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?

- ระบบที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์เพื่อที่คุณสามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้ หนังสือเล่มนี้จะยกตัวอย่างจากไฟล์ที่เป็นซอร์สโค้ดของซอฟต์แวร์ แต่ขอให้เข้าใจว่าจริง ๆ แล้วคุณสามารถใช้ version control กับไฟล์ชนิดใดก็ได้

3.2. Git ต่างกับ Github อย่างไร?

- Git คือ Version Control System ส่วน Github บริษัทที่พัฒนาเกี่ยวกับ Git

3.3. Repository คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือ ที่เก็บไฟล์ในลักษณะคลาว แต่ก่อนที่จะน าเข้าไปเก็บได้ต้องได้รับอนุญาตก่อน

3.4. Clone คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือการก็อป Repository จาก Remote มาลงเครื่องเรา

3.5. Commit คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือ การเก็บข้อมูลที่ถูกแก้ไขไว้ใน VCS หรือการ Backup

3.6. Staged และ Unstaged คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- Unstaged คือไฟล์ที่เราไม่ได้น าขึ้นระบบ ส่วน Staged ก็คือไฟล์ที่เรานำขึ้นไปไว้ในระบบ

3.7. Push คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือการพลักเอา Commit ไปไว้ใน Remote ในระบบ

3.8. Pull คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือการ ดึง Commit ที่เราอัพขึ้นระบบลงมาแก้ไข

3.9. Fetch คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือการเช็คข้อมูลก่อนว่าข้อมูลที่เราจะอัพขึ้นไปมีการแก้ไขหรือไม่

3.10. Conflict ใน VSC คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือโค้ดที่ทับซ้อนกัน หรือความขัดแย้งของโค้ด

3.11. Merge Commit คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

- คือการที่เราจะอัพไฟล์ขึ้นไปในระบบแต่มีไฟล์ที่เพื่อนทำไว้อยู่แล้ว เราจึงต้องทำการ Pull ลงมาก่อนเพื่ออัพเดตข้อมูลเก่า

3.12. ขั้นตอนที่อยู่ในระหว่าง Development Process ภายใน VSC มีอะไรบ้าง?

3.13. จงบอกและอธิบายขั้นตอนการติดตั้งส่วนขยายใน Eclipse เพื่อให้ใช้งาน Git

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

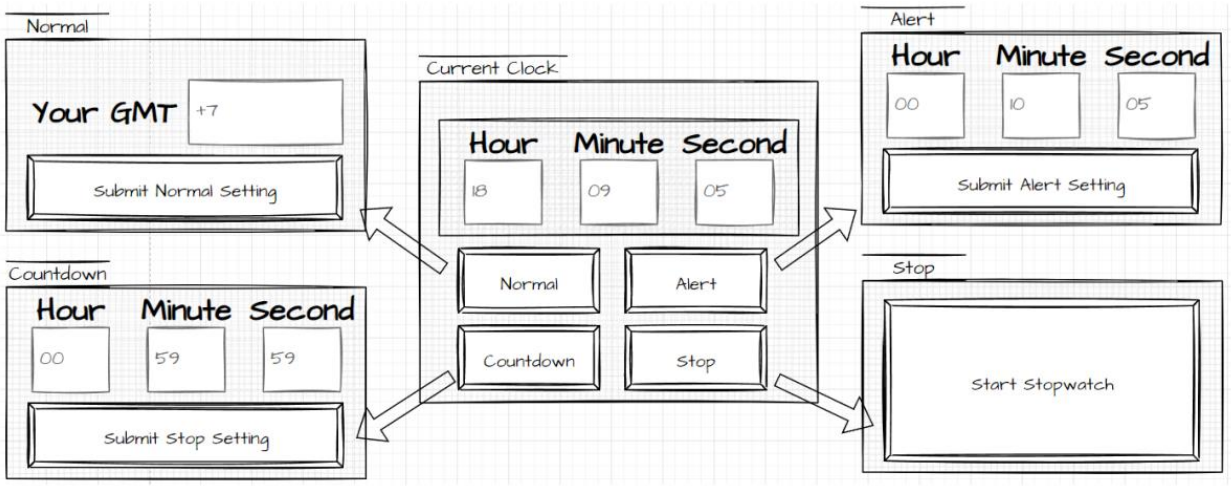
4.4. การสร้างผลงานโค้ดโปรแกรมใน Github

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 พร้อมทั้งหาสมาชิกร่วมกลุ่ม จำนวน 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 ซึ่งจะแสดงส่วนการตั้งค่า GMT ให้กับนาฬิกาหลักหลังจากกดปุ่ม Submit Normal Setting เรียบร้อยแล้ว

