2.5 **public Fraction divide(Fraction f) – หากส่วนเป็น 0 ให้ร้องว่า “error – zero by division”**

ข้อนี้มีปัญหานิดหน่อย ตรงที่ ถ้าหารด้วย 0 แล้วเราพิมพ์ออกไป เราก็ยังไม่รู้อยู่ดีว่าจะรีเทิร์นอะไร เพราะหารด้วย 0 มันรีเทิร์นค่าไม่ได้

ดังนั้น จริงๆแล้ว ถ้าเกิดกรณีนี้ขึ้นมา สิ่งที่เราน่าจะทำที่สุดคือ ไม่ต้องให้โปรแกรมรีเทิร์นค่าอะไรเลย สั่งให้โปรแกรม error แล้วหยุดการทำงาน พร้อมทั้งรายงานว่าเกิดการหารด้วย 0 ขึ้น น่าจะเป็นวิธีที่ดีที่สุด

ดังนั้นโค้ดที่เขียน ควรจะมีรูปแบบดังนี้

```
public Fraction divide(Fraction f) throws Exception{
…
…
If(…){ // ถ้าเราตรวจเจอการหารด้วย 0
        throw new Exception(“error – zero by division”);
}
return …
}
```

บอกจาว่า ว่าเมธอดนี้อาจจะเกิด สิ่งผิดปกติตอนรันได้ ในที่นี้อาจจะมีการโยนก้อนวัตถุที่สร้างจากคลาส Exception (ซึ่งเป็นคลาสที่ใช้สำหรับให้เราสร้าง error ขึ้นด้วยตัวเราเอง)

ถ้าจะมีการสร้างก้อนวัตถุ error ขึ้นเอง ในเมธอด เราต้องบอกจาวาด้วยวิธีนี้ ไม่งั้นจาวาไม่ยอมให้คอมไพล์

สร้างก้อนวัตถุ exception แล้วโยนออกมานอกโค้ด โปรแกรมจะหยุดทำงานที่บรรทัดนี้ แล้วจะเริ่มได้ใหม่ตรงโค้ดส่วนที่ดักจับก้อน exception นี้ไว้ แต่ถ้าไม่มีอะไรจับไว้เลย ก้อน exception ก็จะถูกโยน ออกนอกวงเล็บปีกกาไปเรื่อยๆ จนถูกโยนออกนอกวงเล็บปีกกาของเมธอด main

ถ้าตัว exception ถูกโยนออกจาก main แล้ว โปรแกรมจะไม่มีทางรันต่อไป มันจะพิมพ์ “error -……” ตามที่เรานิยามตอนสร้าง exception ออกมาหน้าจอแล้วเลิกการทำงาน ในเมธอดที่เรียก divide ใช้ ถ้าไม่มีการดักจับ exception ด้วย try-catch แล้วละก็ เมธอดนั้นต้องนิยามที่หัวตัวมันเองให้มี throws Exception ด้วย

ลองเขียนแล้วรัน main เพื่อทดสอบดู ว่าตอน error ทำอย่างที่เราต้องการหรือไม่

การเขียน JUnit เพื่อทดสอบโปรแกรมแบบนี้ นอกจากกรณีทดสอบธรรมดาแล้ว ต้องมีกรณีสำหรับเวลาเกิด exception ด้วย ดังรูปแบบดังนี้ (ตัวอย่างนี้ไม่ปัดเป็นเศษส่วนอย่างต่ำนะ)

ต้องมีไว้ เพราะการเรียก divide ครั้งเดียวก็เกิด error ได้แล้ว เลยต้องบอกจาวาไว้

```
public void testDivide() throws Exception {
            Fraction f1 = new Fraction(1,2);
            Fraction f2 = new Fraction(7,8);
            Fraction f3 = f1.divide(f1);

            // ใส่test อื่นๆไปนะ

            Fraction f5 = new Fraction(0,4);

            try{
                  Fraction f6 = f1.divide(f5);  //expect error ตรงนี้
                  fail();    //ถ้าดันรันบรรทัดนี้ได้ แสดงว่า error ไม่ได้เกิดตามที่เราคาด การ test ควรจะ fail
                             //  ดังนั้นจึงสั่งเลย ให้ fail

            }catch (Exception e){
                  // Do nothing, the test will continue to run.
                  // We expect an exception, so the test should be allowed
                  // to progress.

            }
```

จากตรงที่เกิด exception การรันจะมาต่อ ตรงที่ catch exception ชนิดที่เกิดได้ ดังนั้นโค้ดจะต้อง jump จากบรรทัดที่มี f1.divide(f5) มาที่บรรทัดภายในปีกกานี้เลย

ถ้ามาตรงนี้ได้ แสดงว่า exception เกิดขึ้นจริง ตามที่เราคาดไว้ เราจึงไม่ต้องทำอะไร ปล่อย JUnit รันต่อไปได้เลย