ใบงานการทดลองที่ 13

เรื่อง การใช้งาน Inner Class และการใช้งาน Thread

1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการโปรแกรมเชิงวัตถุ การกำหนดวัตถุ การใช้วัตถุ

1.2. รู้และเข้าใจการทำหลายงานพร้อมกัน

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

3.1. Nest Class คืออะไร? มีวัตถุประสงค์เพื่ออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

คือการประกาศ คลาส ภายใน คลาส อีกคลาส หนึ่ง ซึ่ง คลาส ภายใน จะ ถูกเก็บ ซ่อนอยู่ ไม่ให้ใช้งาน ได้ภายนอก คลาสแม่ มีการประกาศ คลาสภายนอก ซึ่งก็คือ ตัวแม่ ส่วนตัว เราสามารถ เก็บ ตัว คลาสภายใน ได้หรือไม่ คลาส ภายนอก จะประกาศ คลาสภายใน ขึ้นเพื่อใช้ สามารถประกาศ Nest Class ได้ทั้ง static และ non-static ความต่างกันจะขึ้นอยู่กับการใช้งาน

3.2. จงยกตัวอย่างการสร้าง Inner Class

```
public class OuterClass {
    private int x;
    public OuterClass (int x) {
        this.x = x;
    }
    public void display() {
        System.out.println (value of x: "+x);
    }
    public class InnerClass {
        public void print() {
            System.out.println ("Inner Class method called");
        }
    }
}
```

3.3. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Properties ภายใน Inner Class

```
1. สร้าง Outer class และ Inner class
Public class Outer class {
    Private int outerProperty = 1;
    public class InnerClass {
        private int innerProperty = 2;
    }
}
```

```
2. สร้าง Instance ของ Outer class และ Inner class
OuterClass outerInstance = new OuterClass();
OuterClass.InnerClass innerClass innerInstance.
OuterInstance.new InnerClass();
```

```
3. เมื่อต้องใช้งาน Properties ของ Inner class ผ่าน Instance ของ Outer class
int innerPropertyValue = innerInstance.innerProperty;
```

3.4. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Method ภายใน Inner Class

```
1. สร้าง Outer Class และ Inner Class ที่มี Method ภายใน
public class OuterClass {
    private int outerProperty = 1;
    public class InnerClass {
        private int innerProperty = 2;
        public void innerMethod() {
            System.out.println("Inner method call");
        }
    }
}
```

```
2. สร้าง Instance ของ Outer class และ Inner class
OuterClass outerInstance = new OuterClass();
OuterClass.InnerClass innerInstance
= outerInstance.new InnerClass();
```

```
3. เรียกใช้งาน Method
innerInstance.innerMethod();
```

3.5. Thread คืออะไร? มีประโยชน์อย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Thread คือ สิ่ง ที่ใช้งานภายในโปรแกรม สำหรับ ทำงานหลายๆ อย่าง พร้อมกัน หรือ สามารถ ใช้งานในรูปแบบ หลายงาน

ประโยชน์ของ Thread 1. แบ่งปันทรัพยากรการทำงาน หน่วยความจำ

2. สามารถ ใช้งานได้ หลายงาน ได้พร้อมกัน

3. ทำให้โปรแกรม มีความ ยืดหยุ่น

4. สนองต่อ งานผู้ใช้งาน

5. จัดการกับ งานที่ทำ ซ้ำๆ ได้ง่าย ขึ้น

ตัวอย่างการใช้งาน

```
public class MyThread extends Thread {
    public void run() {
        // อธิบายการทำงานของ Thread ตัวนี้
    }
}
public class Main {
    public static void main (String[] args) {
        MyThread thread = new MyThread();
        thread.start();
    }
}
```

3.6. การเริ่มต้นใช้งาน Thread มีขั้นตอนอย่างไรบ้าง?

1. สร้าง class โดย extends จาก คลาส Thread หรือ implement จาก interface Runnable
2. Override เมธอด run() เพื่อระบุโค้ด ที่ต้องการให้ Thread ทำงาน
3. สร้าง object ของ Thread จาก class ที่สร้างขึ้น
4. เรียกเมธอด start() เพื่อเริ่มการทำงานของ Thread

3.7. ระหว่าง Thread และ Runnable มีรูปแบบการใช้งานที่เหมือนหรือแตกต่างกันอย่างไร?

1. Thread เป็น class ที่สามารถสร้าง instance ของ Thread ขึ้นมา เพื่อนำไปใช้งาน Method ต่างๆ สามารถจัดการ task พร้อมกัน แต่ข้อเสีย หากคลาสนี้ถูกใช้งานไปแล้ว ไม่สามารถใช้ซ้ำได้อีก
2. Runnable คือ interface ที่ต้องสร้าง class ขึ้นมาเพื่อ Override method run() เพื่อส่งไปยัง background ในการจัดการ เช่นเดียวกับ Thread สามารถจัดการ task พร้อมกัน และสามารถ reuse object ได้

3.8. สถานะ Deadlock มีลักษณะเป็นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Deadlock เกิดขึ้นเมื่อมีหรือมากกว่า Thread ที่พยายามใช้งาน Resource ที่ถูกล็อกโดย Thread อื่นๆ โดยที่ไม่ยอมปล่อย Resource ของตนให้กับ Thread อื่นๆ และ Thread เหล่านี้ก็ไม่สามารถทำงานต่อไปได้ ส่งผลให้เกิดสถานการณ์ที่เรียกว่า Deadlock ขึ้น เช่น Thread หนึ่งถือครอง Resource A และไม่ยอมปล่อย Resource ให้กับ Thread อื่นๆ

