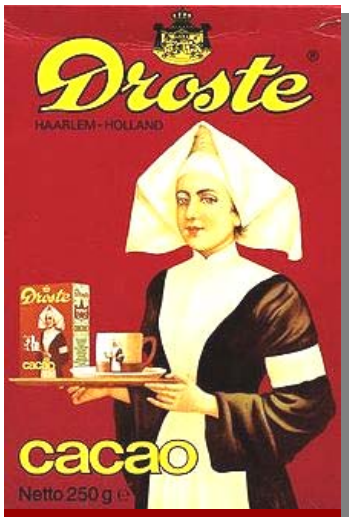
CHAPTER 11 การเวียนเกิด (Recursion)

1. ความรู้เบื้องต้นเกี่ยวกับการเวียนเกิด (Introduction to Recursion)

1. **นิยามของการเวียนเกิด**

   1) **การเวียนเกิด** (Recursion) คือ ปรากฏการณ์ที่มีการวนกลับไปอ้างอิงถึงตัวเองซ้ำแล้วซ้ำเล่าแบบไม่รู้จบ ซึ่งมีปรากฏอยู่ในหลายๆ วงการ เช่น ศิลปะ ดนตรี คณิตศาสตร์ และรวมถึงคอมพิวเตอร์อีกด้วย ในภาษาจาวานั้นการเวียนเกิดจะเกิดขึ้นได้กับเมท็อด

   2) **เมท็อดแบบเวียนเกิด** (Recursive Method) คือ เมท็อดที่เรียกใช้เมท็อดตัวเองซ้ำภายในเมท็อดนั้น (จะทำหน้าที่เป็นทั้งผู้เรียกและผู้ที่ถูกเรียกในเวลาเดียวกันกับเมท็อดตัวเดียวกัน) ซึ่งจะมีกระบวนการทำงานทุกอย่างเหมือนกับการวนซ้ำ (Iteration) ในบทที่ 4 ทุกประการแต่จะไม่มีคำสั่ง `while` หรือ `for` ที่กล่าวไว้ในบทที่ 4 ปรากฏอยู่เลย

   Droste Effect คือภาพที่ปรากฏการเวียนเกิดภายในภาพ

   3) **การเขียนโปรแกรมแบบเวียนเกิด** (Recursive Programming) คือ การเขียนโปรแกรมโดยให้เมท็อดใดเมท็อดหนึ่งหรือหลายเมท็อด เรียกใช้เมท็อดตัวเองซ้ำภายในเมท็อดนั้นไปเรื่อยๆ จนกว่าจะหลุดพ้นจาก เงื่อนไขหรือสิ้นสุดการเวียนเกิด

2. **ขั้นตอนการทำงานของเมท็อดแบบเวียนเกิด** จะประกอบไปด้วย 2 กรณี คือ

   1) **กรณีเวียนเกิด** (Recursion Case) เป็นกรณีที่เมท็อดคืนค่าที่เกิดจากการเรียกใช้งานเมท็อดตัวมันเอง เช่น `return fac(n-1)`, `return fibo(n-1) + fibo(n-2)` เป็นต้น ซึ่งเสมือนเป็นการเรียกซ้ำเมท็อดตัวเองอีกครั้งอยู่ภายในเมท็อดนั้น โดยเมท็อดจะเรียกซ้ำแบบนี้ไปเรื่อยๆ จนกว่าจะเข้าสู่กรณีฐานจึงจะหยุดการเรียกซ้ำ

   2) **กรณีฐาน** (Base Case) เป็นกรณีสิ้นสุดการเวียนเกิด หรือกรณีที่เมท็อดสามารถทำงานเสร็จได้ทันทีโดยไม่ต้องมีการเรียกซ้ำอีก ซึ่งเมท็อดจะคืนค่าคงที่ค่าใดค่าหนึ่งออกมา เช่น `return 1`, `return n` เป็นต้น

3. **ตัวอย่างเมท็อดแบบเวียนเกิด**

   กำหนดให้เมท็อด `f(…)` ใช้หาผลบวกของตัวเลขจำนวนเต็มตั้งแต่ 0 จนถึง n โดยที่ `f(n) = (0+1+2+ … +n)`