4.6.3. หน้าต่าง Countdown จะปรากฏหน้าต่างนี้เมื่อคลกิปุ่ม Countdown ที่อยู่ใ นหน้า Current Clock ซึ่งจะ แสดงส่วนการตั้ง ค่าการนับเวลาถอยหลัง สามารถปรับค่าได้ในระดับชั่วโมง นาทีและวินาที

หลังจากกดปุ่ม Submit เรียบร้อย หน้าต่างการ ตั้งค่าจะหายไป และส่วนการแสดงนาฬิกาใน Current Clock ก็จะทำการเริ่มต้นนับถอยหลังไปเรื่อยๆ จนถึงเลข 0 นาฬิกา 0 นาที 0 วินาที

4.6.4. หน้าต่าง Alert จะปรากฏหน้าต่างนี้เมื่อคลิกปุ่ม Alert ที่อยู่ใ 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

รายชื่อสมาชิกภายในกลุ่มของคุณ และหน้าที่รับผิดชอบภายในกลุ่ม
คนที่ 1 ชื่อ -นามสกุล เขมภาณิน กิติศักดิ์ รหัสนักศึกษา 63543206047-6

ลิงค์งานกลุ่มของคุณอยู่ที่ใน Github
ผลลัพธ์การทำงานของโปรแกรม

โค้ดโปรแกรมภายในหน้าต่าง Current Clock

```
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Button;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.wb.swt.SWTResourceManager;
public class main {
 protected Shell shell;
 private Text tbShowHour;
 private Text tbShowMinute;
 private Text tbShowSecond;
 /*
 * Launch the application.
 * @param args
 */
```

```
 public static void main(String[] args) {
 try {
 main window = new main();
 window.open();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
นาย พีรพัฒน์ ศิริอ้าย
63543206070-8
 /*
 * Open the window.
 */
 public void open() {
 Display display = Display.getDefault();
 createContents();
 shell.open();
 shell.layout();
 while (!shell.isDisposed()) {
 if (!display.readAndDispatch()) {
 display.sleep();
 }
 }
 }
protected void createContents() {
 shell = new Shell();
 shell.setSize(423, 325);
 shell.setText("SWT Application");
 Button btnNormal = new Button(shell, SWT.NONE);
```

```
btnNormal.addSelectionListener(new SelectionAdapter() {
@Override
public void widgetSelected(SelectionEvent e) {


}
});
btnNormal.setBounds(61, 152, 136, 44);
btnNormal.setText("Normal");
Button btnCountdown = new Button(shell, SWT.NONE);
btnCountdown.setBounds(116, 202, 176, 44);
btnCountdown.setText("Countdown");
Button btnAlert = new Button(shell, SWT.NONE);
btnAlert.addSelectionListener(new SelectionAdapter() {
@Override
public void widgetSelected(SelectionEvent e) {
}
});
btnAlert.setBounds(203, 152, 136, 44);
btnAlert.setText("Alert");
Label lblShowTitleHour = new Label(shell, SWT.NONE);
lblShowTitleHour.setBounds(61, 42, 55, 15);
lblShowTitleHour.setText("Hour");
Label lblShowTitleMinute = new Label(shell, SWT.NONE);
lblShowTitleMinute.setBounds(184, 42, 55, 15);
lblShowTitleMinute.setText("Minute");
Label lblShowTitleSecond = new Label(shell, SWT.NONE);
lblShowTitleSecond.setBounds(300, 42, 55, 15);
lblShowTitleSecond.setText("Second");
tbShowHour = new Text(shell, SWT.BORDER);
```

```
 tbShowHour.setBackground(SWTResourceManager.getColor(255, 255, 255));
 tbShowHour.setBounds(42, 63, 76, 60);
 tbShowMinute = new Text(shell, SWT.BORDER);
 tbShowMinute.setBounds(167, 63, 76, 60);
 tbShowSecond = new Text(shell, SWT.BORDER);
 tbShowSecond.setBounds(279, 63, 76, 60);
 setTime();
 }
public void setTime() {
 LocalDateTime localDate = LocalDateTime.now();
 DateTimeFormatter hh = DateTimeFormatter.ofPattern("hh");
 tbShowHour.setText(hh.format(localDate));
 LocalDateTime localDate_mm = LocalDateTime.now();
 DateTimeFormatter mm = DateTimeFormatter.ofPattern("mm");
 tbShowMinute.setText(mm.format(localDate_mm));
 LocalDateTime localDate_ss = LocalDateTime.now();
 DateTimeFormatter ss = DateTimeFormatter.ofPattern("ss");
 tbShowSecond.setText(ss.format(localDate_ss));
 try {
 Thread.sleep(1000);
 } catch (InterruptedException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }
}
```

