ใบงานการทดลองที่ 13

เรื่อง การใช้งาน Inner Class และการใช้งาน Thread

นางสาว เบญจภร
62543502007-

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

เป็น Class ที่ประกาศภายใน body ของ Class หรือ Interface อื่นๆจุดประสงค์หลักของการสร้าง Nested Classes คือการ group Classและ Interface ที่เกี่ยวข้องกันให้อยู่ภายใน File เดียวกัน ถึงแม้ว่าการทำPackage ก็ช่วยในเรื่องดังกล่าวแล้วแต่การทำ Nested Classes ทำให้การ group แข็งแรงมากขึ้นอีกขั้นขั้น

```java
public class OuterClass {

    static int number = 10 ;

    static class InnerClass {
        void printData() {
            System.out.println(number);
        }
    }
    public static void main(String[] args) {
        OuterClass.InnerClass outerClass = new OuterClass.InnerClass();
        outerClass.printData();
    }

}
```

3.5. Thread คืออะไร? มีประโยชน์อย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

3.2. จงยกตัวอย่างการสร้าง Inner Class

```
Package lab13;

abstract class Myclass<T>{
        abstract T add(T num, T num 2);
}
Public class JavaExample {
        Public static void main(String[ ] args) {
                MyClass<Integer> obj = new MyClass<>() {
                        Integer add(Integer x, Integer y) {
                                Return x+y;
                        }

                };
                Integer sum = obj.add(100,101)
                System.out.println(sum);
        }
}
```

นางสาว เบญจภรณ์ ไชยเสนา
62543502007-2 คอบ.คพ

3.3. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Properties ภายใน Inner Class

```
public static void main(String[] args) {
        OuterClass.InnerClass outerClass = new OuterClass.InnerClass();
        outerClass.test += 10 ;
}
```

3.4. จงยกตัวอย่างการเรียกใช้งาน Instance ที่มีการเรียกใช้งาน Method ภายใน Inner Class

```
public static void main(String[] args) {
        OuterClass.InnerClass outerClass = new OuterClass.InnerClass();
        outerClass.printData();
    }
}
```

3.5. Thread คืออะไร? มีประโยชน์อย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Thread คือระบบของจาวาสำหรับการสนับสนุนการทำงานแบบ multi-tasking แบบที่ในระบบปฏิบัติการก็จะให้โปรแกรมสามารถทำงานพร้อมกันได้ เช่น ฟังเพลงไปด้วยพิมพ์งานไปด้วยก็ได้ นอกจากนี้เธรดยังสามารถทำงานพร้อมกันได้ด้วยเรียกว่า multi-threa
ประโยชน์จาก Thread นั้นโปรแกรมจะต้องเป็นแบบ Multithreading ซึ่งจะมีข้อได้เปรียบ เช่น มีการตอบสนองของโปรแกรมที่ดีกว่า การประมวลผลเร็วกว่า ใช้ทรัพยากรน้อยกว่า การใช้ประโยชน์จากระบบมากกว่า และการทำงานแบบขนาน

```java
public class ThreadExample {
    public static void main(String[] args){
        Thread t1 = new Thread(new MyThread());
        t1.start();
    }
}
class MyThread implements Runnable {
    @Override
    public void run() {
        System.out.println("Thread is running...");
    }
}
```

3.6. การเริ่มต้นใช้งาน Thread มีขั้นตอนอย่างไรบ้าง?

3.7. ระหว่าง Thread และ Runnable มีรูปแบบการใช้งานที่เหมือนหรือแตกต่างกันอย่างไร?

Thread คือระบบของจาวาสำหรับการสนับสนุนการทำงานแบบ multi-tasking แบบที่ในระบบปฏิบัติการก็จะให้โปรแกรมสามารถทำงานพร้อมกันได้ เช่น ฟังเพลงไปด้วยพิมพ์งานไปด้วยก็ได้ นอกจากนี้เธรดยังสามารถทำงานพร้อมกันได้ด้วยเรียกว่า multi-threa ประโยชน์จาก Thread นั้นโปรแกรมจะต้องเป็นแบบ Multithreading ซึ่งจะมีข้อได้เปรียบ เช่น มีการตอบสนองของโปรแกรมที่ดีกว่า การประมวลผลเร็วกว่า ใช้ทรัพยากรน้อยกว่า การใช้ประโยชน์จากระบบมากกว่า และการทำงานแบบขนาน

3.8. สถานะ Deadlock มีลักษณะเป็นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Thread เป็นคลาสในแพ็คเกจ java.lang คลาสเธรดขยายคลาสของ วัตถุ และใช้อินเทอร์เฟซ Runnable คลาส Thread มีตัวสร้างและวิธีการในการสร้างและดำเนินการกับเธรด Runnable เป็นอินเตอร์เฟสในแพ็คเกจ java.lang การใช้อินเตอร์เฟสที่เรียกใช้งานได้นั้นเราสามารถกำหนดเธรดได้ ส่วนต่อประสานที่ รัน ได้มีวิธีการเดียว รัน () ซึ่งนำมาใช้โดยคลาสที่ใช้ส่วนต่อประสาน Runnableมันเป็นที่ต้องการที่จะใช้อินเตอร์เฟซที่เรียกใช้แทนการขยายชั้นเรียนด้วย เนื่องจากการใช้ Runnable ทำให้โค้ดของคุณเชื่อมโยงกันอย่างหลวม ๆ เนื่องจากโค้ดของเธรดต่างจากคลาสที่กำหนดงานให้กับเธรด มันต้องใช้หน่วยความจำน้อยลงและยังช่วยให้ชั้นเรียนที่จะรับช่วงชั้นอื่น ๆ