   ใช้หลักการวนซ้ำ (Iteration: **`for`**)

```
1  public static int f(int n) {
2    int s = 0;
3    for(int i = n; i >= 0; i--) {
4      s += i;
5    }
6    return s;
7  }
```

วนกลับหลัง

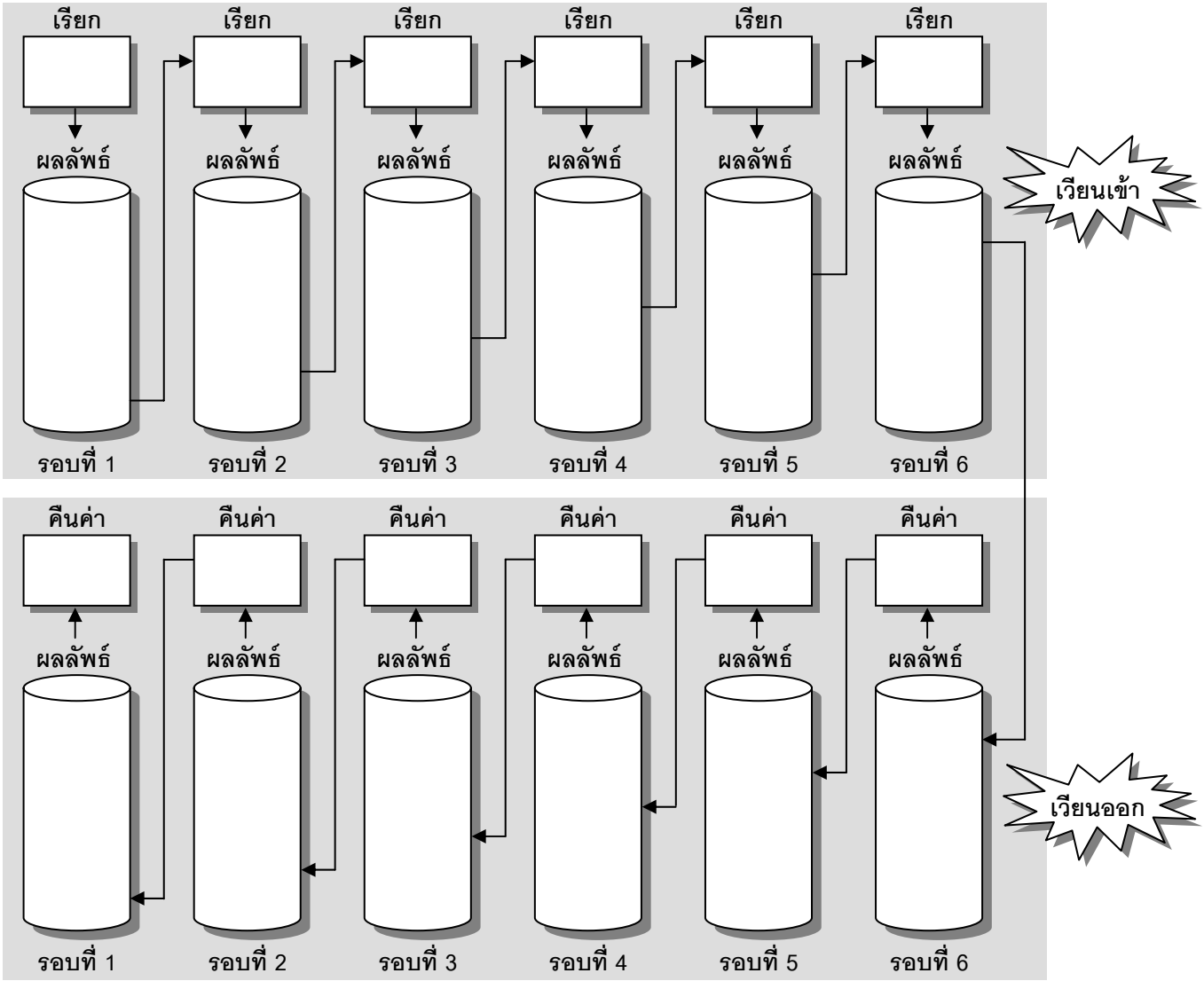
   ใช้หลักการเวียนเกิด (Recursion)

```
1  public static int f(int n) {
2    if(n <= 0) {
3      return 0;
4    } else {
5      return f(n - 1) + n;
6    }
7  }
```

