

การเขียนโปรแกรมย่อย

หัวข้อ

- องค์ประกอบของเมทอด
- การเรียกเมทอดและการจัดเก็บตัวแปร
- การส่งและรับค่าของเมทอด
- เมทอดกับอาเรย์
- ตัวอย่าง

การเขียนโปรแกรม

- แบ่งโปรแกรมเป็นส่วนย่อยๆ
- เรียกส่วนย่อยเหล่านี้ว่า subroutine, subprogram, function หรือ method
- แต่ละ method มีหน้าที่ในตัวเองอย่างเด่นชัด

```
class Game {  
    main(...) {  
        ...  
    }  
  
    begin(...) {  
    }  
  
    option(...) {  
    }  
  
    ending(...) {  
    }  
  
    demo(...) {  
    }  
}
```

ตัวอย่าง : ดัชนีมวลกาย

```
import java.util.Scanner;  
public class BodyMassIndex {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(System.in);  
  
        double weight = readDouble(kb, "น้ำหนัก (kg.) = ");  
  
        double height = readDouble(kb, "ความสูง (cm.) = ");  
        double hm = height / 100.0;  
        double bmi = weight / (hm * hm);  
        System.out.println("ดัชนีมวลกาย = " + bmi);  
    }  
  
    public static double readDouble(Scanner kb,  
                                    String msg){  
        System.out.print(msg);  
        double v = kb.nextDouble();  
        return v;  
    }  
}
```

องค์ประกอบของเมทอด

ประเภทของผลลัพธ์ ชื่อเมทอด รายการพารามิเตอร์

```
public static int readInt(Scanner kb, String msg){  
    System.out.print(msg);  
    int v = kb.nextInt();  
    return v;  
}
```

คืนผลจากเมทอด

static – เมทอดประจำคลาส
public – เมทอดสาธารณะ
private – เมทอดที่ใช้ได้เฉพาะในคลาสที่เขียน

ส่วนหัว : ชื่อเมทอด

- ใช้กฎการตั้งชื่อเหมือนกับตัวแปร
- มักตั้งชื่อเมทอดขึ้นต้นด้วยตัวอักษรเล็ก
- มักตั้งชื่อเมทอดให้เป็นกริยา

fillEllipse	drawLine	fade	readDouble
print	println	hasNext	nextInt
random	max	min	cos

ส่วนหัว : ประเภทของผลลัพธ์

- **int, double, String, ...**
- ถ้าไม่คืนผล, ใช้คำว่า **void**

```
public static int max(int a, int b)
```

```
public static double sqrt(double x)
```

```
public static int getCurrentMonth()
```

```
public static void printError(String msg)
```

ส่วนหัว : รายการของพารามิเตอร์

- parameter คือตัวแปรสำหรับรับข้อมูลจากผู้เรียก

```
public static void printError(String msg)
```

- parameter แต่ละตัว เขียนเหมือนการประกาศตัวแปร (คั่นด้วย , ไม่ต้องปิดท้ายด้วย ;)

```
public static int max(int a, int b)
```

- รายการของ parameters อยู่ภายในวงเล็บ (ถ้าไม่รับ parameter ใดๆ ก็ไม่ต้องใส่อะไรในวงเล็บ)

```
public static int getCurrentMonth()
```

ส่วนหัว : ตัวอย่างที่ไม่ผิด

```
public static int a( int a )
```

```
public static void Nothing( )
```

```
public static void $$$ ( double $1, int $2 )
```

ชื่อเมทอดซ้ำกันได้ถ้ารายการของประเภทพารามิเตอร์ไม่เหมือน

```
public static int f(int a, int b) {  
    ...  
}  
public static int f(int a, int b, double c) {  
    ...  
}
```

ส่วนหัว : ตัวอย่างผิด ๆ

```
public static getFirstCharacter(String txt)
```

```
public static int min( int a, b )
```

```
public static void clear( void )
```

```
public static int f(int a, int b) {  
    ...  
}  
public static double f(int a, int b) {  
    ...  
}  
public static double f(int x, int y) {  
    ...  
}
```

