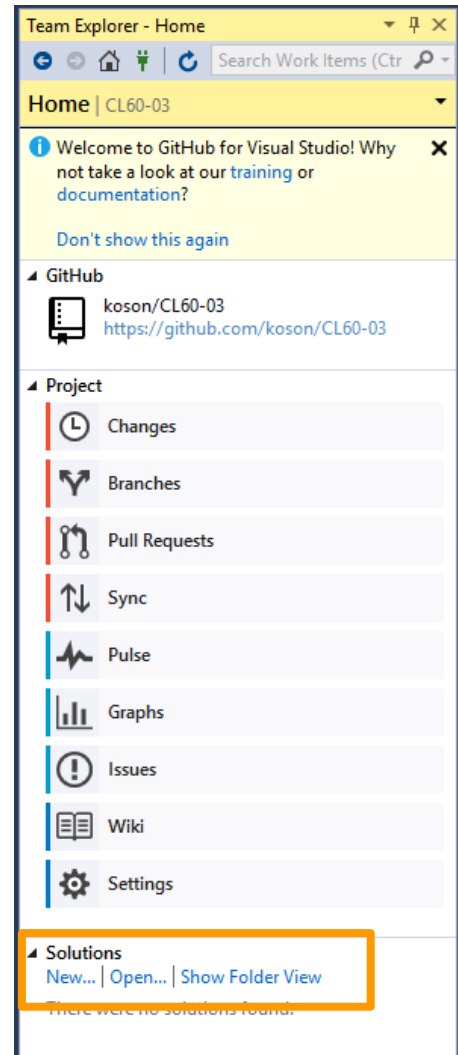
- ในช่อง Name ให้เพิ่มชื่อ repository
CL60-03
- ในช่อง Description ให้ใส่รายละเอียดสั้นๆ
Computer Laboratory 1 : Lab 03
- ในช่อง Local path ให้ใส่ path ไปยังงานที่เคยทำไว้แต่อยู่ใน folder เดียวกัน เช่น ถ้างานเดิม c:\Code\CL60-02 ให้ใส่เป็น
c:\Code\CL60-03
- ในช่อง Git ignore และ License คงไว้อย่างที่ปรากฏ
- กดปุ่ม **Create**



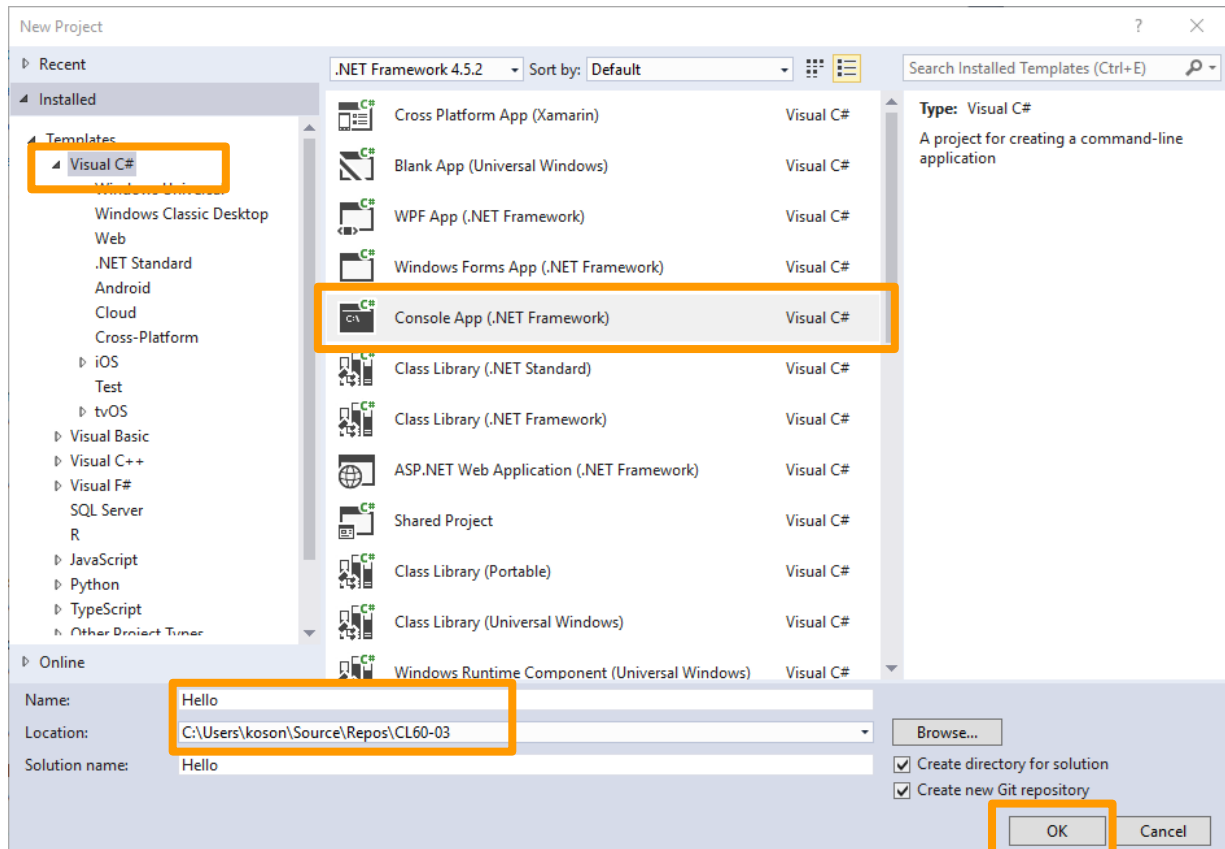
เมื่อสร้างเสร็จ ให้ตรวจสอบรายการ repository ที่เพิ่มขึ้นบน GitHub.com