กรณีฐาน

กรณีเวียนเกิด

โจทย์ข้อที่ 1 [ระดับง่าย] จากการทำงานแบบเวียนเกิดของเมท็อด `f(…)` ก่อนหน้านี้ จงแสดงรายละเอียดการทำงานของเมท็อดดังกล่าวเมื่อเรียกใช้ `f(5)` *(10 คะแนน)*



โจทย์ข้อที่ 2 [ระดับง่าย] จงแสดงผลลัพธ์จากการเรียกใช้เมท็อดแบบเวียนเกิดต่อไปนี้ *(6 คะแนน)*

```
1  public static int mul(int x, int y) {
2    if (x >= 1) {
3      return y + mul(x - 1, y);
4    } else {
5      return 0;
6    }
7  }
```

เรียกการเวียนเกิดแบบนี้ว่า Linear Recursion

การเรียกใช้	คำตอบ
`mul(4, 3)`	
`mul(5, 7)`	
`mul(20, 10)`	

โจทย์ข้อที่ 3 [ระดับง่าย] จงแสดงผลลัพธ์จากการเรียกใช้เมท็อดแบบเวียนเกิดต่อไปนี้ *(6 คะแนน)*

```
1  public static int gcd(int m, int n) {
2    if (m < n) return gcd(n, m);
3    if (m % n == 0) {
4      return n;
5    } else {
6      return gcd(n, m % n);
7    }
8  }
```

