BigInteger

BigInteger Class in Java

- เวลาที่เราดำเนินการทางคณิตศาสตร์ที่เป็นการคำนวณเลขจำนวนเต็มที่มีขนาดใหญ่มากๆๆ เช่น 50!

- เลขจำนวนเต็มที่มีขนาดใหญ่มากๆ เหล่านี้เกินขอบเขตของชนิดข้อมูลพื้นฐานใน C/C++
- ใน Python ไม่มีปัญหา ส่วน Java มี class ที่เรียกว่า BigInteger มาช่วยจัดการ

- ในหัวข้อนี้เราจะมาดูว่าถ้าใช้ Java ทำอย่างไร

ตัวอย่างการหา factorial ของจำนวนใหญ่ๆ

- Factorial ของจำนวนที่มากกว่า 13 data type int ก็ไม่สามารถหาค่าได้เนื่องจาก overflow เพราะว่า factorial เหล่านี้มีขนาดใหญ่ แม้ว่าเราจะเปลี่ยนเป็น long ก็ได้ประมาณ 20! ก็ overflow อยู่ดี

- ในการหาค่า factorial ของจำนวนที่ใหญ่มากๆ เราจะใช้ BigInteger class ซึ่งอยู่ใน package java.math

- ตัวอย่างต่อไปเป็นการหา factorial โดยใช้ BigInteger

```java
import java.math.BigInteger;
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        BigInteger fact = factorial(n);
        System.out.println("Factorial is " + fact);
   }

   public static BigInteger factorial(int n) {
        BigInteger fact = new BigInteger("1");
        for (int i = 1; i <= n; i++) {
                  //fact = fact.multiply(new BigInteger(i + ""));
                  fact = fact.multiply(BigInteger.valueOf(i));
        }
        return fact;
   }
}
```

- BigInteger class ถูกใช้แทนจำนวนใหญ่ๆ โดยที่ overflow จะไม่เกิดขึ้นเหมือนใน int long
- BigInteger object นั้นถูกสร้างโดยการผ่าน string ที่เป็นค่า integer ไปให้

- BigInteger fact = new BigInteger("1"); เป็นการสร้าง fact ให้มีค่าเป็น 1 หรือสร้างด้วยวิธี

  BigInteger fact = BigInteger.ONE;

- ข้อสังเกตใน loop คล้ายกับ factorial program ทั่วไป
- การคูณกันของ BigInteger นั้นทำได้โดยการเรียกฟังก์ชัน multiply ซึ่งจะนำเอา BigInteger เป็น argument
- สิ่งที่ต้องระวัง BigInteger class เป้น immutable นั่นหมายความว่า object ที่เรียก multiply function ไม่สามารถเปลี่ยนค่าที่มันเก็บได้ ดังนั้นเมื่อการคูณถูกเรียก จะมี BigInteger ตัวใหม่คืนค่ามาซึ่งเราจะเอามาเก็บในตัวแปรใหม่
- `fact = fact.multiply(new BigInteger(i + ""));`

- แต่ถ้าเราเรียกเพียง
- fact.multiply(new BigInteger(i + ""));

- เราจะได้ 1
- เพราะว่าเริ่มต้นมันถูกกำหนดค่าเป็น 1 และการคูณหลังจากนั้นไม่มีผลกับค่าที่มันเก็บไว้นั่นเอง

Primitive arithmetic ของ BigInteger

- การประกาศ
  - Int a, b;
  - BigInteger A, B;
- การกำหนดค่า
  - a = 24;
  - b = 28;
  - A = BigInteger.valueOf(24);
  - B = BigInteger.valueOf(28);

- นอกจากนี้ยังสามารถกำหนดค่าเริ่มต้นโดยใช้ Integer ที่เป็น string ได้ด้วย
  - A = BigInteger.valueOf("24");
  - B = BigInteger.valueOf("12345679812345679 8");
- ค่าคงที่บางค่านิยามตอนกำหนดค่าได้
  - A = BigInteger.ONE;

- Mathematical operations
  - int c = a + b;
  - BigInteger C = A.add(B);
  - นอกจาก add() ก็มี subtract(), multiply(), divide(), remainder() ซึ่งฟังก์ชันเหล่านี้ใช้ BigInteger เป็น argument ดังนั้นถ้า type ที่ส่งมาไม่ใช่ก็แปลงก่อน
  - String s = "123456798";
  - BigInteger C = A.add(new BigInteger(s));
  - int v = 123456;
  - BigInteger C = A.add(BigInteger.valueOf(v));

- การนำค่าออกจาก BigInteger
  - int x = A.intValue(); //ต้องระวัง size ของ int x
  - long y = A.longValue(); //ต้องระวัง size ของ long y
  - String z = A.toString();
- การเปรียบเทียบ
  - if (a < b){ } //a b เป็น int
  - if (A.compareTo(B) < 0) { } //A B เป็น BigInteger
  - การเปรียบเทียบจะคืนค่า –1 เมื่อน้อยกว่าเท่ากับ
  - 0 เมื่อเท่ากับ
  - 1 เมื่อมากกว่า

- แล้วถ้าเราอยากวน loop ด้วย BigInteger หละ
- เอาข้างต้นมารวมกัน

```
for (BigInteger bi = BigInteger.valueOf(5);
        bi.compareTo(BigInteger.ZERO) > 0;
        bi = bi.subtract(BigInteger.ONE)) {

    System.out.println(bi);
}
```