# ใบงานการทดลองที่ 10
# เรื่อง การควบคุมเวอร์ชันการทำงานผ่านโปรแกรม Eclipse

## 1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการติดต่อกับผู้ใช้งาน และการหลายงานพร้อมกัน

1.2. รู้และเข้าใจการติดต่อระหว่างงาน

## 2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

## 3. ทฤษฎีการทดลอง

3.1. Version Control System (VCS) คืออะไร? มีประโยชน์อย่างไร?

Version Control System (VCS) เป็นระบบที่ช่วยในการบริหารจัดการเอกสารหรือโค้ดต่างๆ โดยที่มีการบันทึกประวัติการแก้ไขไฟล์ หรือการเปลี่ยนแปลงต่างๆ ที่เกิดขึ้นในโค้ด ทำให้สามารถตรวจสอบและเปรียบเทียบไฟล์หรือโค้ดเวอร์ชันต่างๆ ได้ง่ายขึ้น

3.2. Git ต่างกับ Github อย่างไร?

Git และ Github เป็นสองสิ่งที่ไม่เหมือนกัน โดย Git เป็น Version Control System (VCS) ที่ใช้สำหรับการจัดการรหัสของโปรเจ็กต์ ในขณะที่ Github เป็นเว็บไซต์ที่ให้บริการเป็น Host สำหรับ Git repository และมีฟีเจอร์ต่าง ๆ ที่ช่วยให้การทำงานร่วมกันระหว่างผู้พัฒนางานต่าง ๆ ได้สะดวกและสามารถเก็บรักษาข้อมูลโปรเจ็กต์ได้อย่างมีประสิทธิภาพมากขึ้น

3.3. Repository คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Repository เป็นที่เก็บรวบรวมไฟล์ของโค้ด และ version control history ของโปรเจ็กต์ โดยที่ repository สามารถเก็บได้ทั้งโค้ดต้นฉบับ และ version ต่าง ๆ ที่เปลี่ยนแปลงจากโค้ดต้นฉบับ เช่น การเพิ่มฟีเจอร์ การแก้ไขบั๊ก และการพัฒนาเวอร์ชันต่าง ๆ

3.4. Clone คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Clone หมายถึงการสร้างรอบสำเนา (copy) ของ repository จากที่อื่นมาไว้ที่เครื่องคอมพิวเตอร์ของเราเพื่อทำงาน โดยสามารถ clone repository จาก server หรืออื่นๆ ได้ทั้งแบบ public หรือ private

3.5. Commit คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

การ Commit ในระบบ Version Control System เป็นการบันทึก Snapshot หรือสถานะปัจจุบันของไฟล์และโค้ดทั้งหมดใน Repository โดยในแต่ละครั้งที่ทำการ Commit จะต้องระบุข้อความอธิบายการเปลี่ยนแปลงที่เกิดขึ้น ซึ่งเป็นการบอกว่าเราได้ทำอะไรกับโค้ดหรือไฟล์นั้นๆใน Snapshot นี้

### 3.6. Staged และ Unstaged คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Staged และ Unstaged เป็นสถานะของไฟล์ที่ถูกติดตามการเปลี่ยนแปลงใน Git repository ระหว่างการทำงานกับ Git ซึ่งเป็นส่วนหนึ่งของ Git staging area หรือ index ซึ่งเป็นตัวกลางที่ใช้ในการเตรียมไฟล์ที่จะถูก commit เข้า repository โดยที่ Git จะเปรียบเทียบสถานะของไฟล์ใน working directory กับ staging area เพื่อตรวจสอบว่าไฟล์ได้ถูกเปลี่ยนแปลงหรือไม่ โดยมีความหมายดังนี้