เรียกการเวียนเกิดแบบนี้ว่า Tail Recursion

การเรียกใช้	คำตอบ
`gcd(28, 16)`	
`gcd(9, 14)`	
`gcd(75, 30)`	

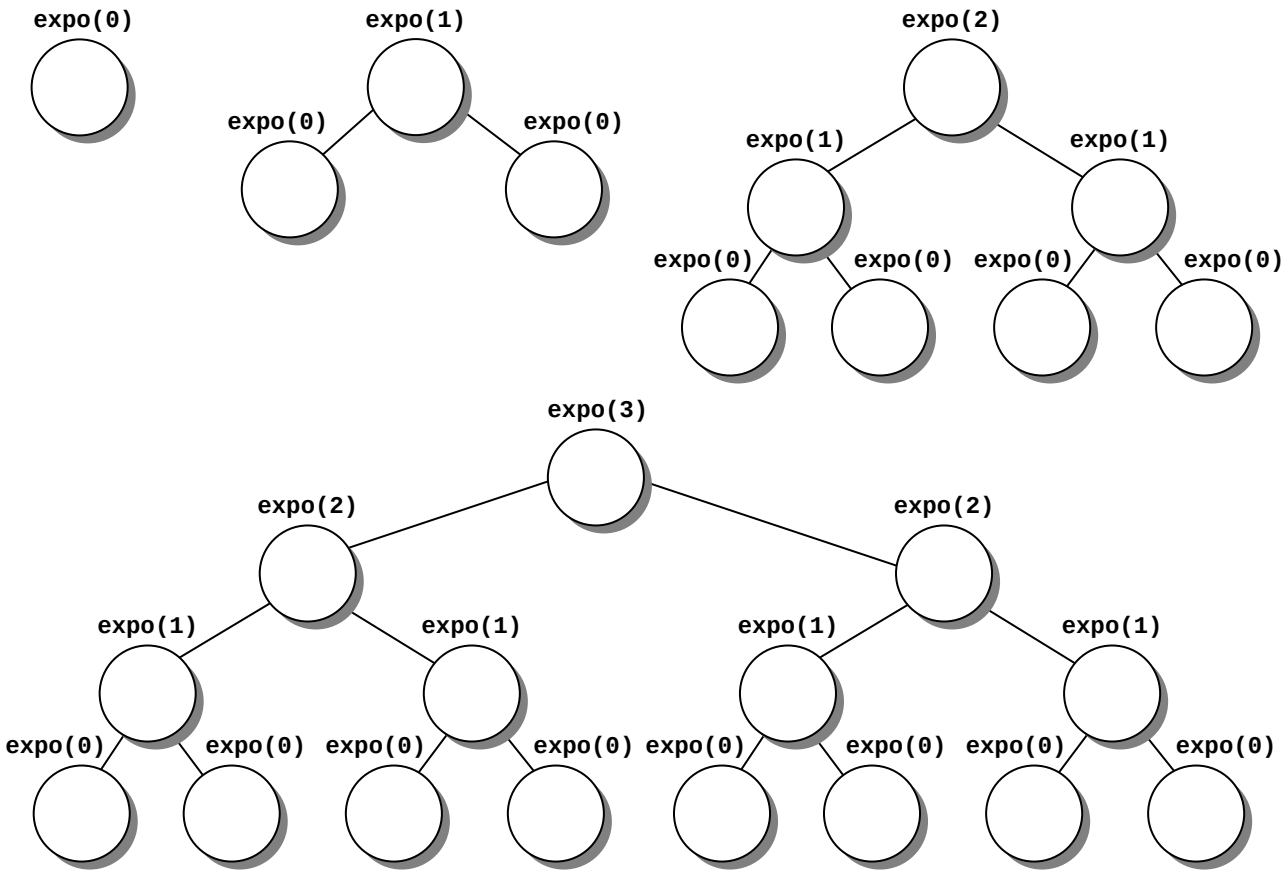
โจทย์ข้อที่ 4 [ระดับง่าย] จงแสดงผลลัพธ์และรายละเอียดจากการเรียกใช้เมท็อด **expo(…)** *(15 คะแนน)*

```
1  public static int expo(int n) {
2    if (n <= 0) {
3      return 1;
4    } else {
5      return expo(n - 1) + expo(n - 1);
6    }
7  }
```

เรียกการเวียนเกิดแบบนี้ว่า Binary Recursion

การเรียกใช้	คำตอบ
expo(4)	
expo(7)	
expo(11)	

จงแสดงรายละเอียดของต้นไม้แบบทวิภาค (Binary Tree) **จากการเรียกใช้ expo(0), expo(1), expo(2) และ expo(3)**



โจทย์ข้อที่ 5 [ระดับง่าย] จงแสดงผลลัพธ์จากการเรียกใช้เมท็อดแบบเวียนเกิดต่อไปนี้ *(6 คะแนน)*

```
1  public static void printX(int x) {
2    if (x >= 1) {
3      printX(1, x);
4    }
5  }
6  public static void printX(int i, int x) {
7    if (i <= x) {
8      System.out.print(i);
9      printX(i + 1, x);
10   } else {
11     System.out.println();
12     printX(x - 1);
13   }
14 }
```

เรียกการเวียนเกิดแบบนี้ว่า Mutual Recursion

การเรียกใช้และคำตอบ	
printX(4)	printX(7)

โจทย์ข้อที่ 6 [ระดับปานกลาง] จงแสดงผลลัพธ์ในตารางต่อไปนี้ให้ถูกต้อง ซึ่งเกิดจากการเรียกใช้เมท็อด akm(m, n) แบบเวียนเกิด โดยกำหนดให้ใช้ค่าพารามิเตอร์ m และ n ตามตาราง *(10 คะแนน)*

```
1  public static int akm(int m, int n) {
2    if (m == 0) {
3      return n + 1;
4    } else if (n == 0) {
5      return akm(m - 1, 1);
6    } else {
7      return akm(m - 1, akm(m, n - 1));
8    }
9  }
```

เรียกการเวียนเกิดแบบนี้ว่า Nested Recursion

เป็น Recursion ที่ไม่สามารถเปลี่ยนเป็น Iteration ได้ เสนอโดย Wilhelm Ackermann

พารามิเตอร์ m	พารามิเตอร์ n				
	0	1	2	3	4
0					
1					
2					
3					

2. การเวียนเกิดแบบง่าย (Basic Recursion)

