ใบงานการทดลองที่ 12
เรื่อง การใช้งานคำสั่ง try catch และ throw exception

1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการใช้วัตถุ การทำหลายงานพร้อมกัน และการติดต่อระหว่างงาน

1.2. รู้และเข้าใจการจัดการกับความผิดปกติในการเขียนโปรแกรมเชิงวัตถุ

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

3.1. Java Exception คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Java Exception คือ สิ่งที่เกิดขึ้นเมื่อโปรแกรมทำงานผิดพลาดหรือมีปัญหาต่างๆ ที่ทำให้โปรแกรมไม่สามารถทำงานต่อได้ตามปกติ โดยส่วนใหญ่จะเกิดจากข้อผิดพลาดของโค้ดภายในโปรแกรม ซึ่งอาจจะเกิดขึ้นได้ในหลายสถานการณ์ เช่น เมื่อเปิดไฟล์ไม่ได้, มีการอ่านหรือเขียนข้อมูลล้มเหลว หรือทำงานกับอุปกรณ์ฮาร์ดแวร์ไม่ได้ เป็นต้น

3.2. คำสั่ง try มีลักษณะการทำงานอย่างไร?

คำสั่ง try ใช้สำหรับเข้ารหัสโค้ดที่เป็นไปได้ว่าอาจเกิดข้อผิดพลาด (exception) ขึ้น โดยรอบคำสั่งที่มีโค้ดเกี่ยวกับการเรียกใช้หรือประมวลผลข้อมูลที่อาจเป็นต้นเหตุของ exception และจัดการกับ exception โดยใช้คำสั่ง catch ที่ตามหลัง หาก exception ถูกเกิดขึ้นจะทำการเข้ารหัสโค้ดที่เขียนใน block ของคำสั่ง catch เพื่อจัดการแก้ไขหรือแสดงข้อความตอบกลับสำหรับผู้ใช้งาน

3.3. คำสั่ง catch มีลักษณะการทำงานอย่างไร?

คำสั่ง catch เป็นคำสั่งที่ใช้ในการจัดการ Exception หลังจาก Exception เกิดขึ้นแล้ว โดย catch จะรับ Exception ที่ถูก throw ออกมาจาก try block แล้วจัดการกับ Exception นั้นๆ โดยการทำงานของ catch จะมีลักษณะดังนี้

3.4. คำสั่ง finally มีลักษณะการทำงานอย่างไร?

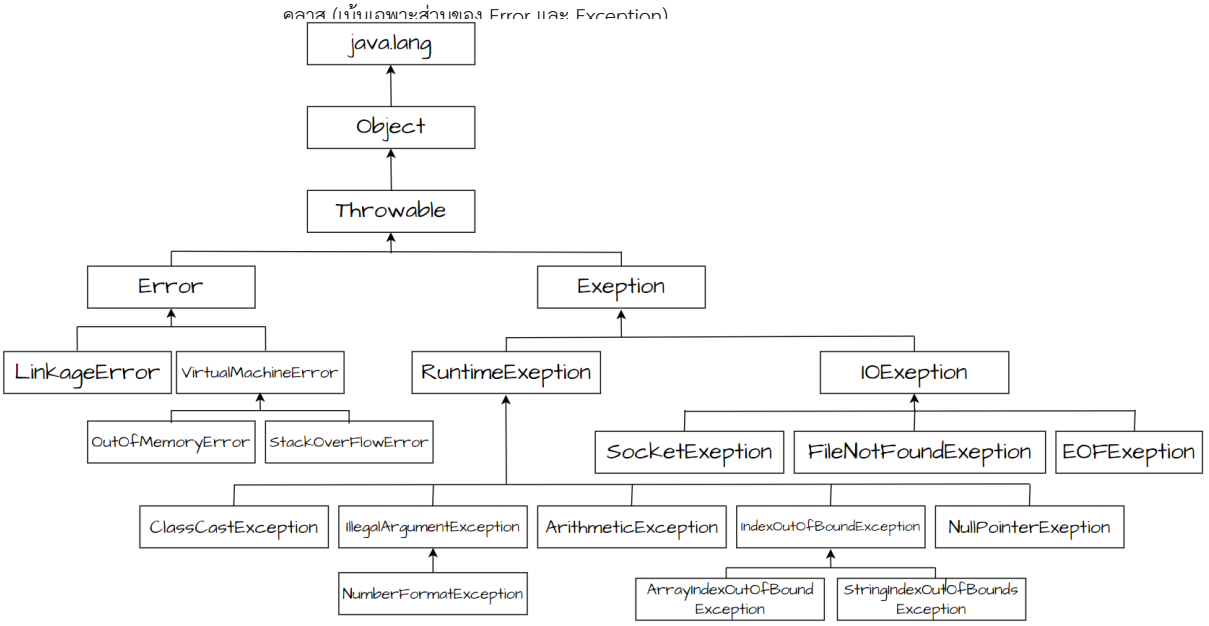
คำสั่ง finally ในภาษา Java จะถูกใช้เพื่อระบุบล็อกโค้ดที่ต้องการให้ทำงานเสมอ ไม่ว่าจะมีการเกิด Exception หรือไม่ก็ตาม โดยที่บล็อกโค้ดในคำสั่ง finally จะถูกทำงานทุกครั้งที่โปรแกรมทำงานถึงจุดสุดท้ายของบล็อก try-catch โดยไม่ว่าจะเกิด Exception หรือไม่ก็ตาม

