

## ใบงานการทดลองที่ 13

### เรื่อง การใช้งาน Inner Class และการใช้งาน Thread

#### 1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการโปรแกรมเชิงวัตถุ การกำหนดวัตถุ การใช้วัตถุ

1.2. รู้และเข้าใจการทำหลายงานพร้อมกัน

#### 2. เครื่องมือและอุปกรณ์

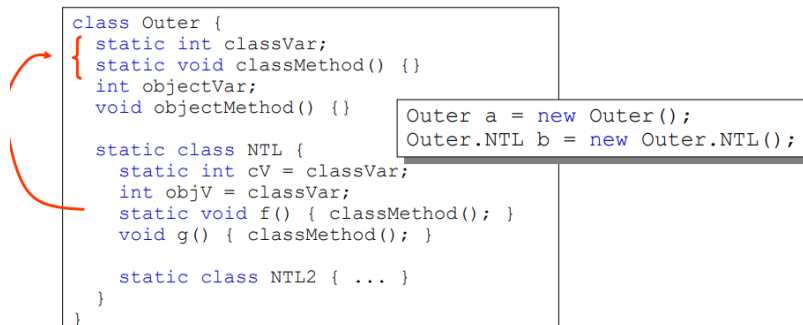
เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

#### 3. ทฤษฎีการทดลอง

3.1. Nest Class คืออะไร? มีวัตถุประสงค์เพื่ออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

#### Nested top-level classes

- อยู่ใน top-level หรือ nested top-level
- มี static กำกับ class
- อ้างอิง static members ของ outer class ได้เท่านั้น
- มีสมาชิกได้ทั้งแบบ static และ non-static



```
class Outer {
    static int classVar;
    static void classMethod() {}
    int objectVar;
    void objectMethod() {}

    static class NTL {
        static int cV = classVar;
        int objV = classVar;
        static void f() { classMethod(); }
        void g() { classMethod(); }

        static class NTL2 { ... }
    }
}
```

```
Outer a = new Outer();
Outer.NTL b = new Outer.NTL();
```

### 3.2. จงยกตัวอย่างการสร้าง Inner Class

- เริ่มจาก Java 2 ภาษา Java ยอมให้มีการกำหนดคลาสขึ้นภายในคลาส เรียกว่า inner classes ซึ่งมี 4 ประเภท ดังนี้

1. Static Inner Classes คือ static class ที่ซ่อนอยู่ในคลาสอื่น

```
// StaticInnerTest.java

package com.mycomp.inner;

class A {

    static class B {

        void f() { System.out.println("Hello"); }

    }

}

public class StaticInnerTest {

    public static void main(String args[]) {

        A.B b = new A.B();

        b.f();

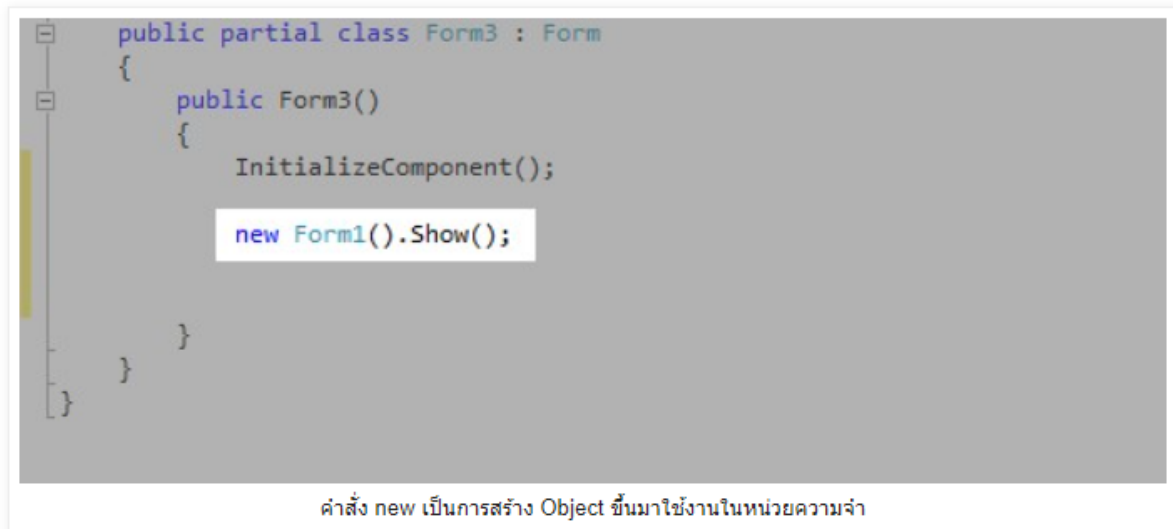
    }

}
```