1. **การเวียนเกิดแบบง่าย** ได้แก่ การเวียนเกิดที่ทำงานเสร็จสิ้นอยู่ภายในเมท็อดเดียว ซึ่งในบทนี้จะเน้นการเวียนเกิดแบบ Linear Recursion, Tail Recursion และ Binary Recursion เป็นหลัก
2. **ลักษณะของเมท็อดที่ปรากฏในการเวียนเกิดแบบง่าย** โดยทั่วไปแล้วจะเป็นเมท็อดที่ใช้หาผลลัพธ์ทางคณิตศาสตร์ เช่น อนุกรม (Series), ฟังก์ชัน (Functions) หรือ อัลกอริทึม (Algorithms) เป็นต้น
3. **การแปลงสมการคณิตศาสตร์ให้เป็นเมท็อดเวียนเกิดแบบง่าย** มีขั้นตอนดังนี้

   กำหนดโจทย์ จงหาผลบวกของตัวเลขจำนวนเต็มตั้งแต่ 0 จนถึง n (โดยที่ $n \geq 0$)

   1) แปลงโจทย์ที่กำหนดให้ ให้เป็นสมการหรือฟังก์ชันทางคณิตศาสตร์ หรือสมการในรูปทั่วไปซึ่งจะได้

   $$f(n) = \sum_{i=0}^{n} i \qquad n = 0,1,2,3,\ldots$$

   2) แยกสมการออกเป็น 2 กรณี (หรืออาจจะมากกว่า 2 กรณีก็ได้) ซึ่งได้แก่

      (1) กรณีที่สมการเท่ากับค่าของพจน์แรกเพียงพจน์เดียวซึ่งจะได้

      $$f(n) = 0 \qquad n = 0 \qquad \text{-------} (1)$$

      (2) กรณีที่สมการอยู่ในรูปทั่วไป แต่จะดึงพจน์สุดท้ายออกมาให้เห็นในสมการซึ่งจะได้

      $$f(n) = \sum_{i=0}^{n-1} i + n \qquad n = 1,2,3,\ldots$$

      $$f(n) = f(n-1) + n \qquad n = 1,2,3,\ldots \quad \text{-------} (2)$$

3) แปลงสมการทั้ง 2 กรณีที่ได้จากขั้นตอนที่ 2 ให้เป็นเมท็อดในภาษาจาวาโดยใช้คำสั่ง `if-else` เพื่อระบุเงื่อนไขของแต่ละกรณี ซึ่งจะให้กรณีที่ 1 เป็นกรณีฐาน และกรณีที่ 2 เป็นกรณีเวียนเกิด

```
1  public static int f(int n) {
2    if(n > 0) {
3      return f(n - 1) + n;
4    } else {
5      return 0;
6    }
7  }
```

เมื่อ n น้อยกว่าหรือเท่ากับ 0 จะหยุดการเวียนเกิด

เมื่อ n มากกว่า 0 จะทำการเวียนเกิด (ไปเรื่อยๆ)

```
1  public static int f(int n) {
2    if(n <= 0) {
3      return 0;
4    } else {
5      return f(n - 1) + n;
6    }
7  }
```

ในการเวียนเกิดแต่ละรอบ จะมีการลดค่า n ลง 1

<u>โจทย์ข้อที่ 7</u> <u>[ระดับง่าย]</u> จงพิจารณาสมการต่อไปนี้เพื่อเขียนเป็นเมท็อด `f(…)` ที่ทำงานแบบเวียนเกิด โดยให้รับค่า n เข้ามาทางพารามิเตอร์ และไม่อนุญาตให้ใช้เมท็อดจากคลาส `Math` *(10 คะแนน)*

$$f(n) = \sum_{i=1}^{n} \frac{1}{i^3} \qquad n \geq 1$$

กรณีฐาน

กรณีเวียนเกิด

<u>โจทย์ข้อที่ 8</u> <u>[ระดับง่าย]</u> จงพิจารณาสมการต่อไปนี้เพื่อเขียนเป็นเมท็อด `f(…)` ที่ทำงานแบบเวียนเกิด โดยให้รับค่า n เข้ามาทางพารามิเตอร์ และอนุญาตให้ใช้เมท็อดจากคลาส `Math` *(10 คะแนน)*

$$f(n) = \begin{cases} \sum_{i=1}^{n} (i^i + 2i) & n \geq 1 \\ 0 & n = 0 \end{cases}$$