3.5. ลักษณะโครงสร้างของคำสั่ง try catch เป็นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

```
try {
    // ส่วนของโปรแกรมที่อาจเกิด Exception
} catch (ExceptionType1 e1) {
    // การจัดการ ExceptionType1
} catch (ExceptionType2 e2) {
    // การจัดการ ExceptionType2
} finally {
    // คำสั่งที่ต้องการให้ทำเสมอ ไม่ว่าจะมี Exception เกิดขึ้นหรือไม่ก็ตาม
}
```

4. ลำดับขั้นการปฏิบัติการ

4.1. จากผังงานต่อไปนี้ จงเขียนโค้ดโปรแกรมเพื่อแสดงตัวอย่างการจัดการความผิดปกติของคลาสการจัดการสิ่งผิดปกติจนครบทุกคลาส (เว้นแต่เฉพาะส่วนของ Error และ Exception)



ตัวอย่างโค้ดโปรแกรมการจัดการสิ่งผิดปกติในส่วนของ Error

```
public static BigDecimal addOne(Object obj){
                        BigDecimal a = (BigDecimal)obj;
                        return a.add(BigDecimal.ONE);
                }

                public static void a() { b(); }
                public static void b() { c(); }
                public static void c() {
                        throw new IllegalStateException("Just a test");
                }


                private static void createConnection() throws Exception,
IOException {

        String host = null;
                        int port = 0;
                        Socket socket = new Socket(host, port);

                }

                private static void initiateIO() throws IOException {

        Socket socket = null;
                        PrintWriter outbound = new
PrintWriter(socket.getOutputStream(), true);

                }

                public static void check(int i){

        if (i == 0)
                return;
        else {
           check(i++);
        }
    }
```

ตัวอย่างโค้ดโปรแกรมการจัดการสิ่งผิดปกติในส่วนของ Exeption

```
}catch(VirtualMachineError e) {
                        System.out.println("  This is
VirtualMachineError");
                }// End try..catch ---| VirtualMachineError

                    try {
                            int arrSize = 15;
                    long memoryConsumed = 0;

                    long[] memoryAllocated = null;
                    for (int loop = 0; loop < Integer.MAX_VALUE;
loop++) {
                       memoryAllocated = new long[arrSize];
                       memoryAllocated[0] = 0;
                       memoryConsumed += arrSize * Long.SIZE;
                       arrSize *= arrSize * 2;
                       Thread.sleep(100);
                            }
                       }catch(OutOfMemoryError e) {
                            System.out.println("   ----| This is
OutOfMemoryError");
                       }// End try..catch ---| OutOfMemoryError

                       try {
                            check(5);
                       }catch(StackOverflowError e) {
                            System.out.println("   ----| This is
StackOverflowError");
                       }// End try..catch ---| StackOverflowError
```

5. สรุปผลการปฏิบัติการ

}

}---

---

6. คำถามท้ายการทดลอง

6.1. เพราะเหตุใดการใช้ catch(Exception e) ; จึงไม่เหมาะสมกับการจัดการสิ่งผิดปกติที่ดีที่สุด

การใช้ catch(Exception e) เพื่อจัดการกับสิ่งผิดปกติในโปรแกรม ถือว่าไม่เหมาะสมเนื่องจากมันจะจัดการกับ Exception ทุกประเภทที่อาจเกิดขึ้นในโปรแกรม ซึ่งอาจทำให้เกิดปัญหาที่ไม่คาดคิดได้ เช่น การดักจับ Exception บางอย่างที่ไม่ได้เกิดขึ้นจริง หรือการแยกแยะประเภทของ Exception และจัดการกับมันอย่างเหมาะสมไม่ได้

6.2. การจัดการสิ่งผิดปกติจากการตัวเลขต่างๆ ด้วยเลขศูนย์ ควรเลือกใช้วิธีใด?

การใช้ Exception: วิธีนี้จะเหมาะสมกับกรณีที่ต้องการจัดการกับข้อผิดพลาดที่เกี่ยวกับตัวเลขที่มีค่าเป็นศูนย์ โดยใช้ Exception เช่น ArithmeticException เพื่อจัดการกับการหารด้วยศูนย์ หรือการใช้งานตัวแบ่งเป็นศูนย์

6.3. การจัดการสิ่งผิดปกติจากการเรียกใช้งาน Element เกินขนาดของอาเรย์ ควรเลือกใช้วิธีใด?

Exception Handling เพื่อจัดการกับข้อผิดพลาดที่อาจเกิดขึ้นได้อย่างถูกต้องและปลอดภัยมากที่สุด โดยการใช้ try-catch หรือ throws เพื่อจัดการ Exception ที่อาจเกิดขึ้นได้