นาย พีรพัฒน์ ศิริอ้าย
63543206070-8

ใบงานการทดลองที่ 13
เรื่อง การใช้งาน Inner Class และการใช้งาน Thread

1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการติดต่อกับผู้ใช้งาน และการหลายงานพร้อมกัน

1.2. รู้และเข้าใจการติดต่อระหว่างงาน

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

3.1 Nest Class คืออะไร? มีวัตถุประสงค์เพื่ออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Nest Class (หรือ Inner Class) คือการประกาศคลาสภายในคลาสอีกอันหนึ่ง โดยคลาสที่อยู่ภายในจะถือว่าเป็นส่วนประกอบของคลาสภายนอกนั้น ๆ โดย Nest Class จะอยู่ภายใต้ขอบเขตของคลาสภายนอก ซึ่งทำให้สามารถเข้าถึงสมาชิกและเมทอดของคลาสภายนอกได้โดยตรง

3.2. จงยกตัวอย่างการสร้าง Inner Class

```
public class InnerClass { public void printX() {
        System.out.println("x = " + x); }
}
public void createInnerClass() {
        InnerClass inner = new InnerClass();
        inner.printX();
 }}
```

3.3. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Properties ภายใน Inner Class

```
public class OuterClass {
  private int x;
  public class InnerClass {
        public void printX() {
        System.out.println("x = " + x);
}
```

3.4. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Method ภายใน Inner Class

```
public class OuterClass {
  private int x;
  public class InnerClass {
    public void printX() {
    System.out.println("x = " + x);
} }
```

3.5. Thread คืออะไร? มีประโยชน์อย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

เทร็ด คือหน่วยการทำงานย่อยที่อยู่ในกระบวนการ มีการแบ่งปันทรัพยากรต่าง ๆ ในกระบวนการนั้น ๆ จุดประสงค์ของเทร็ดคือการเรียกใช้ซีพียูให้เกิดประโยชน์สูงสุด เทร็ดทำให้การทำงานของโปรแกรมง่าย และมีประสิทธิภาพมากขึ้นและมีประโยชน์ต่อระบบที่มีซีพียูหลายตัว

3.6. การเริ่มต้นใช้งาน Thread มีขั้นตอนอย่างไรบ้าง?

สร้าง Instance ของ Thread object โดยสร้าง object ใหม่ของคลาสที่สืบทอดมาจากคลาส Thread หรือผ่านการสร้าง Inner Class กำหนดชื่อเรียกของ Thread โดยใช้ constructor ของ Thread object สร้างเมธอด run() ที่ประกอบไปด้วยงานที่ต้องการให้ Thread ทำ

3.7. ระหว่าง Thread และ Runnable มีรูปแบบการใช้งานที่เหมือนหรือแตกต่างกันอย่างไร?

Thread และ Runnable เป็นคลาสที่ใช้ในการสร้าง Thread ในภาษา Java โดยมีรูปแบบการใช้งานที่แตกต่างกันอย่างมาก

3.8. สถานะ Deadlock มีลักษณะเป็นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Deadlock เกิดขึนเมื่อสองหรือมากกว่า Thread จะไม่สามารถดำเนินการต่อไปได้เนื่องจากมีการรอคอยกันอยู่ เช่น Thread A รอให้ Thread B ทำงานเสร็จก่อน ในขณะเดียวกัน Thread B ก็รอให้ Thread A ทำงานเสร็จก่อน จนกระทั่งไม่มี Thread ใดสามารถดำเนินการต่อไปได้อีก