โค้ดโปรแกรมภายในหน้าต่าง Normal

```
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
```

```
import org.eclipse.swt.widgets.Text;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Label;
import org.eclipse.wb.swt.SWTResourceManager;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
public class currenttime {
protected Shell shell;
private Text txtGmt;
public String txt = "";
/**
* Launch the application.
* @param args
*/
public static void main(String[] args) {
try {
currenttime window = new currenttime();
window.open();
} catch (Exception e) {
e.printStackTrace();
}
}
/**
* Open the window.
*/
public void open() {
Display display = Display.getDefault();
createContents();
```

```
shell.open();
shell.layout();
while (!shell.isDisposed()) {
if (!display.readAndDispatch()) {
display.sleep();
}
}
}
/**
* Create contents of the window.
*/
protected void createContents() {
Text gmt ;
shell = new Shell();
shell.setSize(423, 325);
shell.setText("Normal Setting");
txtGmt = new Text(shell, SWT.BORDER);
txtGmt.setText("GMT+01:00");
txtGmt.setBounds(210, 56, 131, 91);
Label lblYourGmt = new Label(shell, SWT.NONE);
lblYourGmt.setFont(SWTResourceManager.getFont("Segoe UI Historic", 21, SWT.NORMAL));
lblYourGmt.setBounds(52, 82, 151, 46);
lblYourGmt.setText("Your GMT ");
Button btnNewButton = new Button(shell, SWT.NONE);
btnNewButton.addSelectionListener(new SelectionAdapter() {
@Override
public void widgetSelected(SelectionEvent e) {
main form1 = new main();
form1.open();
```

```
}
});
btnNewButton.setBounds(55, 171, 301, 60);
btnNewButton.setText("Submit Normal Setting");
Label lblExexmpleGmt = new Label(shell, SWT.NONE);
lblExexmpleGmt.setBounds(210, 22, 120, 28);
lblExexmpleGmt.setText("Exemple : GMT+02:30");
}
}
```

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

```
    Component Jframe = null;
    JOptionPane.showMessageDialog(Jframe, "Hello World", "Alarm Ringing", JOptionPane.PLAIN_MESSAGE);
  }
```

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

6. คำถามท้ายการทดลอง

6.1. ควร Commit อย่างไร เพื่อหลีกเลี่ยงการเกิด Conflict ให้เหมาะสมที่สุด

- คือการคุยกันกับเพื่อนในกลุ่มให้มากที่สุดเพื่อหลีกเลี่ยง Conflict

6.2. ควรมีหลักเกณฑ์ในการ Push ขึ้นไปบน Remote เมื่อใดจึงจะเหมาะสมที่สุด

- เมื่อเราท าโค้ดเสร็จแล้วบางส่วนแล้วค่อยทยอยอัพขึ้นเรื่อยๆจนเสร็จ

6.3. เมื่อใดจึงควรใช้คำสั่ง Fetch

- ใช้เมื่อเราจะอัพไฟล์ขึ้นไปในระบบ

6.4. เราควรจะแยก Branch เมื่อใด? และควรจะ Merge Branch เมื่อใด?

- เมื่อเวลาที่เราจะทำงานทั้งหมดมารวมกันเพื่อทำการส่งลูกค้า