ใบงานการทดลองที่ 10

เรื่อง การควบคุมเวอร์ชันการทำงานผ่านโปรแกรม Eclipse

1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการติดต่อกับผู้ใช้งาน และการหลายงานพร้อมกัน

1.2. รู้และเข้าใจการติดต่อระหว่างงาน

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

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

Version Control System (VCS) เป็นระบบที่ช่วยในการบริหารจัดการเอกสารหรือโค้ดต่างๆ โดยที่มีการบันทึกประวัติการแก้ไขไฟล์หรือการเปลี่ยนแปลงต่างๆ ที่เกิดขึ้นในโค้ด ทำให้สามารถตรวจสอบและเปรียบเทียบไฟล์หรือโค้ดเวอร์ชันต่างๆ ได้ง่ายขึ้น

3.2. Git ต่างกับ Github อย่างไร?

Git เป็นระบบที่ช่วยจัดการการแก้ไขใน Repository ส่วน GitHub เป็นบริการจัดเก็บ Repository ออนไลน์พร้อมกับฟีเจอร์อำนวยความสะดวกต่าง ๆ ที่ให้เราไปทำงานร่วมกันคนอื่นได้

3.3. Repository คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คือการเก็บสำรองข้อมูลและการเปลี่ยนแปลงของ Source Code ทำให้สามารถย้อนกลับไปที่เวอร์ชั่นใดๆ ก่อนหน้า และดูรายละเอียดการเปลี่ยนแปลงของแต่ละเวอร์ชั่นได้ นอกจากนั้นยังสามารถดูได้ว่าใครเป็นคนแก้ไข

3.4. Clone คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

เวลาที่ผู้อ่านมี Repository อยู่บน Remote ซักแห่งอยู่แล้ว และต้องการ Sync มาลงเครื่องของเรา เราจะต้องทำสิ่งที่เรียกว่า Clone Repository หรือก็คือการก๊อป Repository จาก Remote คือ เวลาที่ผู้อ่านมี Repository อยู่บน Remote ซักแห่งอยู่แล้ว และต้องการ Sync มาลงเครื่องของเรา เราจะต้องทำสิ่งที่เรียกว่า Clone Repository หรือก็คือการก๊อป Repository จาก Remote

3.5. Commit คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

เวลาที่มีข้อมูลที่แก้ไขเสร็จแล้ว (โค้ดที่เขียนคำสั่งบางอย่างเสร็จแล้ว) แล้วอยากจะทำการ Backup เก็บไว้ใน VCS จะเรียกกนว่า Commit

3.6. Staged และ Unstaged คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

เวลาเราแก้ไขโค้ดหรือแก้ไขข้อมูล ไฟล์ที่ถูกแก้ไขจะอยู่ในสถานะ Unstaged และเวลาที่เราทำอะไรเสร็จเรียบร้อย แล้วอยากจะ Commit เก็บไว้ จะต้องเลือกไฟล์ที่ต้องการเพื่อย้ายเข้าสู่ในสถานะ Staged ก่อนถึงจะทำการ Commit ได้

3.7. Push คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

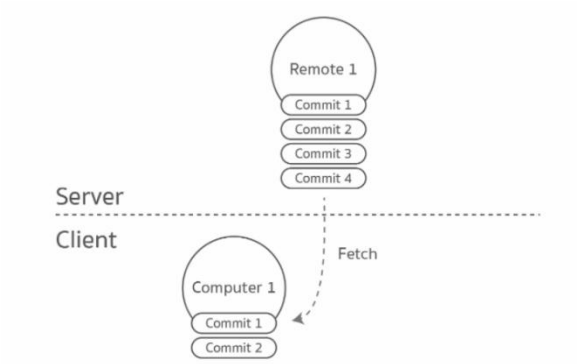
Push คือการนำโค้ดหรือไฟล์เข้าตัวระบบ Git Repository