### 3.7. Push คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คำว่า "Push" เป็นคำสั่งใน Git ที่ใช้สำหรับการอัพโหลด (upload) การเปลี่ยนแปลงของโค้ดจากเครื่อง local repository ขึ้นไปยัง remote repository บนเว็บโฮสติ้งเซิร์ฟเวอร์ (เช่น GitHub, GitLab, Bitbucket เป็นต้น) เพื่อให้คนอื่นสามารถเข้าถึงและดาวน์โหลด (download) โค้ดเหล่านั้นได้

### 3.8. Pull คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Pull เป็นคำสั่งที่ใช้ในการดึง (fetch) ข้อมูลจาก remote repository มายัง local repository โดยรวมถึงการทำ merge ข้อมูลใน remote repository กับ local repository ด้วย

### 3.9. Fetch คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คำสั่ง Fetch ใน Git คือการดึงข้อมูล (fetch) จาก Repository บน Server มายัง Local Repository โดยไม่ทำการส่งการเปลี่ยนแปลงที่เกิดขึ้นใน Local Repository กลับไปยัง Repository บน Server ด้วย ซึ่งจะมีประโยชน์ในกรณีที่ต้องการดูสถานะปัจจุบันของ Repository บน Server หรือต้องการตรวจสอบว่ามีการแก้ไขหรือเพิ่มไฟล์ใน Repository บน Server หรือไม่

### 3.10. Conflict ใน VSC คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Conflict เกิดขึ้นใน Version Control System (VCS) เมื่อมีการแก้ไขไฟล์เดียวกันในบรรทัดเดียวกันโดยหนึ่งฐานแล้ว Conflict จะเกิดขึ้นเมื่อทั้ง 2 ไฟล์ที่มีการแก้ไขไม่เหมือนกัน และไม่สามารถรวมการแก้ไขได้โดยอัตโนมัติ จึงจำเป็นต้องแก้ไขด้วยมือ หรือใช้เครื่องมือในการรวมการแก้ไข (merge tool) เพื่อรวมการแก้ไขทั้ง 2 และแก้ไข Conflict ให้ถูกต้องตามวิธีการที่เหมาะสม

### 3.11. Merge Commit คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Merge commit เป็นการรวมการเปลี่ยนแปลงของ branch ที่ต่างกันไปในขั้นตอนสุดท้ายของการ merge โดยเรียก Merge commit เนื่องจากเป็นการสร้าง commit ใหม่ขึ้นมา ซึ่งมีคุณสมบัติพิเศษที่เป็นเฉพาะของการ merge โดยเฉพาะ ดังนั้นการ merge commit จึงถือว่าเป็นอีกหนึ่งหัวข้อที่สำคัญในการทำงานกับ Version Control System (VCS) เช่น Git

### 3.12. ขั้นตอนที่อยู่ในระหว่าง Development Process ภายใน VSC มีอะไรบ้าง?

สร้าง branch ใหม่: สร้าง branch ที่ใช้สำหรับพัฒนาฟีเจอร์หรือแก้ไขปัญหาแบบ isolated จาก branch หลัก และทำการ switch ไปยัง branch ใหม่นี้

พัฒนาและทดสอบ: ทำการพัฒนาและทดสอบฟีเจอร์หรือการแก้ไขปัญหาบน branch ใหม่ โดยมักจะใช้วิธีการ commit เพื่อเก็บ snapshot ของการเปลี่ยนแปลงที่ทำใน branch ใหม่นี้ โดยอัตโนมัติ

### 3.13. จงบอกและอธิบายขั้นตอนการติดตั้งส่วนขยายใน Eclipse เพื่อให้ใช้งาน Git

เปิด Eclipse และเลือกเมนู Help จากนั้นเลือก Eclipse Marketplace

ค้นหา Git ในช่องค้นหา และคลิกที่ปุ่ม Go

เลือก Git Team Provider จากผลลัพธ์ที่แสดงออกมา และคลิกที่ปุ่ม Install

ตั้งค่าการติดตั้งโดยกด Next และยอมรับข้อกำหนดและเงื่อนไข จากนั้นคลิก Next

## 4. ลำดับขั้นการปฏิบัติการ

4.1. ลงทะเบียน Github และตกแต่ง Profile ของตนเองให้เรียบร้อย

4.2. สร้าง Repository ใน Github

4.3. ทำการติดตั้งส่วนเสริมของ Git ลงใน Eclipse เพื่อเตรียมใช้งาน Version Control System ของ Github