3. เพิ่ม solution ใหม่ไปยัง repository

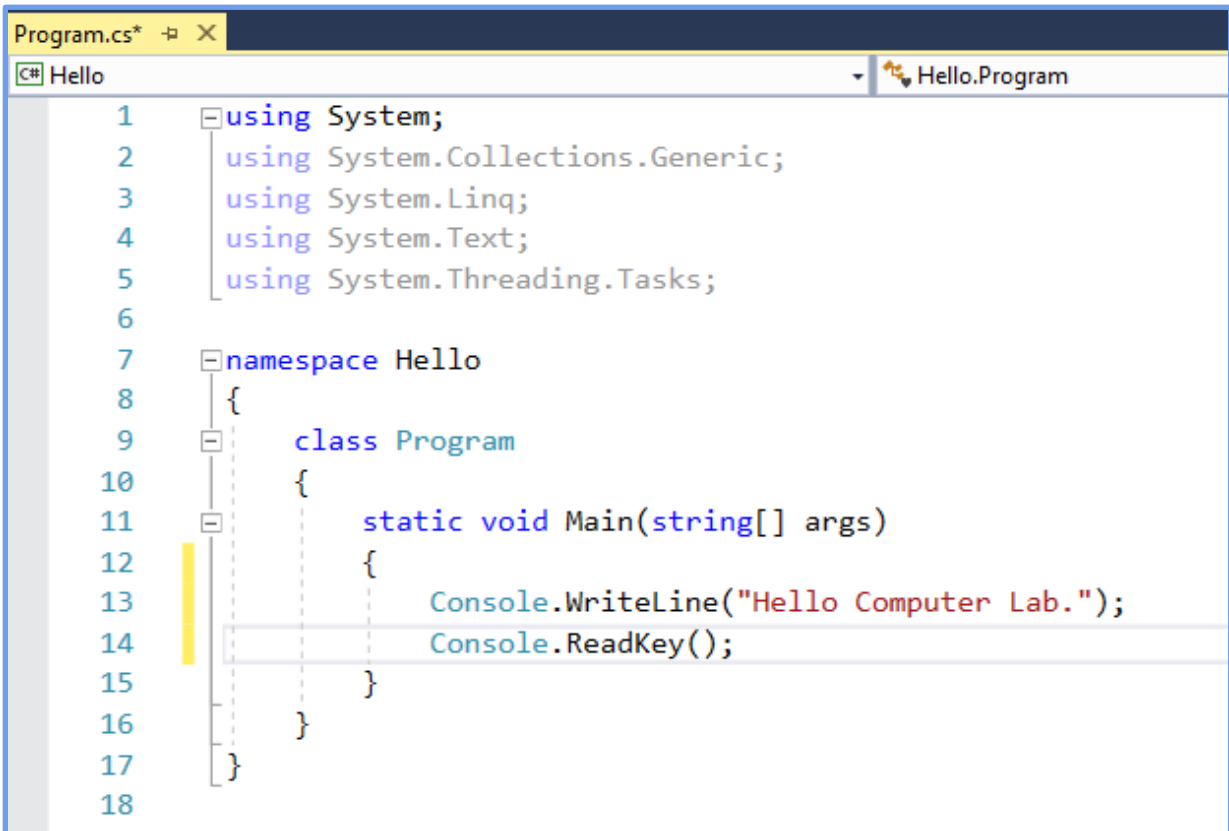
- คลิกที่ New... จะปรากฏหน้าต่าง ให้สร้าง Solution ใหม่ให้ทำตามข้อ 4



4. ใส่รายละเอียดของ Solution ใหม่ที่ต้องการสร้าง

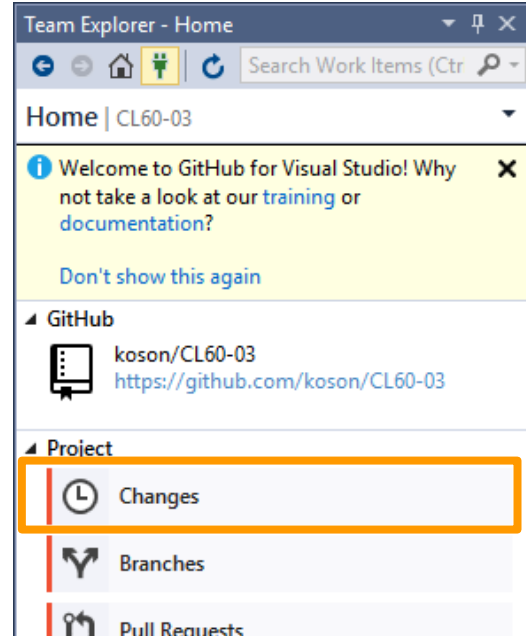


5. เมื่อ Solution ใหม่ถูกสร้างขึ้นมา ให้แก้ไข code ตามรูปต่อไปนี้

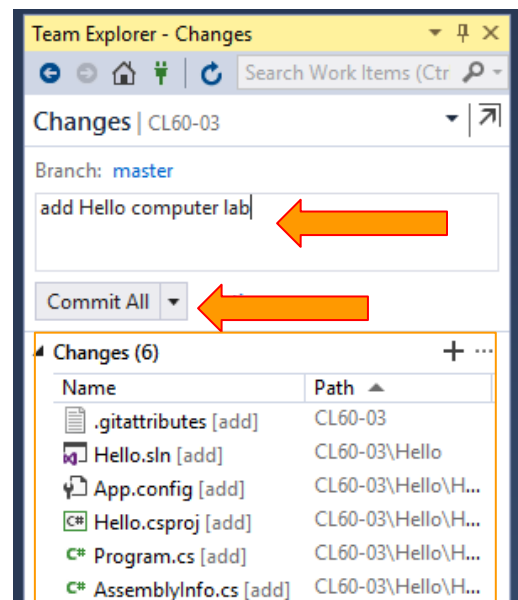


```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Hello
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Hello Computer Lab.");
14             Console.ReadKey();
15         }
16     }
17 }
18
```

6. เมื่อ แก้ไขเสร็จ ให้คลิกที่ Changes
เพื่อดูการเปลี่ยนแปลง เทียบได้กับการสั่ง git status

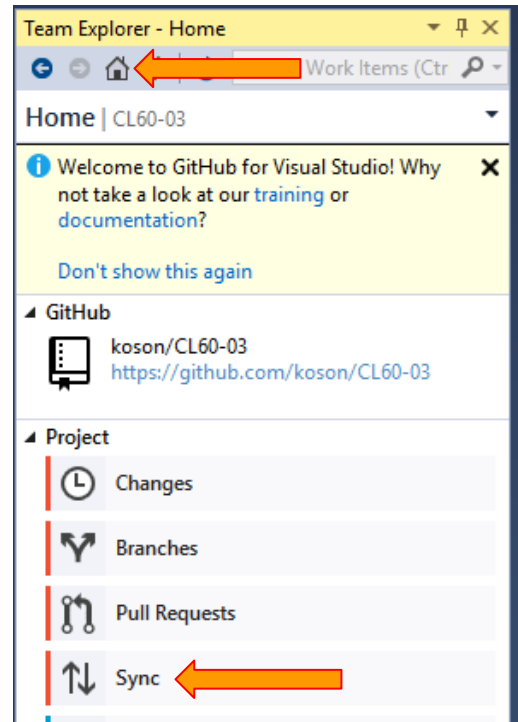


7. Team Explorer จะรายงานไฟล์ที่มีการเปลี่ยนแปลง
พร้อมทั้งมีช่องให้ใส่ข้อความสำหรับการ commit
เมื่อกรอกข้อความเสร็จให้คลิกปุ่ม Commit All



การสั่ง Commit All จะเป็นการเก็บบันทึกการเปลี่ยนแปลงที่แสดงในรูปไว้ใน Local repository
ถ้าหากเราต้องการบันทึกการเปลี่ยนแปลงไว้บนเว็บ GitHub จะต้องทำการ Sync ซึ่งคำสั่ง Sync นี้จะทำการ
pull และ push ให้เราโดยอัตโนมัติทั้ง 2 คำสั่ง

8. ให้ทำการ sync repo ที่เราสร้างขึ้นไปบน Github



เมื่อ sync เสร็จ ให้ตรวจสอบรายการไฟล์ใน repository บน Github.com

การทดลอง 3.2 การ clone จาก repository ด้วย Visual Studio

9. สร้าง repository บน GitHub.com โดยมีรายละเอียดดังนี้

- Repository name : HelloWorld
- [x] Initialize this repository with a README
- [Add .gitignore : VisualStudio]

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



koston

Repository name

HelloWorld



Great repository names are short and memorable. Need inspiration? How about [redesigned-parakeet](#).