ส่วนตัว : การคืนการทำงานสู่ผู้เรียก

- เมทอดแบบ **void** คืนการทำงาน เมื่อทำคำสั่ง **return** หรือ ทำจนถึง } ที่ปิดตัวเมทอด

```
public static void main(String[] args) {  
    Scanner kb = new Scanner(System.in);  
    double a = readDouble("a = ");  
    if (a < 0) {  
        System.out.println("ค่า a ต้องมีมากกว่า 0");  
        return;  
    }  
    double x = 1.0;  
    while ((x*x - a) >= 1e-5) {  
        x = (x + (a / x)) / 2.0;  
    }  
    System.out.println("รากที่สอง = " + x);  
}
```

ส่วนตัว : การคืนการทำงานสู่ผู้เรียก

- เมทอดที่มีการคืนผล คืนการทำงาน เมื่อ
 - ทำคำสั่ง **return** ตามด้วยนิพจน์ที่แทนค่าที่คืน
 - คืนผลได้เพียงค่าเดียวเท่านั้น
 - ค่าที่คืนต้องเป็น
 - ประเภทเดียวกับที่ประกาศไว้ที่หัว
 - ประเภทที่เปลี่ยนเป็นแบบที่ประกาศไว้อย่างอัตโนมัติได้

```
public static double clip(double x) {  
    if (x < 0) return 0;  
    return x;  
}
```

ส่วนตัว : พารามิเตอร์และตัวแปร

- ตัวแปรเฉพาะที่ (local variables)
 - ประกาศในเมทอดใด ใช้ได้เฉพาะในเมทอดนั้น
 - ระบบสร้างตัวแปรเฉพาะที่ เมื่อเมทอดถูกเรียก
 - ระบบลบตัวแปรเฉพาะที่ เมื่อเมทอดทำงานเสร็จ
 - ซ้ำกันไม่ได้ ถ้าอยู่คนละเมทอด

```
public static int b(int x) {  
    int a = x + 8;  
    return a;  
}  
public static int a(int x) {  
    int a = 2 * x;  
    return a;  
}
```

ตัวอย่างผิด ๆ

```
public static int abs(int a) {  
    if ( a < 0 ) a = -a;  
}
```

```
public static int getBlackColor() {  
    return 0.0;  
}
```

```
public static double length(double dx, double dy){  
    double dx = Math.abs(dx);  
    return Math.sqrt( dx*dx + dy*dy );  
}
```

GCD : หาค่าร่วมมาก

```
public static void main(String[] args) {  
    Scanner kb = new Scanner(System.in);  
    System.out.print("n = ");  
    int n = kb.nextInt();  
    System.out.print("m = ");  
    int m = kb.nextInt();  
    System.out.println("gcd = " + gcd(n,m) );  
}  
public static int gcd(int n, int m) {  
    while (m > 0) {  
        int t = n % m;  
        n = m;  
        m = t;  
    }  
    return n;  
}
```

โปรแกรม : วันใดของสัปดาห์

```
public static String day(int d, int m, int y) {  
    if (m < 3) {  
        m = m + 12; y = y - 1;  
    }  
    int c = y / 100, k = y % 100;  
    int w = (d + 26*(m+1)/10 + k + k/4 + c/4 + 5*c) % 7;  
    String dow = "";  
    if (w == 0) dow = "เสาร์";  
    if (w == 1) dow = "อาทิตย์";  
    if (w == 2) dow = "จันทร์";  
    if (w == 3) dow = "อังคาร";  
    if (w == 4) dow = "พุธ";  
    if (w == 5) dow = "พฤหัสบดี";  
    if (w == 6) dow = "ศุกร์";  
    return dow;  
}
```