4.4. การสร้างผลงานโค้ดโปรแกรมใน Github

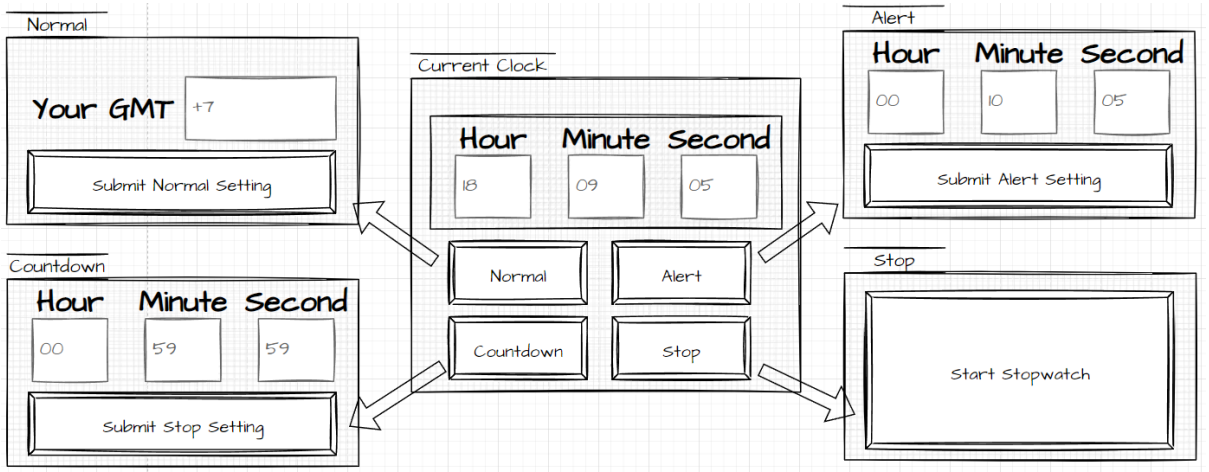
4.4.1. เชื่อมต่อ Eclipse ของคุณเข้ากับ Github

4.4.2. ทำการ Push โค้ดโปรแกรมตั้งแต่การทดลองที่ 1 ถึง 8 ขึ้นสู่ Remote ใน Github ผ่านโปรแกรม Eclipse

| **ลิงค์ Github ที่เก็บไฟล์ข้อมูลของการทดลองที่ 1 ถึง 8 ของคุณ** |
| --- |
| **ลิงค์การทดลองที่ 1 ->**<br>**ลิงค์การทดลองที่ 2 ->**<br>**ลิงค์การทดลองที่ 3 ->**<br>**ลิงค์การทดลองที่ 4 ->**<br>**ลิงค์การทดลองที่ 5 ->**<br>**ลิงค์การทดลองที่ 6 ->**<br>**ลิงค์การทดลองที่ 7 ->**<br>**ลิงค์การทดลองที่ 8 ->** |

4.5. ทำการ Push โค้ดโปรแกรมตั้งแต่การทดลองที่ 1 ถึง 8 ขึ้นสู่ Remote โดยใช้โปรแกรม Eclipse

4.6. สร้างโปรเจคใหม่ใน Eclipse ที่เชื่อมต่อกับ Github ให้เรียบร้อย พร้อมทั้งหาสมาชิกร่วมกลุ่มจำนวน 3-4 คน เพื่อสร้างโปรแกรม “นาฬิกาสารพัดประโยชน์” ที่มีส่วนประกอบของฟีจเจอร์ต่างๆ ดังนี้