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: VisualStudio

Add a license: None

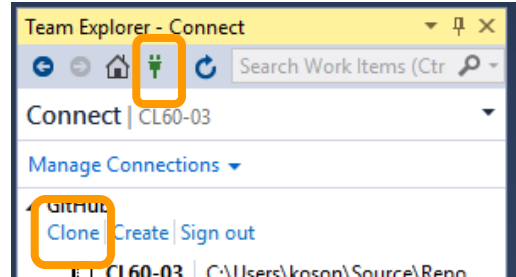


Create repository

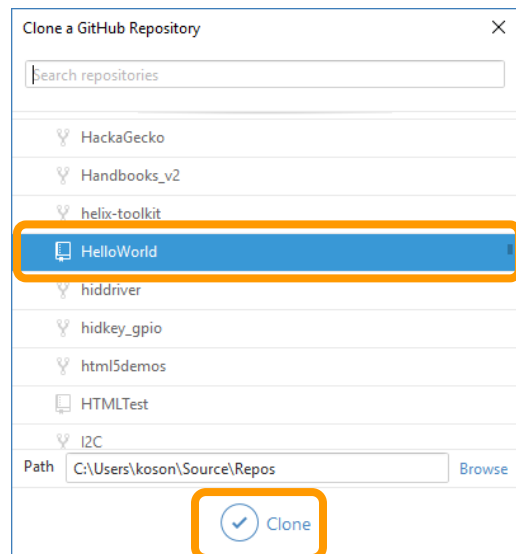


10. ใน Visual Studio ให้ไปที่หน้าต่าง Team Explorer แล้วคลิกที่ Manage Connections

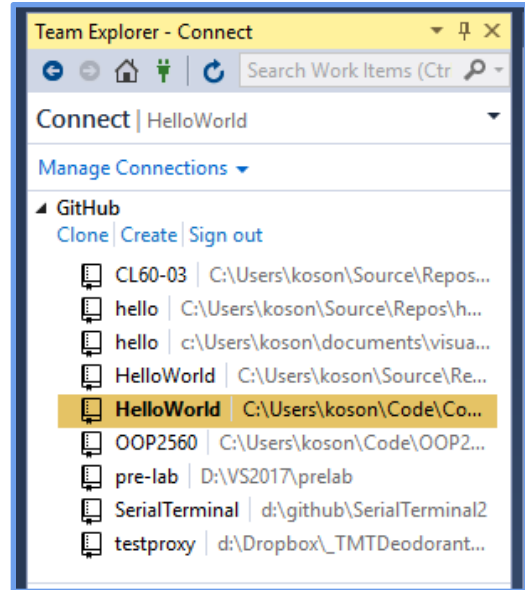
11. คลิกที่ Clone



12. ในหน้าต่าง Clone a GitHub Repository ให้เลือก repository ชื่อ HelloWorld ที่ได้สร้างไว้ในขั้นตอนที่ 8 แล้วกดปุ่ม Clone

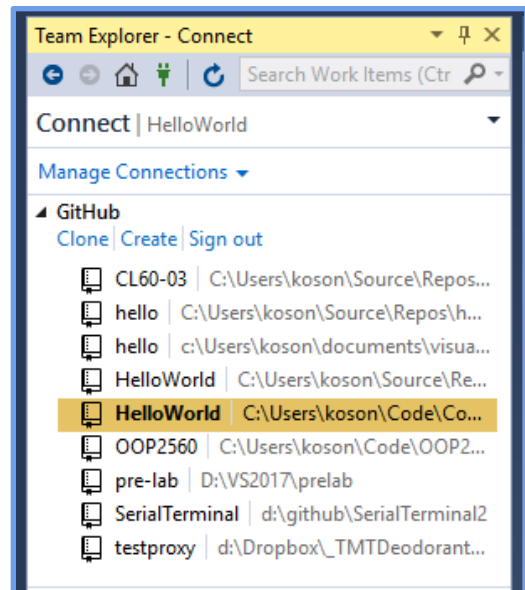


13. ให้ Double click ที่ชื่อ repository ที่ clone มา



14. หน้าต่าง Team Explorer จะแจ้งว่ายังไม่มี Solution ใดๆ ใน repository ให้เราคลิกที่ New... เพื่อสร้าง Solution ใหม่

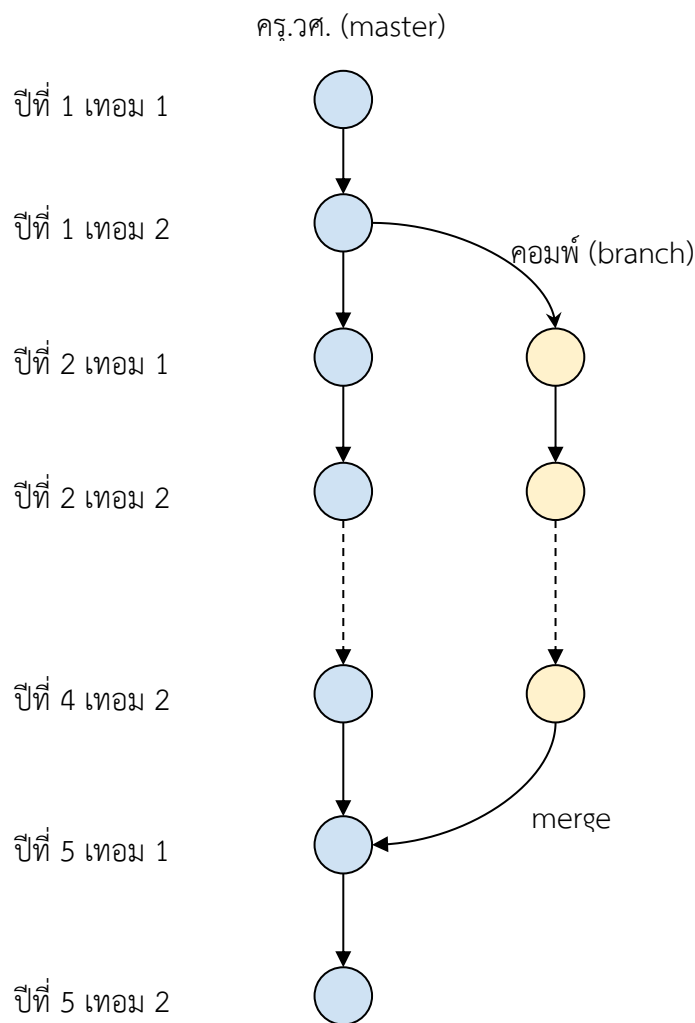
15. ให้ทำการสร้าง Solution พร้อมทั้ง Sync กับ Github ตามขั้นตอนที่ 4 - 8