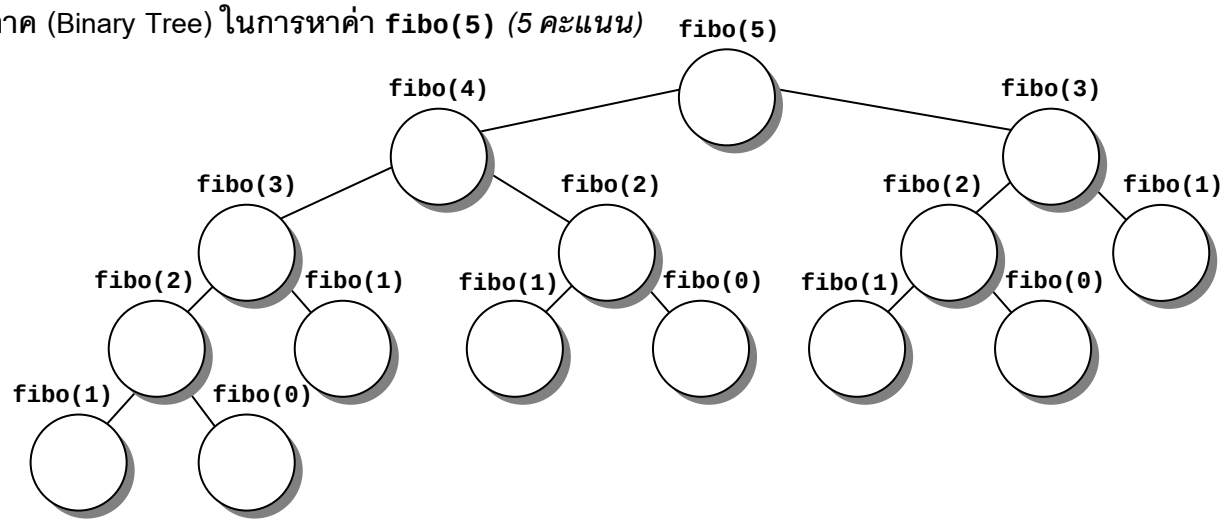
โจทย์ข้อที่ 9 [ระดับง่าย] จงพิจารณาสมการต่อไปนี้เพื่อเขียนเป็นเมท็อด **g(…)** ที่ทำงานแบบเวียนเกิด โดยให้รับค่า **n** เข้ามาทางพารามิเตอร์ และไม่อนุญาตให้ใช้เมท็อดจากคลาส **Math** *(10 คะแนน)*

$$g(n) = \begin{cases} \prod_{i=2}^{n}\left(1 + \frac{1}{i}\right) & n \geq 2 \\ 1 & n = 1 \\ 0 & n = 0 \end{cases}$$

โจทย์ข้อที่ 10 [ระดับง่าย] จงเขียนเมท็อด **fac(…)**แบบเวียนเกิดเพื่อคำนวณค่าแฟกทอเรียล (Factorial) ของตัวเลขจำนวนเต็ม **n** เช่น **fac(5)** คือค่าของ 5! ซึ่งเท่ากับ 120 เป็นต้น *(10 คะแนน)*

โจทย์ข้อที่ 11 [ระดับง่าย] จงเขียนเมท็อด **fibo(…)** แบบเวียนเกิดเพื่อคำนวณหาคำตอบของสมการฟิ-โบแนคซี (Fibonacci) ซึ่งมีรูปแบบสมการเป็นดังนี้ **fibo(n) = fibo(n - 1) + fibo(n - 2)** เมื่อ **n >= 2** โดยกำหนดให้ **fibo(0) = 0** และ **fibo(1) = 1** *(10 คะแนน)*