4.6.1. หน้าต่าง Current Clock เพื่อแสดงนาฬิกาที่จะทำงานตามโหมดต่างๆ ที่ผู้ใช้สั่งตามปุ่มต่างๆ

4.6.2. หน้าต่าง Normal จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Normal ที่อยู่ในหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้งค่า GMT ให้กับนาฬิกาหลักหลังจากกดปุ่ม Submit Normal Setting เรียบร้อยแล้ว

4.6.3. หน้าต่าง Countdown จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Countdown ที่อยู่ในหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้งค่าการนับเวลาถอยหลัง สามารถปรับค่าได้ในระดับชั่วโมง นาที และวินาที หลังจากกดปุ่ม Submit เรียบร้อย หน้าต่างการตั้งค่าจะหายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะทำการเริ่มต้นนับถอยหลังไปเรื่อยๆ จนถึงเลข 0 นาฬิกา 0 นาที 0 วินาที

4.6.4. หน้าต่าง Alert จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Alert ที่อยู่ในหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้งค่าเวลาปลุกเมื่อเวลาปัจจุบันเดินทางมาถึงเวลาที่กำหนดไว้ สามารถปรับค่าได้ในระดับชั่วโมง นาที และวินาที หลังจากกดปุ่ม Submit เรียบร้อย หน้าต่างการตั้งค่าจะหายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะแสดงเวลาตามปกติ แต่เมื่อถึงเวลาที่ตั้งปลุกเอาไว้ ระบบก็จะปรากฏหน้าต่างแจ้งเตือน

4.6.5. (**หากมีสมาชิกในกลุ่มไม่ถึง 4 คน ไม่ต้องทำฟีจเจอร์นี้**) หน้าต่าง Stop จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Stop ที่อยู่ในหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้งค่าการจับเวลา หลังจากกดปุ่ม Start Stopwatch เรียบร้อย หน้าต่างการตั้งค่าจะหายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะเริ่มต้นจับเวลา โดยเริ่มตั้งแต่ 0 นาฬิกา 0 นาที 0 วินาที และ

จำนวนวินาทีจะเริ่มต้นเพิ่มขึ้นไปเรื่อยๆ จนกว่าผู้ใช้งานจะกดปุ่ม Stop อีกครั้ง เพื่อเป็นการหยุดการทำงานของนาฬิกาจับเวลา

4.7. จากฟีจเจอร์การทำงานของนาฬิกาข้างต้น ให้นักศึกษาแบ่งหน้าที่ในการกับเพื่อนร่วมงานในกลุ่มเพื่อสร้าง Repository และทำงานร่วมกันภายใน Remote นี้

4.7.1. ผู้รับผิดชอบทั้งหมด สร้างและพัฒนาส่วนของ Current Clock

4.7.2. ผู้รับผิดชอบคนที่ 1 สร้างและพัฒนาส่วนของ Normal

4.7.3. ผู้รับผิดชอบคนที่ 2 สร้างและพัฒนาส่วนของ Countdown

4.7.4. ผู้รับผิดชอบคนที่ 3 สร้างและพัฒนาส่วนของ Alert

4.7.5. ผู้รับผิดชอบคนที่ 4 (**ถ้ามี**) สร้างและพัฒนาส่วนของ Stop

4.8. นักศึกษาจะต้องทำงานร่วมกัน เพื่อให้เห็นภาพรวมการใช้งาน Eclipse ร่วมกับ Github ให้มองเห็นการทำงานเพื่อการแยก Branch, การ Merge Branch, การจัดการโค้ดโปรแกรมเมื่อเกิด Conflict