เมื่อ sync เสร็จ ให้ตรวจสอบรายการไฟล์ใน repository บน GitHub.com

การ branch

แนวคิดในการทำ Branching เทียบเคียงกับในชีวิตประจำวัน



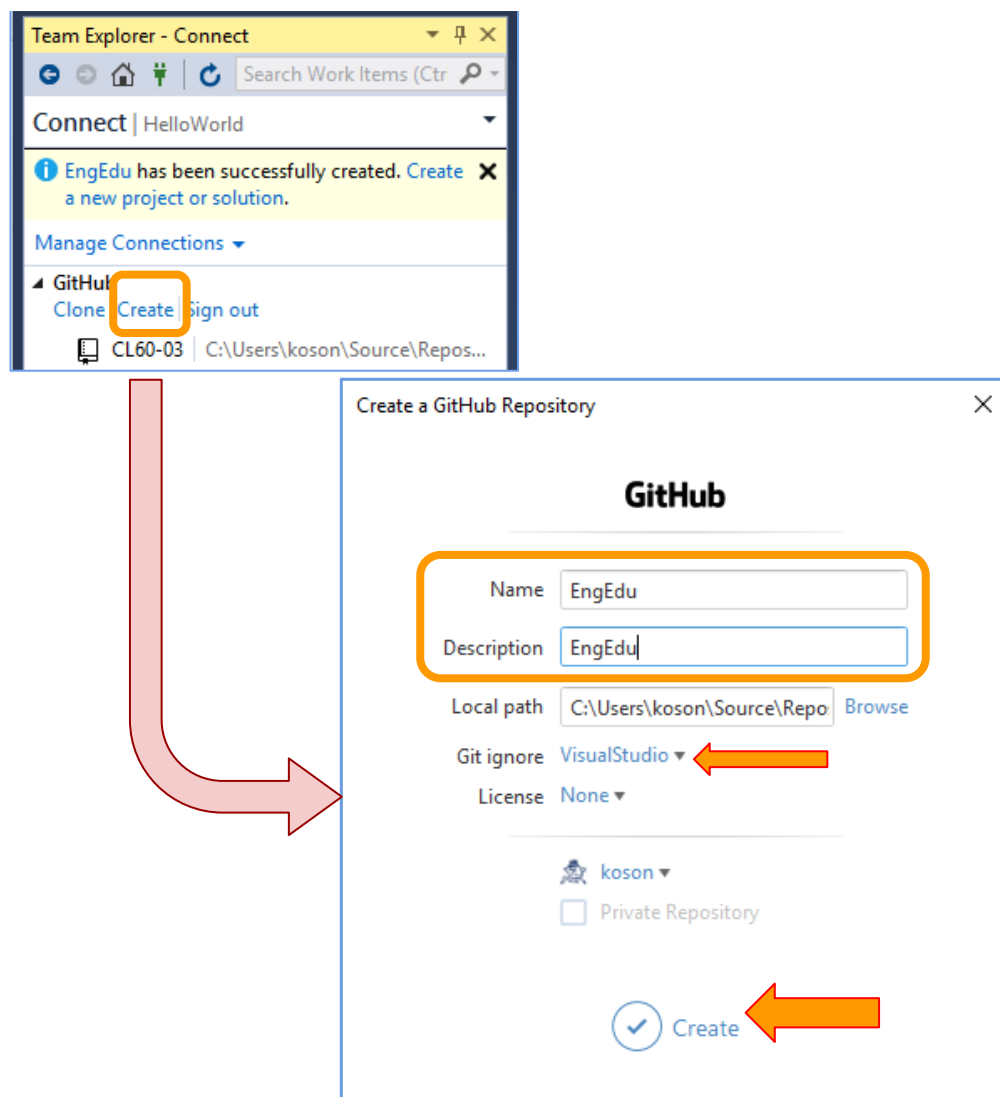
แนวคิดในการทำ branching บน git อาจเทียบได้กับรูปทางด้านบน นั่นคือ การเข้าเรียนในภาควิชา ครุ.วศ. เป็นจุดเริ่มต้นของการศึกษา (หรือเทียบได้กับพัฒนา Project ของเรา) โดยแต่ละ commit (แทนด้วยสัญลักษณ์วงกลม) อาจจะหมายถึงการเรียนในแต่ละภาคการศึกษา ในแต่ละ commit จึงเป็นการเพิ่มรายวิชาลงในทรานสคริปต์ (ตามเทอมที่ลงทะเบียน) เมื่อนักศึกษาขึ้นชั้นปีที่ 2 จะต้องแยกกันไปเรียนตามความถนัดหรือตามความสามารถที่ต้องการพัฒนา จนกระทั่งจบชั้นปีที่ 4 จะต้องกลับมารวมกันอีกครั้ง เพื่อออกไปทำการฝึกสอนในฐานะนักศึกษา ภาควิชา ครุ.วศ.