### 3.3. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Properties ภายใน Inner Class

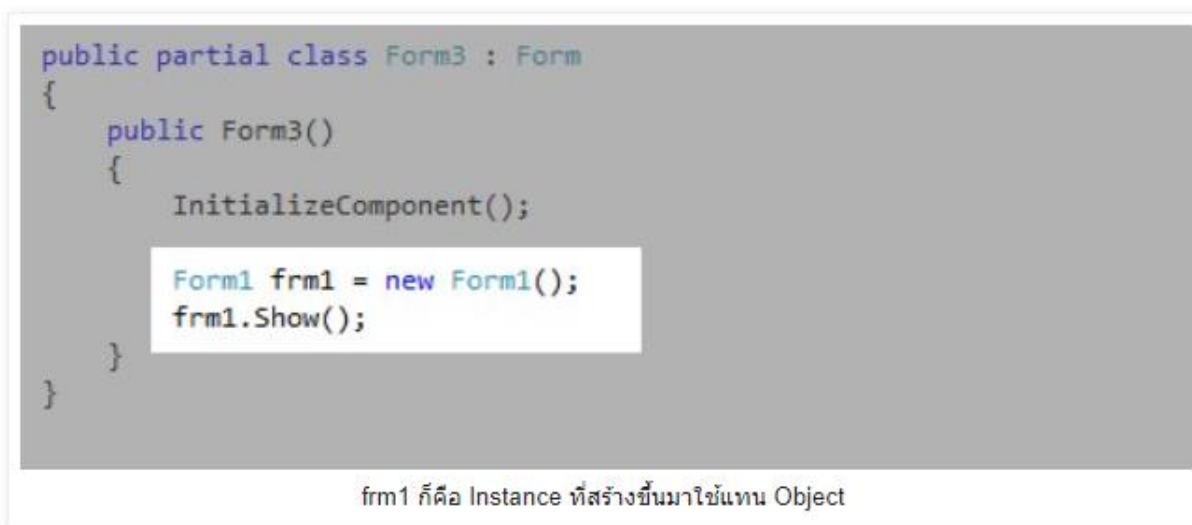
- ในการเขียนโปรแกรมแบบ OOP นั้น แน่นอนว่าต้องเจอคำว่า Object กันจนแทบจะเบื่อไปเลย บางทีอาจจะติดปากเรียกโน่น นี่ นั่นว่า Object ในชีวิตประจำวัน Object นั้นก็คือวัตถุชิ้นหนึ่งๆ ที่เราจับต้องได้ในชีวิตประจำวัน แต่ถ้าในทาง Programming แล้ว มันก็คือวัตถุเสมือน ที่สร้างขึ้นมาจาก Class ซึ่งมีคุณสมบัติและความสามารถ จะมากหรือน้อยก็ขึ้นอยู่กับ Class ที่สร้างมันขึ้นมา

การเรียกใช้ Object โดยตรงอย่างใน C# หรือ Java ก็เป็นอีกสิ่งหนึ่งที่เห็นได้อย่างชัดเจน



### 3.4. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Method ภายใน Inner Class

- ส่วน Instance นั้นเปรียบเสมือนตัวแทนของ Object แต่เราก็สามารถนำ Instance ไปใช้เพื่อเป็นตัวแทนอ้างอิงถึง Class ได้เช่นเดียวกัน ฉะนั้นเมื่อประกาศ Instance ก็คือ Type หนึ่งนั่นเอง



### 3.5. Thread คืออะไร? มีประโยชน์อย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

- Thread คือ หน่วยการทำงานย่อยที่อยู่ใน process มีการแบ่งปันทรัพยากรต่างๆ ใน process นั้นๆ โดยปกติ Process ที่มี 1 thread จะเรียกว่า Single thread หรือเรียกว่า Heavy Weight Process ซึ่งมักพบใน OS รุ่นเก่า แต่ถ้า 1 process มีหลาย thread จะเรียกว่า Multithread หรือ Light Weight Process ซึ่งพบได้ใน OS รุ่นใหม่ที่ใช้กันในปัจจุบันทั่วไป และ Multithread ก็เป็นที่นิยมมากกว่า Single thread