4. ลำดับขั้นการปฏิบัติการ

4.1. จงสร้างหน้า GUI เพื่อทำการทดสอบสร้าง Thread ที่มีส่วนประกอบดังต่อไปนี้

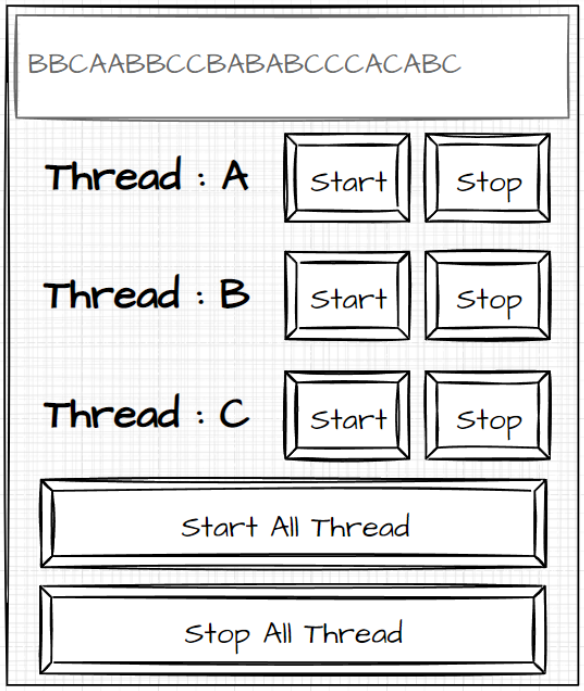
4.1.1. สร้าง Thread A ที่สร้างจาก Inner Class

4.1.2. สร้าง Thread B และ C จาก Class ปกติ

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

4.1.4. สร้างปุ่ม Start All Thread เพื่อทำให้ Thread แต่ละตัวทำงานพร้อมกัน

4.1.5. สร้างปุ่ม Stop All Thread เพื่อให้ Thread แต่ละตัวหยุดทำงานพร้อมกัน



โค้ดโปรแกรมของปุ่ม Start และ Stop ของ Thread A

โค้ดโปรแกรมของปุ่ม Start และ Stop ของ Thread B

โค้ดโปรแกรมของปุ่ม Start และ Stop ของ Thread C

โค้ดโปรแกรมของปุ่ม Start All Thread

โค้ดโปรแกรมของปุ่ม Stop All Thread

5. สรุปผลการปฏิบัติการ

---

6. คำถามท้ายการทดลอง

6.1. Inner Class แตกต่างจาก Class แบบปกติอย่างไร?

- Inner class คือ class ภายใน class ซึ่งๆ ที่อยู่ภายในนี้ เพื่อให้จบภายใน class ที่ตัวเอง และใช้ Inner class จะไม่สามารถเข้าถึงได้ภายนอก class ที่ประกาศ จนกว่าจะถูกประกาศ เป็น public หรือ protected ออกมา
- ข้อดีของ Inner class สามารถจัดกลุ่ม class แบบปกติ แต่ต้องอยู่ภายใน class ที่ต้องการใช้งาน และสามารถเข้าถึง Properties และ Method ของ class ที่ครอบ Outer class ได้โดยตรง

6.2. เมื่อใดจึงเป็นช่วงเวลาที่ดีที่สุดในการใช้งาน Inner Class

การใช้งาน Inner class ที่เหมาะสมควรที่จะใช้ตอนที่ใช้งานเฉพาะเกี่ยวข้องเพื่อให้ภายในส่วนย่อย พร้อมโปรแกรม โดยไม่ต้องเปิดเผยตลอดทั้งไฟล์ ซึ่งทำให้ลดการพึ่งพาโปรแกรม หรือให้ลดการเข้าถึงที่ไม่จำเป็น สามารถใช้ Inner class ที่เหมาะสมกับกรณีที่ต้องการ ส่งผลเฉพาะเกี่ยวข้องให้ภายในส่วนย่อยโปรแกรมเท่านั้น ไม่กระทบ ถึงเพื่อน เขียนสะดวก ทำให้ง่าย กับการดูแลรักษาโปรแกรม หรือต้องใช้ตัวแปร ที่ในส่วนที่พร้อมโปรแกรมอัตโนมัติ

6.3. ข้อควรระวังในการใช้งาน Thread คืออะไร?

1. Race condition: เกิดจากที่การเข้าถึงข้อมูลในหน่วยความจำที่ใช้ร่วมกับโดย Thread หลายๆ ตัว แล้วเกิดการเปลี่ยนแปลงข้อมูลพร้อมกัน ที่ไม่ได้ตั้งใจ
2. Deadlock: เกิดขึ้นเมื่อ Thread หลายๆ ถ้าแต่ต้องรอซึ่งกันและกัน ที่ต่างๆ ที่ต้องการใช้
3. Starvation: เกิดขึ้นเมื่อ Thread ที่มีความสำคัญต่ำกว่า ถูกเข้าถึงทรัพยากรช้ากว่าปกติ
4. Memory consistency Error: เกิดขึ้นเมื่อเกิดข้อมูลที่ไม่ตรงกันในหน่วยความจำ เมื่อมีการใช้งาน concurrency ทำให้เกิดความผิดพลาดขึ้นได้
5. Context Switching Overhead: การสลับการทำงานระหว่าง Thread ทำให้เกิด overhead ในการเปลี่ยน context และการเก็บ state ของ Thread แต่ละตัว
6. Performance การใช้งาน Thread ที่ไม่ถูกต้อง เกิดผลกระทบต่อประสิทธิภาพ หรือทำให้ใช้ทรัพยากรไม่คุ้มค่า ทำให้ประสิทธิภาพ ของระบบ ปฏิบัติการ เกิดช้าลง ที่ไม่ควร