ในการพัฒนา software มีบ่อยครั้งที่เราต้องทำการเพิ่มเติมความสามารถพิเศษให้กับซอฟต์แวร์

หรือทำการแก้ bug ที่พบ โดยไม่ต้องการให้เกิดผลกระทบต่อ source code ที่อยู่ในสาขาหลัก (master) เราจึงต้องทำการแยกแขนง (branch) ออกมาเพื่อพัฒนา ซึ่งในการแยก branch ออกมาพัฒนานี้ ก็ยังสามารถทำงานเป็นทีมและยังคงต้อง clone มาทำงานกับ local repository เช่นเดียวกัน

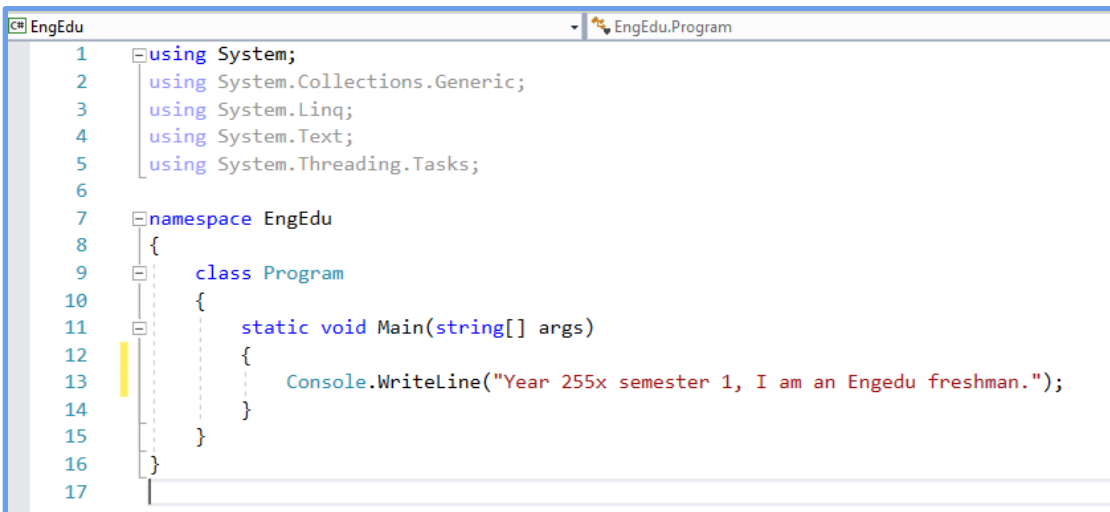
การทดลอง 3.3 การทำ Branching

16. ในหน้าต่าง Team Explorer ให้ทำการสร้าง repository ใหม่ โดยใช้ชื่อว่า EngEdu



17. ทำการเพิ่ม Solution ให้กับ repository ใช้ชื่อว่า EngEdu

- ชนิดเป็น Console App (.NET Framework)
- Location คงไว้อย่างที่ปรากฏ
- แก้ไขโปรแกรมให้เป็นอย่างรูปด้านล่างนี้

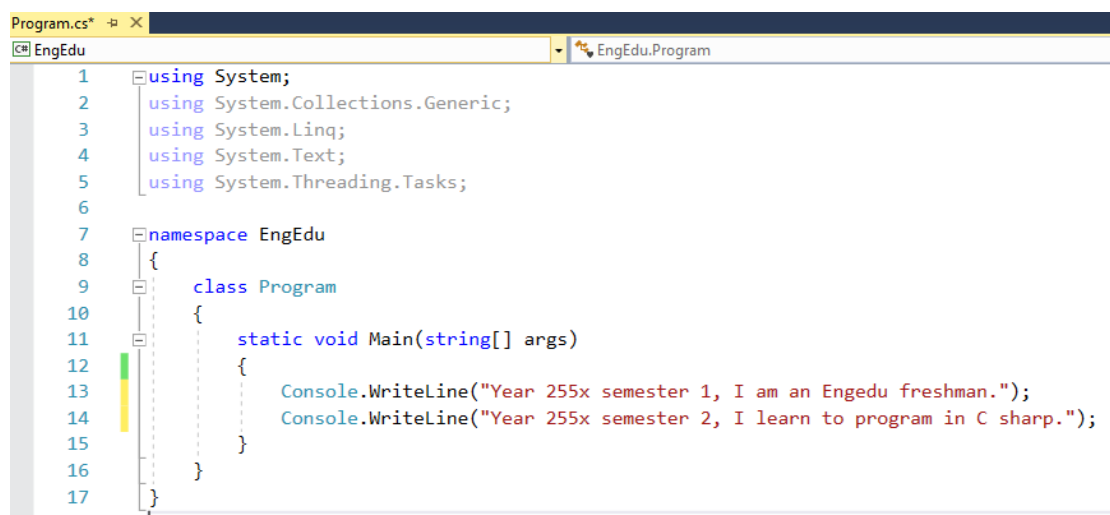


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace EngEdu
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            Console.WriteLine("Year 255x semester 1, I am an Engedu freshman.");
14        }
15    }
16 }
17
```

18. ทำการ sync กับ GitHub.com

- ใช้ commit message “first year semester 1”
- ดูการเปลี่ยนแปลงของ repository EngEdu บน GitHub.com

19. ทำการเพิ่มเติมไฟล์ Program.cs โดยเพิ่มข้อความดังต่อไปนี้



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace EngEdu
8 {
9     class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Year 255x semester 1, I am an Engedu freshman.");
14             Console.WriteLine("Year 255x semester 2, I learn to program in C sharp.");
15         }
16     }
17 }
```

20. ทำการ sync กับ GitHub.com

- ใช้ commit message “first year semester 2”
- ดูการเปลี่ยนแปลงของ repository EngEdu บน GitHub.com

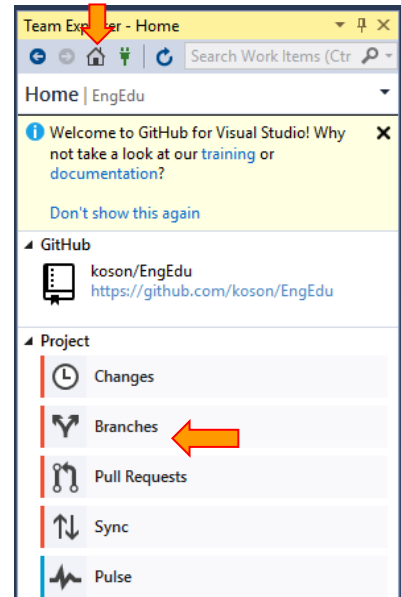
เมื่อนักศึกษาเรียนถึงชั้นปีที่ 2 จะต้องมีการแยกแขนงวิชา เพื่อศึกษาในแขนงวิชาที่ตนถนัด หรือต้องการพัฒนาศักยภาพเป็นพิเศษ เมื่อนักศึกษาแยกแขนงออกมาแล้ว ก็จะต้องเรียนในวิชาของแขนงวิชาเป็นหลัก แต่จะยังคงมีนักศึกษาบางส่วน ที่พักการเรียนในบางวิชาของปี 1 เทอม 1 จึงต้องลงทะเบียนใน ปี 2 เทอม 1 ซึ่งจะยังคงเดินตามแผนการศึกษาในรายวิชาของภาควิชาต่อไป (เรียกชื่อ branch นี้ว่า master)