โจทย์ข้อที่ 12 [ระดับง่าย] จากเมท็อด **fibo(…)** ในโจทย์ด้านบนจงแสดงรายละเอียดของต้นไม้แบบทวิภาค (Binary Tree) ในการหาค่า **fibo(5)** *(5 คะแนน)*



โจทย์ข้อที่ 13 [ระดับปานกลาง] จงเขียนเมท็อด **pow(…)** แบบเวียนเกิดเพื่อคำนวณหาค่ายกกำลังของ a^b โดยกำหนดให้ **a** เป็นจำนวนจริง และ **b** เป็นจำนวนเต็มใดๆ (เต็มบวก เต็มลบ เต็มศูนย์) *(10 คะแนน)*

*ไม่*อนุญาตให้ใช้เมท็อดจากคลาส **Math**

โจทย์ข้อที่ 14 [ระดับยาก] จงเขียนเมท็อด **f(…)**แบบเวียนเกิดจากสมการต่อไปนี้ โดยให้รับค่า **n** เข้ามาทางพารามิเตอร์ และ*ไม่*อนุญาตให้ใช้เมท็อดจากคลาส **Math** *(10 คะแนน)*

$$f(n) = 1 + \cfrac{n}{1 + \cfrac{n-1}{1 + \cfrac{n-2}{1 + \cfrac{n-3}{1 + \ddots}}}}$$

โจทย์ข้อที่ 15 [ระดับยาก] จงพิจารณาสมการที่กำหนดให้ต่อไปนี้ เพื่อเขียนเป็นเมท็อดที่มีการทำงานแบบเวียนเกิด โดยให้รับค่าตัวแปรที่ใช้ในการคำนวณเข้ามาทางพารามิเตอร์ *(20 คะแนน)*

1) $g(n, m) = 1^m - 2^m + 3^m - \ldots + (n^m)$ $\quad n = 1,2,3,\ldots$ *(10 คะแนน)*

อนุญาตให้ใช้เมท็อดจากคลาส **Math**

2) $h(n) = 1 + \frac{1}{2} - \frac{1}{3} + \frac{1}{4} - \ldots + \frac{1}{n}$ $\quad n = 1,2,3,\ldots$ *(10 คะแนน)*

*ไม่*อนุญาตให้ใช้เมท็อดจากคลาส **Math**

3. การเวียนเกิดแบบซับซ้อน (Advanced Recursion)

1. **การเวียนเกิดแบบซับซ้อน** ได้แก่

   การเวียนเกิดที่ใช้เมท็อดตั้งแต่ 2 เมท็อดขึ้นไปในการทำงาน (เมท็อดมีการ Overload กัน) ซึ่งในบทนี้จะเน้นการเวียนเกิดแบบ Mutual Recursion เป็นหลัก

2. **ลักษณะของเมท็อดที่ปรากฏในการเวียนเกิดแบบซับซ้อน** โดยทั่วไปแล้วจะเป็นเมท็อดที่ใช้ประมวลผลเกี่ยวกับสตริง (String) และอาเรย์ (Arrays) ดังตัวอย่างต่อไปนี้

```
1   public static int max(int a[]) {
2     return max(a, 0, a[0]);
3   }
4
5   public static int max(int a[], int i, int m) {
6     if (i < a.length) {
7       if (a[i] > m) m = a[i];
8       return max(a, i + 1, m);
9     } else {
10      return m;
11    }
12  }
```

เมท็อด max(…) มีการ Overload กัน

เมท็อด max(…) ใช้หาค่าสมาชิกที่มากที่สุดในอาเรย์ โดยเมท็อด max(…) ด้านบนจะรับอาเรย์เข้ามาแล้วส่งต่อไปให้กับเมท็อด max(…) ด้านล่างเพื่อเวียนเกิด (Indirect Recursion)