นางสาว เบญจภรณ์ ไชยเสนา
62543502007-2 คอบ.คพ

```java
class Philosopher implements Runnable {
    private Object leftFork;
    private Object rightFork;
    public Philosopher(Object leftFork, Object rightFork) {
        this.leftFork = leftFork;
        this.rightFork = rightFork;
    }
    private void doAction(String action) throws InterruptedException {
        System.out.println(Thread.currentThread().getName() + " " + action);
        Thread.sleep((int)(Math.random() * 100));
    }
    @Override
    public void run() {
        try {
            while (true) {
                doAction(System.nanoTime() + ": Thinking");
                synchronized (leftFork) {
                    doAction(System.nanoTime() + ": Pick up left fork");
                    synchronized (rightFork) {
                        doAction(System.nanoTime() + ": Pick up right fork");
                        doAction(System.nanoTime() + ": Eating");
                        doAction(System.nanoTime() + ": Put down right fork");
                    }
                    doAction(System.nanoTime() + ": Put down left fork");
                }
            }
        }
        catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            return;
        }
    }
}
```

นางสาว เบญจภรณ์ ไชยเสนา
62543502007-2 คอบ.คพ

```
Philosopher 2 799755593967100: Thinking
Philosopher 5 799755594082500: Thinking
Philosopher 4 799755594052000: Thinking
Philosopher 1 799755594026800: Thinking
Philosopher 3 799755594026900: Thinking
Philosopher 5 799755611764400: Pick up left fork
Philosopher 5 799755626762700: Pick up right fork
Philosopher 2 799755647782800: Pick up left fork
Philosopher 4 799755657771400: Pick up left fork
Philosopher 3 799755665771100: Pick up left fork
Philosopher 5 799755728791900: Eating
Philosopher 5 799755780778200: Put down right fork
Philosopher 1 799755841786100: Pick up left fork
Philosopher 5 799755841782100: Put down left fork
Philosopher 5 799755873770200: Thinking
Philosopher 4 799755873775600: Pick up right fork
Philosopher 4 799755953765400: Eating
Philosopher 4 799755999778800: Put down right fork
Philosopher 4 799756045799200: Put down left fork
Philosopher 5 799756045819900: Pick up left fork
Philosopher 3 799756133790800: Pick up right fork
```

```
Start
Threadouter outer = new Threadouter();
Threadouter.ThreadA threadA = outer.new ThreadA() ;
threadA.start();
public class Threadouter {
 public class ThreadA extends Thread {
 Threadlab window = new Threadlab();
 int count = 0;
 boolean state = true;
 public void stateA() { state = false; }
 public void stateAstart() { state = true; }
 public void run() {
 while( state ) {
 String text;
this.window.text = text + "A" ;
 System.out.print( this.window.text );
 try {
 TimeUnit.SECONDS.sleep(1);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
 }
 }
 }
Stop
threadA.stateA();
```

โค้ดโปรแกรมของปุ่ม Start และ Stop ของ Thread B

```
Start
ThreadB threadB = new ThreadB();
threadA.start();
public class ThreadB extends Thread {
 Threadlab window = new Threadlab();
boolean state = true;
public void stateB() { state = false; }
public void run() {
 while( state ) {
 this.window.text = window.text + "B" ;
 System.out.print( this.window.text );
 try {
 TimeUnit.SECONDS.sleep(1);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
 }
}
Stop
threadB.stateB();
```

โค้ดโปรแกรมของปุ่ม Start และ Stop ของ Thread C

```
Start
ThreadC threadC = new ThreadC();
threadC.start();
public class ThreadC extends Thread {
 Threadlab window = new Threadlab();
boolean state = true;
public void stateC() { state = false; }
public void run() {
 while( state ) {
 this.window.text = window.text + "C" ;
 System.out.print( this.window.text );
 try {
 TimeUnit.SECONDS.sleep(1);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
 }
}
Stop
threadC.stateC();
```

โค้ดโปรแกรมของปุ่ม Start All Thread

```
threadA.start();
threadB.start();
threadC.start();
```

โค้ดโปรแกรมของปุ่ม Stop All Thread

```
threadA.stateA();
threadB.stateB();
threadC.stateC()
```

5. สรุปผลการปฏิบัติการ

การใช้งาน thread นั้นเป็นการทำงานแบบขนานที่ทำงานหลายๆคำสั่งพร้อมๆกันโดยที่ไม่ต้องทำงานเป็นลำดับ งานใดทำเสร็จก่อนก็ทำการ return ก่อน

6. คำถามท้ายการทดลอง

6.1. Inner Class แตกต่างจาก Class แบบปกติอย่างไร?

การใช้งาน thread นั้นเป็นการทำงานแบบขนานที่ทำงานหลายๆคำสั่งพร้อมๆกันโดยที่ไม่ต้องทำงานเป็นลำดับ งานใดทำเสร็จก่อนก็ทำการ return ก่อน

6.2. เมื่อใดจึงเป็นช่วงเวลาที่ดีที่สุดในการใช้งาน Inner Class

หาก code เริ่มที่จะซับซ้อนและจำเป็นที่จะต้องสร้างอีก class แต่ไม่อยากทำไฟล์แยก

6.3. ข้อควรระวังในการใช้งาน Thread คืออะไร?

คำสั่งที่จะป้อนให้ thread นั้นจำเป็นที่จะต้องมีจุดสิ้นสุดไม่ deadlock