ทำนองเดียวกัน ในการพัฒนาซอฟต์แวร์ เมื่อได้วางตลาดรุ่นที่ 1.0 แล้ว ก็อาจจะต้องมีการพัฒนา features เพิ่มเติม เป็นรุ่น 1.1, 1.2, ... หรือรุ่น 2.0 สิ่งหนึ่งที่นักพัฒนาต้องพบกับความยุ่งยากคือ การพัฒนารุ่นใหม่ ไปพร้อมๆ กับการแก้บั๊กของรุ่นเก่า จะเกิดความยุ่งยากในการรักษา code เพื่อไม่ให้ code

ของแต่ละรุ่นหรือแต่ละความมุ่งหมายในการพัฒนารบวงกัน ความสามารถด้าน branching ของ Git จึงเข้ามามีบทบาทสำคัญ

21. การแยก branch ใน Visual Studio

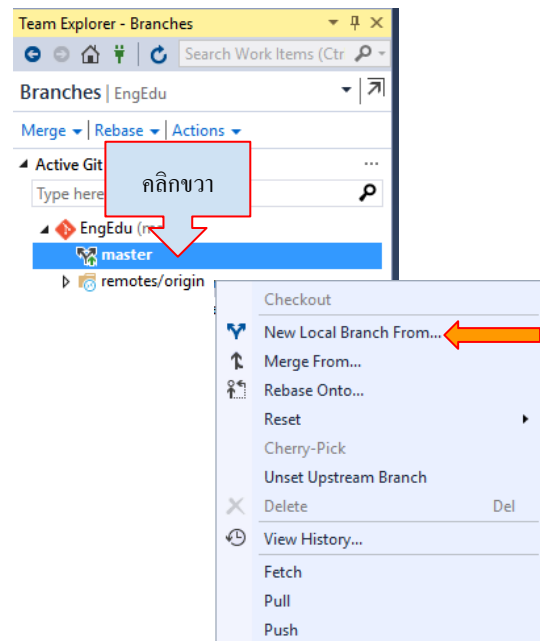
- ในหน้าต่าง Team Explorer ให้คลิกที่ Home และ Branches (อยู่ภายใต้หัวข้อ Project)



22. ภายใต้อัฒวข้อ Active Git Repositories

จะพบว่ามี Repo ของเรตามด้วย (master) และมี branch ที่ชื่อว่า master เป็นโนนดลูก แสดงเป็นตัวหนา

- คลิกขวาที่ branch master
- เลือก New Local Branch From...



23. ตั้งชื่อ Branch

- ใส่ชื่อ Branch เป็น Computer
- คลิก Create Branch

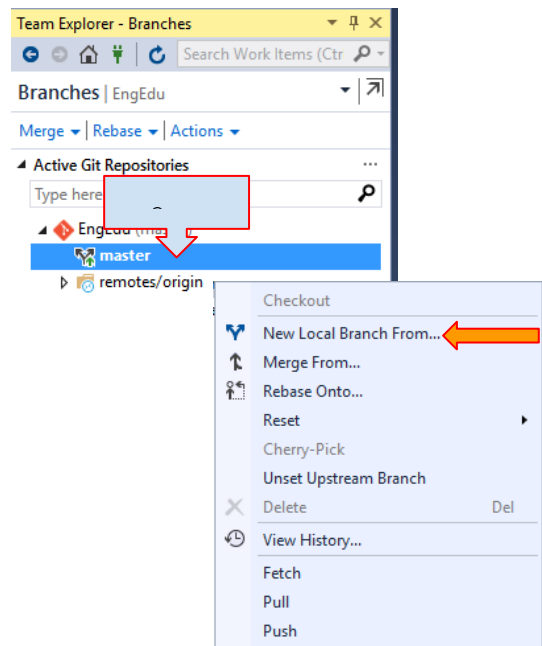
สังเกตว่า ในโหมด Engedu จะมีการแสดงเป็น EngEdu (Computer) และโหมดลูกที่ชื่อ Computer จะเปลี่ยนเป็นตัวหนาขณะที่ master กลายเป็นตัวปกติ นั่นคือ ขณะนี้ เราได้ย้ายไปทำงานใน branch ที่ชื่อ Computer

***** การเปลี่ยนแปลงใด ๆ จะถูกกระทำภายใต้ branch ปัจจุบัน (ที่แสดงด้วยตัวหนา) เท่านั้น *****

การเปลี่ยน branch ทำได้ง่ายๆ โดยการ double click ที่ชื่อ branch

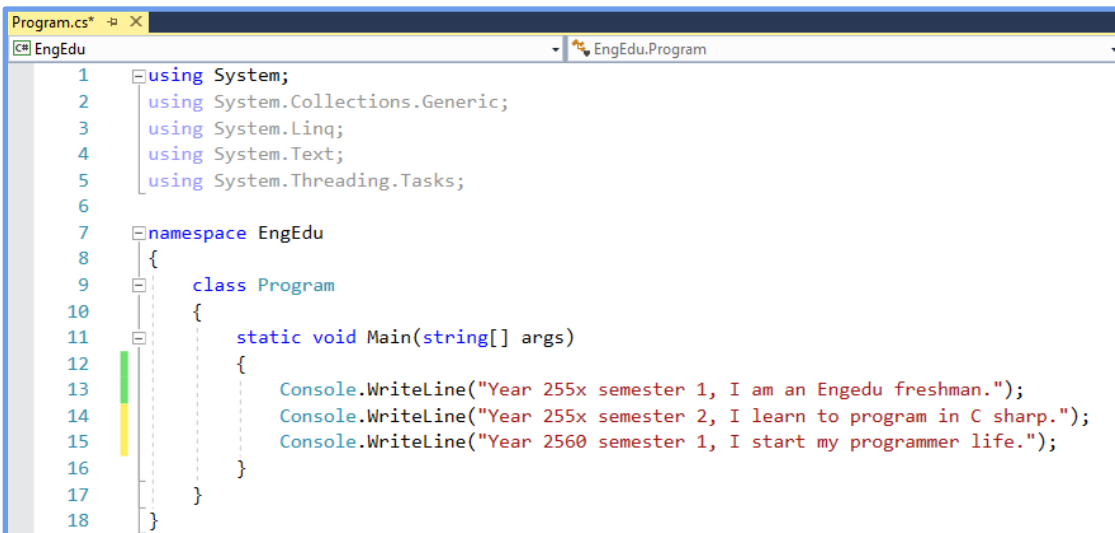
***** ทุกครั้งที่มีการเปลี่ยน branch

1. git จะลบไฟล์ทั้งหมดใน working directory
2. git จะทำการ checkout ไฟล์ใน branch ปัจจุบันออกมาทำงาน แทนไฟล์ทั้งหมดใน working directory



24. ทำให้แน่ใจว่า เรากำลังทำงานกับ Branch ที่ชื่อ Computer

- ในโหมด EngEdu จะต้องแสดงเป็น EngEdu (Computer) และโหมดลูกที่ชื่อ Computer จะต้องเป็นตัวหนา
- แก้โปรแกรมเป็นดังต่อไปนี้