3.8. Pull คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คือ เวลา Sync จาก Remote เพื่อดึงข้อมูล Commit ใหม่ๆลงมาเก็บไว้ในเครื่องจะเรียกขั้นตอนนี้ว่า Pull

3.9. Fetch คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

อยากเช็คสถานะของ Remote เฉยๆว่ามีใคร Push ข้อมูลใหม่ขึ้นไปที่ Remote หรือป่าว เราเรียกวิธีนี้ว่า Fetch



3.10. Conflict ใน VSC คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คือ การเกิดปัญหาการชนของข้อมูลในไฟล์งานที่ทำร่วมกันกับเพื่อนเรา ซึ่งในช่วงที่เราพัฒนาโปรแกรมหรือขียนโค้ดกับเพื่อนร่วมงานอยู่นั้น เราไม่สามารถรู้ได้เลยว่าเพื่อนเราจะเขียนโค้ดไปในรูปแบบไหน

3.11. Merge Commit คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คือการที่มีการแตก branch ออกไป develop แยกกัน โดยที่มีการแก้ไขไฟล์เดียวกันซึ่งโค้ดนั้นอาจมีการทับซ้อน หรืออยู่บรรทัดเดียวกัน เมื่อใครคนใดคนหนึ่งนำโค้ดมา Merge รวมกันนั้นจะเกิดสิ่งที่เรียกว่า Conflict คือโค้ดของทั้งสองคนมีความขัดแย้งกัน

3.12. ขั้นตอนที่อยู่ในระหว่าง Development Process ภายใน VSC มีอะไรบ้าง?

3.13. จงบอกและอธิบายขั้นตอนการติดตั้งส่วนขยายใน Eclipse เพื่อให้ใช้งาน Git

1.Install Plugin ทำการ Click ไปที่ Help และ Install new software

2.จากนั้นก็พิมพ์ http://download.eclipse.org/egit/updates ลงในช่อง URLแล้วติ๊กที่ Egit

3.หลังจากนั้นกด Next แล้วรออาจใช้เวลานาน รอจนกว่าตัวโปรแกรมจขึ้นให้ restart แล้วเปิดโปรแกรมใหม่หลังจากนั้นก็สามารถใช้ส่วนของ Git ได้เลย