#### ประโยชน์ของมัลติเทรด ( Multi-Threads)

1. การตอบสนอง (Response) ในเรื่องของการทำงานมีการตอบสนองที่ดีกับผู้ใช้ (user) ถ้าการทำงานของ โปรแกรมประยุกต์ของผู้ใช้นั้นมีบางส่วนภายในโปรเซสถูกบล็อกหรือใช้เวลานานเกินไป OS ก็ยังสามารถจัดสรรให้งานอื่น ๆ ภายในโปรเซสนั้นประมวลผลต่อไปได้

2. การใช้ทรัพยากรร่วมกัน (Share Resource) สามารถใช้โค้ด (code) ใช้โปรแกรม (application) และใช้หน่วยความจำ (memory) ร่วมกันระหว่างโปรเซสเดียวกันได้
3. ประหยัด (Economic) ประหยัดการใช้หน่วยความจำในการทำงานของโปรเซส เนื่องจากแต่ละ Thread มีการใช้หน่วยความจำของโปรเซสร่วมกัน
4. ด้านโครงสร้างของมัลติเทรด (Multithread Architecture) การเอื้อประโยชน์ด้านโครงสร้างระบบ ที่งานย่อยภายในโปรเซสให้สามารถทำงานร่วมกัน ประสานจังหวะการทำงานและใช้ทรัพยากรของโปรเซสร่วมกันได้

### 3.6. การเริ่มต้นใช้งาน Thread มีขั้นตอนอย่างไรบ้าง?

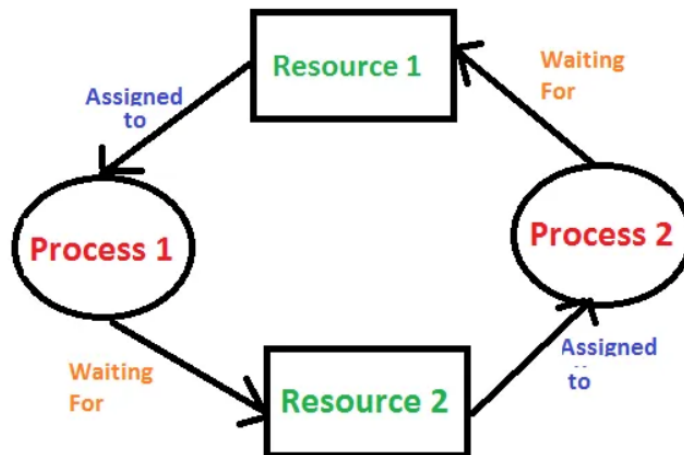
- ในเรื่องของการทำงานมีการตอบสนองที่ดีกับผู้ใช้ (user) ถ้าการทำงานของ โปรแกรม ประยุกต์ของผู้ใช้นั้นมีบางส่วนภายในโปรเซสถูกบล็อกหรือใช้เวลามากเกินไป OS ก็ยังสามารถจัดสรรให้งานอื่น ๆ ภายในโปรเซสนั้นประมวลผลต่อไปได้

### 3.7. ระหว่าง Thread และ Runnable มีรูปแบบการใช้งานที่เหมือนหรือแตกต่างกันอย่างไร?

- Thread คือการเรียกใช้ CPU ให้เกิดประโยชน์สูงสุด Thread ทำให้การทำงานของ โปรแกรมง่าย และมีประสิทธิภาพมากขึ้นและมีประโยชน์ต่อระบบที่มี multi CPU หรือ multi core เพราะสามารถเรียกใช้ thread หลายๆ ตัวได้พร้อมๆ กัน โดย thread แต่ละตัว ของโปรเซส เดียวกันจะทำงานแตกต่างกันแต่มีความเกี่ยวข้องกันบางอย่างและต้องทำงานอยู่ภายใต้ สภาพแวดล้อมเดียวกัน
- Runnable - เป็นสถานะที่เรียกว่าพร้อมทำงาน โดยมันจะเป็นสถานะที่เกิดจากเมื่อ Thread ทำงานเสร็จแล้วหรือถูก Blocked. Blocked - สถานะนี้จะเกิดจากการที่ Thread นั้นอยู่ใน สถานะ Running แล้วเราสั่ง Blocked ซึ่งอาจเกิดขึ้น โดยการเรียกใช้ Method wait(),suspend(),sleep()