**รายชื่อสมาชิกภายในกลุ่มของคุณ และหน้าที่รับผิดชอบภายในกลุ่ม**

**คนที่ 1 ชื่อ-นามสกุล** ..... **รหัสนักศึกษา** .....
**หน้าที่รับผิดชอบ** .....
**คนที่ 2 ชื่อ-นามสกุล** ..... **รหัสนักศึกษา** .....
**หน้าที่รับผิดชอบ**.....
**คนที่ 3 ชื่อ-นามสกุล** ..... **รหัสนักศึกษา** .....
**หน้าที่รับผิดชอบ** .....
**คนที่ 4 ชื่อ-นามสกุล** ..... **รหัสนักศึกษา** .....
**(ถ้ามี) หน้าที่รับผิดชอบ** .....

**ลิงค์งานกลุ่มของคุณที่อยู่ใน Github**

**ผลลัพธ์การทำงานของโปรแกรม**

```
import java.awt.*;
import java.awt.event.*;
import java.text.*;
import java.util.Date;
import javax.swing.*;
/*
  /*
   To change this template, choose Tools | Templates
   and open the template in the editor.
*/

/*
   AlarmTrigger.java
   Created on Mar 28, 2015, 10:40:19 AM
*/
/**
   @author Sujit Reddy
*/
public class AlarmTrigger extends javax.swing.JFrame
{
```

โค้ดโปรแกรมภายในหน้าต่าง Current Clock

```
int ss, mm, hh;
  private  boolean verify = false;
  /** Creates new form AlarmTrigger */
  public AlarmTrigger() {
    initComponents();
    setTitle("....Alarm System....");
    getContentPane().setBackground(Color.WHITE);
    final DateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");
    ActionListener timerListener = new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        if (!verify)
        {
          jLabel6.setText("Alarm Not Set");
        }
        if (verify)
        {
          jLabel6.setText("Alarm Set to " + hh + ":" + mm + ":" + ss);
        }
        Date date = new Date();
        String time = timeFormat.format(date);
        jLabel2.setText(time);
        int sc = date.getSeconds();
        int mn = date.getMinutes();
        int hr = date.getHours();
        if (sc == ss && mn == mm && hr == hh)
        { System.out.print("Matched ");
          verify = false;

          // Code to Do actions
          Component Jframe = null;
          JOptionPane.showMessageDialog(Jframe, "Hello World", "Alarm Ringing", JOptionPane.PLAIN_MESSAGE);
        }

      }
    };
    Timer timer = new Timer(1000, timerListener);
    // to make sure it doesn't wait one second at the start
    timer.setInitialDelay(0);
    timer.start();

  }
```

โค้ดโปรแกรมภายในหน้าต่าง Normal

```
    private void initComponents() {

        jDialog1 = new javax.swing.JDialog();
        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jPanel4 = new javax.swing.JPanel();
        jLabel7 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
```

โค้ดโปรแกรมภายในหน้าต่าง Countdown

```
javax.swing.GroupLayout jDialog1Layout = new javax.swing.GroupLayout(jDialog1.getContentPane());
    jDialog1.getContentPane().setLayout(jDialog1Layout);
    jDialog1Layout.setHorizontalGroup(
      jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGap(0, 400, Short.MAX_VALUE)
    );
    jDialog1Layout.setVerticalGroup(
      jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGap(0, 300, Short.MAX_VALUE)
    );

    jPanel2.setBackground(new java.awt.Color(255, 255, 255));

    jLabel1.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
    jLabel1.setText("Time:");

    jLabel2.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
    jLabel2.setText("Time");

    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
      jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(69, 69, 69)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 148,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                  .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 150,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 187,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(50, Short.MAX_VALUE))
    );
```

**โค้ดโปรแกรมภายในหน้าต่าง** Alert

**โค้ดโปรแกรมภายในหน้าต่าง** Stop

**5. สรุปผลการปฏิบัติการ**

## 6. คำถามท้ายการทดลอง

6.1. ควร Commit อย่างไร เพื่อหลีกเลี่ยงการเกิด Conflict ให้เหมาะสมที่สุด

1.ดึงข้อมูลล่าสุดจาก repository ก่อนทำการแก้ไขโค้ด

2.แก้ไขโค้ดของคุณบน branch ของคุณโดยเลือกทีจะแก้ไขเพียงไฟล์เดียวในแต่ละครั้งเท่านั้น