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

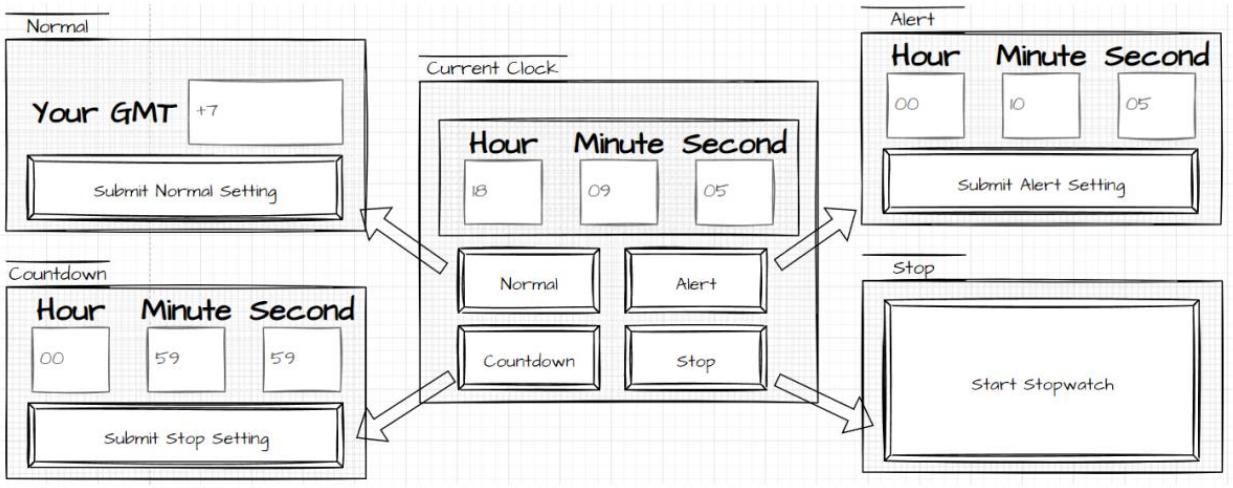
4.4. การสร้างผลงานโค้ดโปรแกรมใน Github

4.4.1. เชื่อมต่อ Eclipse ของคุณเข้ากับ Github

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

4.5. ทำการ Push โค้ดโปรแกรมตั้งแต่การทดลองที่ 1 ถึง 8 ขึ้นสู่Remote โดยใช้โปรแกรม Eclipse

4.6. สร้างโปรเจคใหม่ใน Eclipse ที่เชื่อมต่อกับ Github ให้เรียบร้อย พร้อมทั้งหาสมาชิกร่วมกลุ่มจำนวน 3-4 คน เพื่อสร้าง โปรแกรม “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 จะปรากฏหน้าต่างนี้เมื่อคลกิปุ่ม Countdown ที่อยูใ นหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้ง ค่าการนับเวลาถอยหลัง สามารถปรับค่าได้ในระดับชั่วโมง นาทีและวินาทีหลังจากกดปุ่ม Submit เรียบร้อย หน้าต่างการ ตั้งค่าจะหายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะทำการเริ่มต้นนับถอยหลังไปเรื่อยๆ จนถึงเลข 0 นาฬิกา 0 นาที 0 วินาที

4.6.4. หน้าต่าง Alert จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Alert ที่อยูใ นหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้งค่าเวลาปลุกเมื่อ เวลาปัจจุบันเดินทางมาถึงเวลาที่กำหนดไว้ สามารถปรับค่าได้ในระดับชั่วโมง นาที และวินาที หลังจากกดปุ่ม Submit เรียบร้อย หน้าต่างการตั้งค่าจะหายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะแสดงเวลาตามปกติแต่เมื่อถึงเวลา ที่ตั้งปลุกเอาไว้ระบบก็จะปรากฏหน้าต่างแจ้งเตือน

4.6.5. (หากมีสมาชิกในกลุ่มไม่ถึง 4 คน ไม่ต้องทำฟีจเจอร์นี้) หน้าต่าง Stop จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Stop ที่อยู่ในหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้งค่าการจับเวลา หลังจากกดปุ่ม Start Stopwatch เรียบร้อย หน้าต่างการตั้งค่าจะ หายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะเริ่มต้นจับเวลา โดยเริ่มตั้งแต่ 0 นาฬิกา 0 นาที0 วินาทีและ

จำนวนวินาทีจะเริ่มต้นเพิ่มขึ้นไปเรื่อยๆ จนกว่าผู้ใชง้านจะกดปุ่ม Stop อีกครั้ง เพื่อเป็นการหยุดการทำงานของนาฬิกา จับเวลา

4.7. จากฟีจเจอร์การทำงานของนาฬิกาข้างต้น ให้นักศึกษาแบ่งหน้าที่ในการกับเพื่อนร่วมงานในกลุ่มเพื่อสร้าง Repository และทำ งานร่วมกันภายใน Remote นี้