### 3.8. สถานะ Deadlock มีลักษณะเป็นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

- สถานะ Deadlock หรือ สถานะติดตาย ก็คือสถานะที่กลุ่ม process หนึ่ง โดนบล็อกรการทำงาน โดยมีสาเหตุมาจาก ในกลุ่ม process นั้น มี process ตัวหนึ่งรอรับการใช้ทรัพยากรจากระบบ แต่ทรัพยากรที่ต้องการนั้นถูก process ตัวอื่นใช้งานอยู่ ทำให้เกิดการรอกอย ซึ่งสามารถอธิบายการทำงานได้คร่าวๆ คือ



#### 4. ลำดับขั้นการปฏิบัติการ

4.1. จงสร้างหน้า GUI เพื่อทำการทดสอบสร้าง Thread ที่มีส่วนประกอบดังต่อไปนี้

4.1.1. สร้าง Thread A ที่สร้างจาก Inner Class

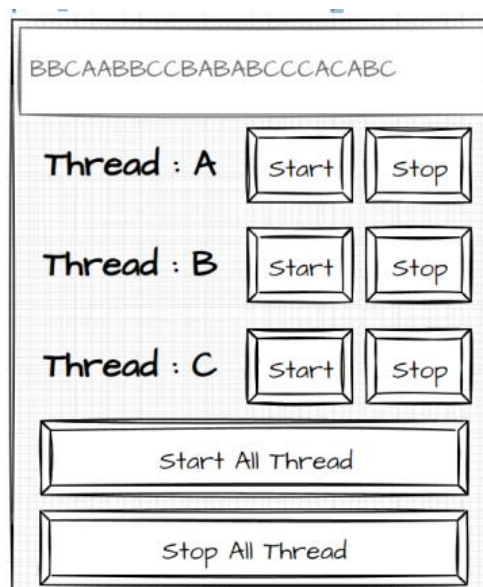
4.1.2. สร้าง Thread B และ C จาก Class ปกติ

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

อักษรของ Thread ในช่อง Textbox

4.1.4. สร้างปุ่ม Start All Thread เพื่อให้ Thread แต่ละตัวทำงานพร้อมกัน

4.1.5. สร้างปุ่ม Stop All Thread เพื่อให้ Thread แต่ละตัวหยุดทำงานพร้อมกัน



## 6. คำถามท้ายการทดลอง

### 6.1. Inner Class แตกต่างจาก Class แบบปกติอย่างไร?

- กรณีนี้ คลาส B เป็น static inner class ของคลาส A และเราเรียกคลาส A ว่าเป็น outer class เราใช้ static inner class เป็น name space อีกระดับหนึ่งต่อจาก package ลงไป ทำให้ต้องอ้างถึงคลาส B ที่อยู่ในคลาส A อย่าง static ด้วย A.B เมื่อถูก compile แล้วคลาส B จะมีไฟล์เป็น A\$B.class Static inner classes จะซ่อนลึกลงไปอีกระดับก็ได้ Static classes ต้องเป็น inner class อยู่ใน outer class ใดคลาสหนึ่ง ไม่สามารถเป็น top level class

### 6.2. เมื่อใดจึงเป็นช่วงเวลาที่ดีที่สุดในการใช้งาน Inner Class

- มันมีประโยชน์อยู่บ้างในการสร้าง Nested class เช่นเมื่อเรามีตัวแปรที่เป็น private ในคลาสแม่ เราสามารถที่จะเข้าถึงตัวแปรนี้ได้ผ่านคลาสที่เป็น InnerClass สาเหตุที่เข้าถึงได้นี้เนื่องจาก InnerClass มี NestedClass เป็นคลาสแม่ (InnerClass อยู่ใน NestedClass)

### 6.3. ข้อควรระวังในการใช้งาน Thread คืออะไร?

- เรื่องราวเกี่ยวกับ thread มีหลายอย่างที่ต้องพิจารณา การยกเลิก (Cancellation) thread เป็นเรื่องที่ต้องทำความเข้าใจ เพราะการยกเลิกหมายถึงการทำให้ thread เป้าหมายจบการทำงานก่อนที่จะทำงานจนเสร็จสมบูรณ์ การยกเลิกนี้มี 2 วิธี

Asynchronous cancellation การยกเลิกที่ thread อื่น สั่งให้ thread เป้าหมายหยุดทำงาน

Deferred cancellation การยกเลิก thread เป้าหมาย โดยใช้ตรวจสอบตนเอง ว่าตนเองต้องถูกยกเลิกด้วยหรือไม่