4. ลำดับขั้นการปฏิบัติการ

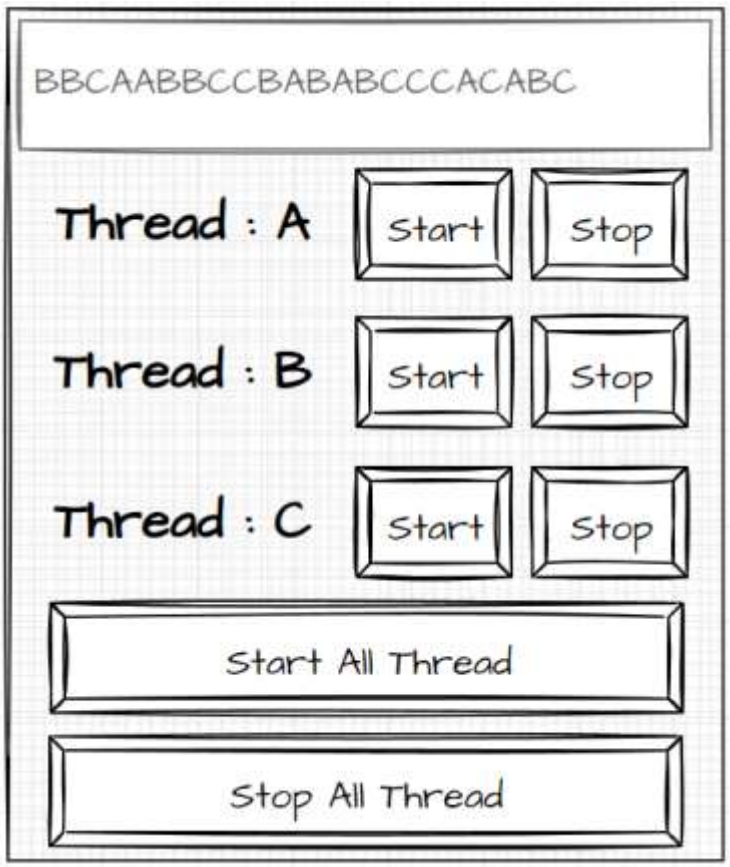
4.1. จงสร้างหน้า GUI เพื่อทำการทดสอบสร้าง Thread ที่มีส่วนประกอบดังต่อไปนี้ สร้าง Repository ใน Github

4.1.2. สร้าง Thread B และ C จาก Class ปกต

4.1.3. แต่ละ Thread จะมีปุ่ม Start เพื่อเริ่มต้นพิมพ์ตัวอักษรของ Thread ลงในช่อง Textbox และ Stop เพื่อหยุดการพิมพ์ตัว อักษรของ Thread ในช่อง Textbox

4.1.4. สร้างปุ่ม Start All Thread เพื่อทำให้ Thread แต่ละตัวทำงานพร้อมกัน

4.1.5. สร้างปุ่ม Stop All Thread เพื่อให้ Thread แต่ละตัวหยุดทำงานพร้อมกัน



นาย พีรพัฒน์ ศิริอ้าย
63543206070-8

5. สรุปผลการปฏิบัติการ

6. คำถามท้ายการทดลอง

6.1. Inner Class แตกต่างจาก Class แบบปกติอย่างไร?

Inner Class เป็น Class ภายใน Class อีกชั้นหนึ่ง ซึ่งมีลักษณะเด่นต่างจาก Class แบบปกติ

6.2. เมื่อใดจึงเป็นช่วงเวลาที่ดีที่สุดในการใช้งาน Inner Class

การใช้งาน Inner Class เหมาะสมมากกับกรณีที่ต้องการใช้งานคลาสภายในคลาสอื่นๆ และมีความสัมพันธ์กันแบบแตกต่างกัน เช่น การใช้งาน Listener ใน GUI ซึ่งส่วนมากจะถูกนำเข้าเป็น Inner Class เพื่อให้สามารถเข้าถึงตัวแปรและเมทอดของ Outer Class ได้ง่าย ๆ และเพิ่มความสะดวกในการจัดการโค้ด

6.3. ข้อควรระวังในการใช้งาน Thread คืออะไร?

Race Condition: เกิดจากการมี Thread ที่แก้ ไขข้อมูลเดียวกันพร้อมกัน อาจทำให้ข้อมูลเสียหายหรือผิดพลาด Deadlock: เกิดจากการรออยู่ระหว่าง Thread 2 หรือมากกว่าและแต่ละ Thread กำลังรอให้ทราบสถานะจาก Thread อื่น ๆ และทั้งหมดติดกัน Starvation: เกิดจากการไม่ได้กำหนดลำดับการประมวลผล Thread ทำให้ Thread บางตัวไม่สามารถทำงานได้ Thread-Safety: เป็นการรักษาความปลอดภัยของการใช้งาน Thread ในรูปแบบต่าง ๆ เพื่อป้ องกันการแชร์ข้อมูลที่ไม่เหมาะสม ซึ่งอาจทำให้ข้อมูลเสียหายหรือผิดพลาด Memory Visibility: เกี่ยวข้องกับการเข้าถึงและอัปเดตข้อมูลในหน่วยความจำที่ใช้ร่วมกันได้ ในกรณีที่ Thread ต่างกันมองเห็นค่าข้อมูลที่แตกต่างกัน อาจทำให้ข้อมูลเสียหายหรือผิดพลาด การระบุและจัดการกับข้อควรระวังดังกล่าวจะช่วยให้การใช้งาน Thread มีประสิทธิภาพและปลอดภัยยิ่งขึ้น