4.7.1. ผู้รับผิดชอบทั้งหมด สร้างและพัฒนาส่วนของ Current Clock

4.7.2. ผู้รับผิดชอบคนที่ 1 สร้างและพัฒนาส่วนของ Normal

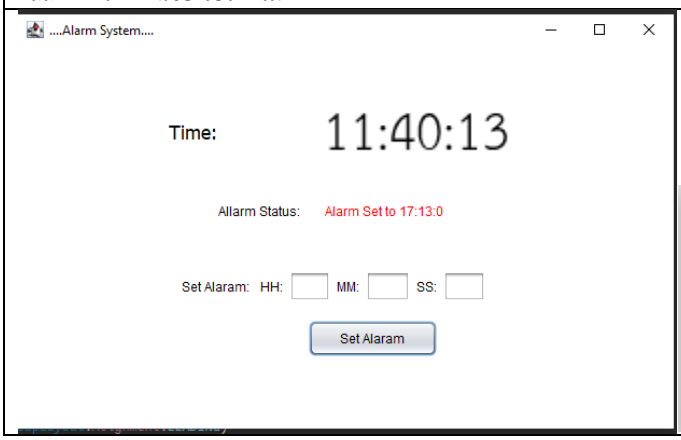
4.7.3. ผู้รับผิดชอบคนที่ 2 สร้างและพัฒนาส่วนของ Countdown

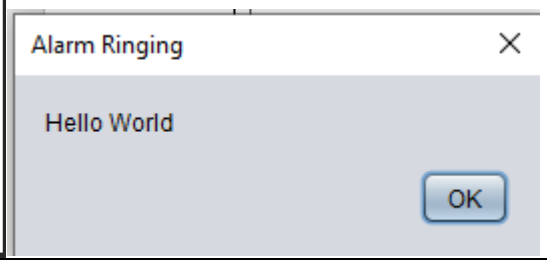
4.7.4. ผู้รับผิดชอบคนที่ 3 สร้างและพัฒนาส่วนของ Alert

4.7.5. ผู้รับผิดชอบคนที่ 4 (ถ้ามี) สร้างและพัฒนาส่วนของ Stop

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

ผลลัพธ์การทำงานของโปรแกรม





โค้ดโปรแกรมภายในหน้าต่าง Current Clock

```
/** 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)
```

โค้ดโปรแกรมภายในหน้าต่าง Normal

```
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 Countdown

```
}
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;
```

โค้ด โปรแกรมภายในหน้าต่าง Alert

```
        // 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();
```

โค้ด โปรแกรมภายในหน้าต่าง Stop

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

5. สรุปผลการปฏิบัติการ

ไม่ได้เชื่อมจาก github กับตัวของ eclipe แต่ทำทุกอย่างใน eclipe หมดเลย อาจจะไม่ค่อยตรงตามแลปที่ให้เชื่อม แต่ตัวของโปรแกรมก็เหมือน

6. คำถามท้ายการทดลอง

6.1. ควร Commit อย่างไร เพื่อหลีกเลี่ยงการเกิด Conflict ให้เหมาะสมที่สุด

ทำส่วนของ Project หรือ ตัวไฟล์งานไว้เลย อย่างรวมไฟล์แล้ว commit ทีเดียว เพราะอาจทำให้การ commit นั้นเกิด การ conflict และอาจทำให้ Pull code มีปัญหาได้

6.2. ควรมีหลักเกณฑ์ในการ Push ขึ้นไปบน Remote เมื่อใดจึงจะเหมาะสมที่สุด

เลือกไฟล์ที่ต้องอัพขึ้น git แล้วหลังจากนั้นค่อย Share project เสร็จเช็คก่อนที่จะไป staged changes ขั้นต่อไปทำการ commit ก่อนแล้วค่อยไป push and commit

6.3. เมื่อใดจึงควรใช้คำสั่ง Fetch

เมื่อต้องการเช็คข้อมูลว่าใครที่ push เข้ามาทำแล้วบ้างเราไม่จำเป็นต้อง pull เข้าเครื่อง fetch ยังสามารถเช็ค history ทั้งหมดได้ด้วย

6.4. เราควรจะแยก Branch เมื่อใด? และควรจะ Merge Branch เมื่อใด?

เมื่อเราจะอัพไฟล์ไปใน Git Branch เพราะเราต่างคนต่างทำโค้ดอาจทำให้เวลาทำ Feature อาจไม่รู้ว่า เป็นของหรืออาจทำให้ Source Code รวมอยู่ในไฟล์เดียวกันได้ เราควร Merge Branch เมื่อมีการจะ push โค้ดเข้าในตัวของ git hub