3.ทำการ commit เป็นบางส่วนของการทำงานเป็นระเบียบ และควรจะ commit เมื่อคุณได้ทำงานบนโค้ดทีสมบูรณ์แล้ว และไม่ควร commit โค้ดทีเป็นช้วคราว หรือทียังไม่ได้ทดสอบให้เรียบร้อย

6.2. ควรมีหลักเกณฑ์ในการ Push ขึ้นไปบน Remote เมื่อใดจึงจะเหมาะสมที่สุด

1.คุณควร push โค้ดของคุณเมื่อคุณได้ทำการ commit โค้ดทีสมบูรณ์และไม่มีข้อผิดพลาด 2.คุณควร push โค้ดของคุณเมื่อคุณทำการแก้ไขโค้ดใน local repository และต้องการให้ผู้อื่นเห็นการเปลียนแปลงโค้ดของคุณ 3.คุณควร push โค้ดของคุณหลังจากทีคุณได้ทำการ pull โค้ดล่าสุดจาก remote repository เพื่อหลีกเลียง conflict ในการ push 4.คุณควร push โค้ดของคุณเมื่อคุณต้องการให้ผู้อื่นสามารถทดสอบและตรวจสอบโค้ดของคุณก่อนการนำไปใช้งาน 5.คุณควร pushโค้ดของคุณหลังจากทีคุณได้ทำการทดสอบและตรวจสอบโค้ดของคุณให้เรียบร้อยแล้ว

6.3. เมื่อใดจึงควรใช้คำสั่ง Fetch

1.เมื่อคุณต้องการดูว่ามีการเปลียนแปลงอะไรบ้างใน remote repository โดยไม่ต้องทำการอัปเดต local repository ของคุณ เช่น คุณอาจต้องการตรวจสอบว่ามีการ push โค้ดล่าสุดจากสมาชิกในทีมของคุณหรือไม่ 2.เมื่อคุณต้องการดูประวัติการเปลียนแปลงของ remote repository โดยไม่ต้องทำการอัปเดต local repository ของคุณ ดังนั้นคุณสามารถใช้คำสั่ง fetch เพื่อดูประวัติการ commit และการเปลียนแปลงของโค้ดจาก remote repository ได เมื่อคุณต้องการดึง branch ใหม่จาก remote repository มายัง local repository โดยไม่ต้องทำการอัปเดตโค้ดของคุณ โดยคุณสามารถใช้คำสั่ง fetch โดยระบุชื่อ branch ทีต้องการดึงมาในคำสั่งได้

6.4. เราควรจะแยก Branch เมื่อใด? และควรจะ Merge Branch เมื่อใด?

Branch เมื่อ คุณต้องการพัฒนาฟีเจอร์ใหม่: การพัฒนาฟีเจอร์ใหม่โดยตรงบน branch ใหม่จะช่วยให้คุณสามารถแยกกิจกรรมของการพัฒนาจากฟีเจอร์อื่นๆ ทีอยู่บน branch อื่นๆ ทีมีอยู่ ทำให้ง่ายต่อการจัดการโค้ดและการรวมโค้ด

merge branch ควรจะทำเมือ ฟีเจอร์ทีพัฒนาบน branch ใหม่เสร็จสมบูรณ์และผ่านการทดสอบ: เมือฟีเจอร์ทีต้องการรวมเข้ากับ branch หลักเสร็จสมบูรณ์และผ่านการทดสอบแล้ว คุณควรจะ merge branch ใหม่เข้ากับ branch หลักเพือให้ฟีเจอร์นันสามารถใช้งานได้