3. **หลักการเขียนเมท็อดเวียนเกิดแบบซับซ้อน**

การเขียนเมท็อดเวียนเกิดแบบซับซ้อนจะต้องเขียนเมท็อดหลายเมท็อดที่ Overload กันเสมอ เพราะเนื่องจากเมท็อดเวียนเกิดตัวแรกที่ประมวลผลอาจต้องใช้ตัวแปรหรือค่าบางค่าเพิ่มเติมภายในเมท็อด และค่าเหล่านั้นจะต้องนำกลับไปใช้ต่อเนื่องในรอบต่อๆ ไปอีกด้วย เช่น ค่าตำแหน่งปัจจุบันในอาเรย์ เป็นต้น ดังนั้นจึงต้องส่งค่าทุกค่าผ่านไปยังพารามิเตอร์ เราจึงจำเป็นต้องสร้างเมท็อดตัวที่สองขึ้นมาที่ Overload กับตัวแรก เพื่อให้สามารถส่งค่าทุกค่าที่ต้องใช้งานผ่านไปยังพารามิเตอร์ได้ครบถ้วน แล้วให้ทำการเวียนเกิดที่เมท็อดตัวที่สองนี้

โจทย์ข้อที่ 16 [ระดับปานกลาง] จงเขียนเมท็อด search(…) แบบเวียนเกิดเพื่อค้นหา*ตำแหน่ง*สมาชิกในอาเรย์ที่มีค่าเท่ากับจำนวนเต็มที่รับเข้ามา ถ้าค้นเจอให้คืนค่าตำแหน่งสมาชิกที่เจอ แต่ถ้าไม่เช่นนั้นให้คืนค่า -1 ***(10 คะแนน)***

โจทย์ข้อที่ 17 [ระดับปานกลาง] จงเขียนเมท็อด **overHundred(…)** แบบเวียนเกิดเพื่อนับตัวเลขจำนวนเต็มที่มีค่ามากกว่า 100 ในอาเรย์ **d** ที่รับเข้ามาว่ามีกี่จำนวน *(10 คะแนน)*

```
public class OverHundredNumber {
  public static void main(String[] args) {
    int d[] = { 99, 101, 13, 78, 200, 534, 47, 1234, 736 };
    System.out.println(overHundred(d));
  }
```

```
} //End of class
```

โจทย์ข้อที่ 18 [ระดับปานกลาง] จงเขียนเมท็อด **revStr(…)** แบบเวียนเกิดเพื่อกลับตำแหน่งของอักขระทุกตัวในสตริงจากหลังมาหน้า เช่น สตริง **"Computer"** จะได้เป็น **"retupmoC"** เป็นต้น *(10 คะแนน)*

โจทย์ข้อที่ 19 [ระดับยาก] จงเขียนเมท็อด **`formulaAtoB(…)`**แบบเวียนเกิดเพื่อพิมพ์สูตรคูณตั้งแม่ **a** จนถึงแม่ **b** ซึ่งรับเข้ามาทางพารามิเตอร์ เช่น **`formulaAtoB(6, 15)`** จะได้สูตรคูณตั้งแต่แม่ 6 ถึงแม่ 15 เป็นต้น *(10 คะแนน)*

โจทย์ข้อที่ 20 [ระดับยาก] จงเขียนเมท็อด **`addArray(…)`** แบบเวียนเกิดเพื่อใช้สำหรับหาผลบวกระหว่างอาเรย์หนึ่งมิติ 2 ชุดใด ๆ ที่มีจำนวนสมาชิกเท่ากัน แล้วคืนค่ากลับ *(10 คะแนน)*

โจทย์ข้อที่ 21 [ระดับยาก] จงเขียนเมท็อด **`isMatrixEquals(…)`** แบบเวียนเกิดเพื่อรับเมตริกซ์ชนิดจำนวนเต็มเข้ามา 2 ตัวเพื่อตรวจสอบว่าเมตริกซ์ทั้งสองเท่ากันหรือไม่ โดยขนาดจะต้องเท่ากันทั้งแถวและหลัก และสมาชิกทุกในตำแหน่งที่ตรงกันต้องมีค่าเท่ากัน *(10 คะแนน)*