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace EngEdu
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            Console.WriteLine("Year 255x semester 1, I am an Engedu freshman.");
14            Console.WriteLine("Year 255x semester 2, I learn to program in C sharp.");
15            Console.WriteLine("Year 2560 semester 1, I start my programmer life.");
16        }
17    }
18 }
```

25. ทำการ sync กับ GitHub.com

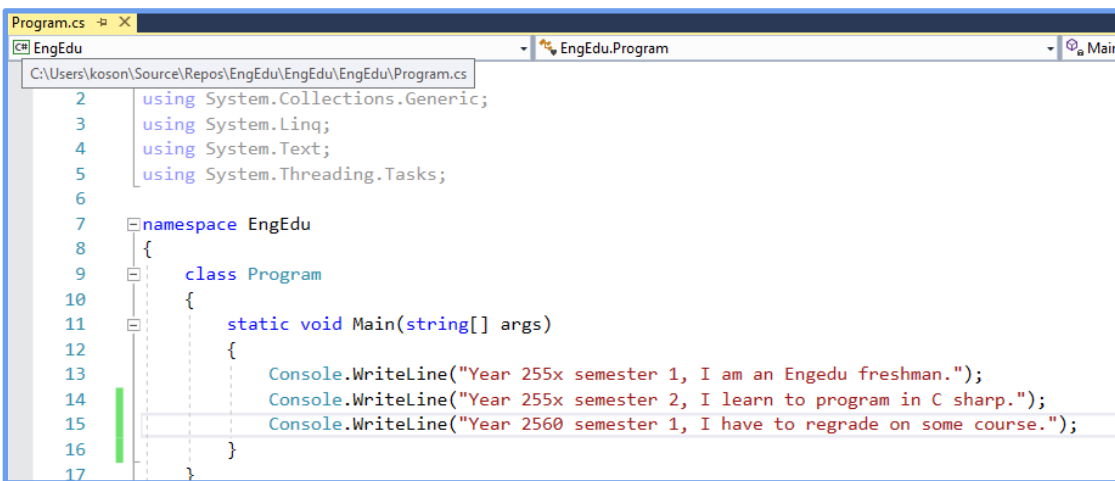
- ใช้ commit message “2nd year semester 1”
- ดูการเปลี่ยนแปลงของ repository EngEdu บน GitHub.com
- กรณีที่ Sync ไม่สำเร็จ ให้ลองใช้การ push เนื่องจากในขณะนี้ ยังไม่มี branch ที่ชื่อ Computer บน GitHub.com

26. เปลี่ยน Branch กลับไปยัง master โดยการ double click ที่ master

- สังเกตการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์ Program.cs
- ทดลองสลับไปมาระหว่าง branch ที่ชื่อ master และ Computer (โดยการ double click)

27. ทำให้แน่ใจว่า เรากำลังทำงานกับ Branch ที่ชื่อ master

- ในโหมด Engedu จะต้องแสดงเป็น EngEdu (master) และโหมดลูกที่ชื่อ master จะต้องเป็นตัวหนา
- แก๊โปรแกรมเป็นดังต่อไปนี้



```
Program.cs
EngEdu
C:\Users\koston\Source\Repos\EngEdu\EngEdu\EngEdu\Program.cs
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace EngEdu
8 {
9     class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Year 255x semester 1, I am an Engedu freshman.");
14             Console.WriteLine("Year 255x semester 2, I learn to program in C sharp.");
15             Console.WriteLine("Year 2560 semester 1, I have to regrade on some course.");
16         }
17     }
18 }
```

28. ทำการ sync กับ GitHub.com

- ใช้ commit message “2nd year semester 1”
- ดูการเปลี่ยนแปลงของ repository EngEdu บน GitHub.com
- กรณีที่ Sync ไม่สำเร็จ ให้ลองใช้การ push เนื่องจากในขณะนี้ ยังไม่มี branch ที่ชื่อ Computer บน GitHub.com

29. เปลี่ยน Branch กลับไปยัง Computer โดยการ double click ที่ Computer

- สังเกตการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์ Program.cs
- ทดลองสลับไปมาระหว่าง branch ที่ชื่อ master และ Computer (โดยการ double click)

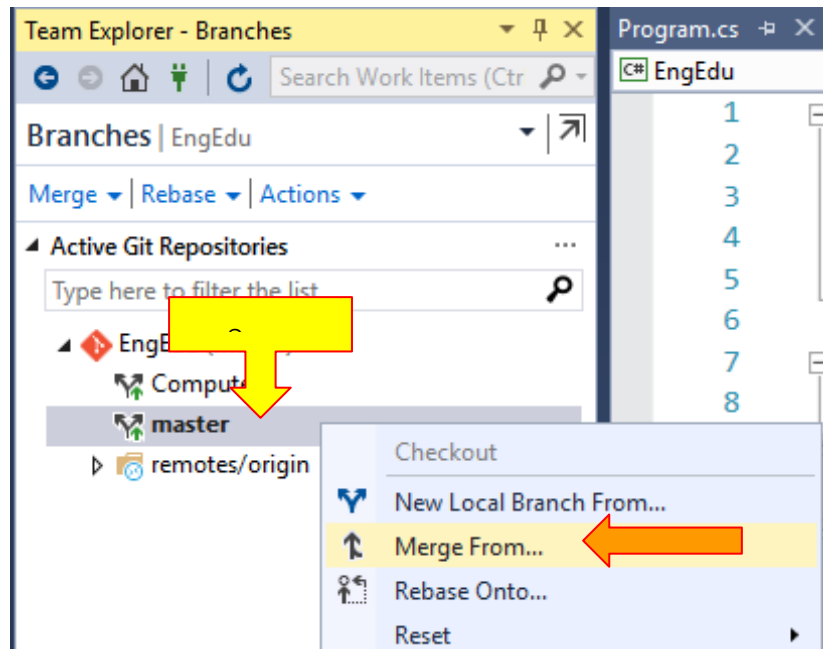
แบบฝึกหัด

ให้ทำการเพิ่ม commit ตามภาคการศึกษาต่างๆ โดยเขียน Hilight
ที่นักศึกษาคิดว่าที่จะพบในแต่ละภาคการศึกษา จนถึงชั้นปีที่ 4 ภาคเรียนที่ 2

เมื่อเราพัฒนาซอฟต์แวร์ โดยการแยก branch และคิดว่าถึงจุดหนึ่งที่สามารถนำไปรวมกับ master ได้ เราจะต้องทำการ merge เพื่อรวม branch เข้าด้วยกัน ซึ่งการ merge นี้ระบบของ git จะทำการตรวจเช็คให้ด้วยว่าสามารถ merge ได้หรือไม่

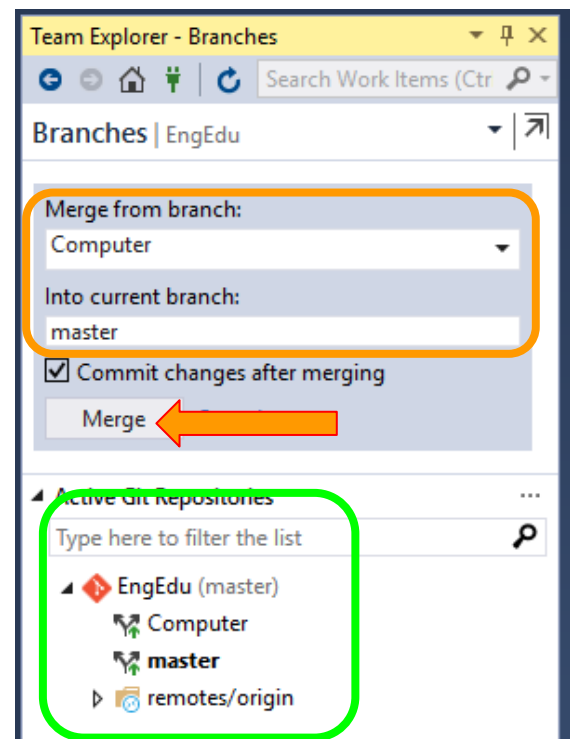
30. การรวม Branch (merge)

- ทำให้แน่ใจว่าอยู่ที่ Branch ปลายทางที่จะทำการ merge ในที่นี้คือ master ดังนั้น ที่ไหน EngEdu ต้องแสดงเป็น EngEdu (master) และไหนดลูก master ต้องเป็นตัวหนา
- คลิกขวาที่ไหนดลูก เลือก Merge From... จากเมนูป๊อปอัพ



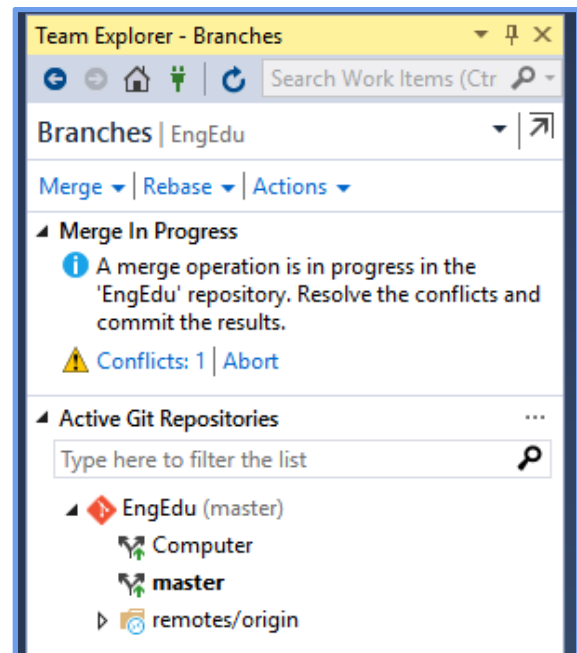
31. ทำการ merge

- Merge from branch : **Computer**
- Into current branch: **master**
- **คลิก Merge**



32. Merge conflict

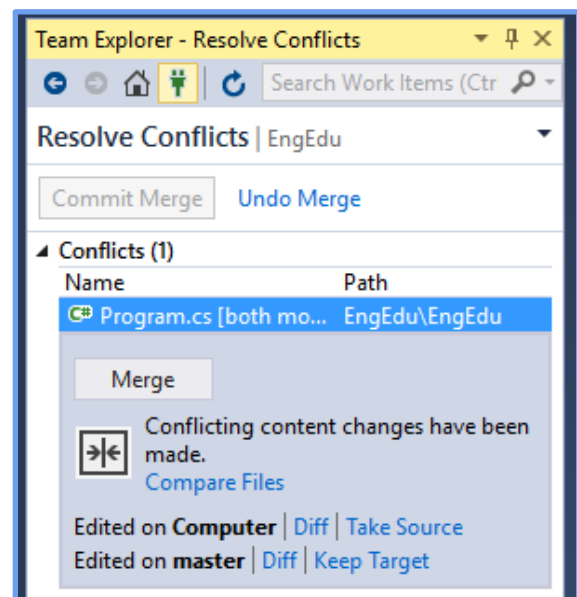
ในการ merge เราจะพบว่า บางครั้งมีการแก้ไข source code ที่หมายเลขบรรทัดเดียวกัน บนไฟล์เดียวกัน กรณีนี้เรียกว่าเกิดการขัดแย้ง เราต้องดำเนินการอย่างใดอย่างหนึ่ง เพื่อจัดการกับความขัดแย้งนั้น เพื่อให้สามารถดำเนินการพัฒนาซอฟต์แวร์ต่อไปได้ ถ้าเจอสถานการณ์นี้ ให้คลิก Conflict: 1



33. การแก้ไข Merge conflict

ในการ merge เราจะพบว่า บางครั้งมีการแก้ไข source code ที่หมายเลขบรรทัดเดียวกัน บนไฟล์เดียวกัน กรณีนี้เรียกว่าเกิดการขัดแย้ง ให้คลิกที่ชื่อไฟล์ที่ขัดแย้งนั้น จะพบว่า เราสามารถแก้ไขได้ในหลายรูปแบบเช่น

1. ยกเลิกการ merge โดยการคลิกที่ Undo Merge
2. เปรียบเทียบความแตกต่างระหว่างไฟล์ทั้งสอง โดยการคลิกที่ Compare file



3. เข้าไปแก้ไขไฟล์ใน branch ปลายทาง

4. เข้าไปแก้ไขไฟล์ที่นำมา merge

ให้นักศึกษาทดลองคลิกที่ตัวเลือกต่างๆ

ถ้าต้องการกลับไปก่อนการ merge ให้กด Undo merge

34. ดูผลการเปลี่ยนแปลงหลังจากการ merge

➤ สังเกตการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์ Program.cs

➤ ทดลองสลับไปมาระหว่าง branch ที่ชื่อ master และ Computer (โดยการ double click)

คำถาม

1. ในกรณีที่ต้องการรักษา source code ของทั้งสองไฟล์ไว้ ควรทำอย่างไร
2. ผลของการ keep target ออกมาเป็นอย่างไร
3. ผลของการ take source ออกมาเป็นอย่างไร
4. ให้ capture หน้าจอของ Compare files, Diff ที่ source, Diff ที่ target

หมายเหตุ หากนักศึกษาทำการทดลองเพื่อที่จะตอบคำถามข้างต้น แล้วทำให้เกิดผลที่ไม่คาดคิด ให้กด Undo merge