การเรียกเมทอด

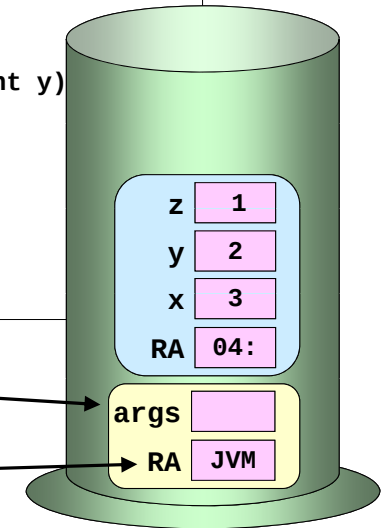
```
01:public class StackFrame {
02:  public static void main(String[] args) {
03:    a(3, 2);
04:    b(5);
05:  }
06:  public static void a(int x, int y) {
07:    int z = x/y;
08:    b(z);
09:  }
10:  public static void b(int x) {
11:    ++x;
12:  }
13:}
```

ตัวแปรและพารามิเตอร์ในเมทอดหนึ่ง "เกิด" เมื่อเมทอดนั้นถูกเรียกให้ทำงาน
ตัวแปรเหล่านี้ "หาย" ไป เมื่อคืนการทำงานกลับสู่ผู้เรียก

การจัดเก็บตัวแปร

```
01:public class StackFrame {
02:  public static void main(String[] args) {
03:    a(3, 2);
04:    b(5);
05:  }
06:  public static void a(int x, int y) {
07:    int z = x/y;
08:    b(z);
09:  }
10:  public static void b(int x) {
11:    ++x;
12:  }
13:}
```

parameters &
local variables
return address



การส่งค่าและการรับผล

```
int a = -9;
int b = clip(a + 5);
```



```
public static int clip(int a) {
  if (a < 0 ) return 0;
  if (a > 255) return 255;
  return a;
}
```



การส่งค่าและการรับผล

```
int a = 300;
int b = clip(a);
```



```
public static int clip(int a) {
  if (a < 0 ) return 0;
  if (a > 255) return 255;
  return a;
}
```



การส่งค่าและการรับผล

```
int a = 300;
int b = clip(a);
```



```
public static int clip(int a) {
    a = Math.max(a, 0);
    a = Math.min(a, 255);
    return a;
}
```



การส่งค่าและการรับผล

```
int q = distance(10, 10, 20, 30);
int q = (int) distance(10, 10, 20, 30);
double x = 20.5;
double p = distance(10, 10, 20, x);
double p = distance(10, 10, 20, (int) x);
```

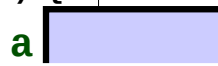
```
public static double distance(int x1, int y1, int x2, int y2){
    double dx = (x1 - x2);
    double dy = (y1 - y2);
    double d = Math.sqrt(dx*dx + dy*dy);
    return d;
}
```

ไม่มีประโยชน์

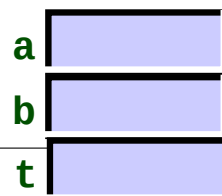
```
int a = 99;
int b = 20;
clear( a );
swap( a, b );
```



```
public static void clear(int a) {
    a = 0;
}
```

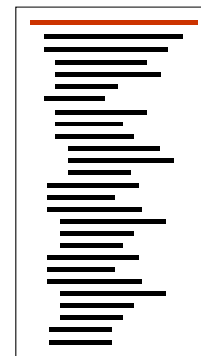


```
public static void swap(int a, int b) {
    int t = a;
    a = b;
    b = t;
}
```



เมื่อใดควรเขียนเมทอดใหม่

เมื่อเมทอดที่เขียนอยู่ยาวเกินไป



เมื่อใดควรเขียนเมท็อดใหม่

เมื่อมีกลุ่มคำสั่งที่เขียนซ้ำ ๆ



ข้อแนะนำในการเขียนเมท็อด

- ควรตั้งชื่อที่สื่อความหมาย
- ควรมีภาระที่ต้องทำหนึ่งอย่างตามชื่อ
- ควรสั้นกะทัดรัด อ่านเข้าใจง่าย
- ควรมีพารามิเตอร์จำนวนไม่มาก
- ควรเขียนหมายเหตุกำกับ

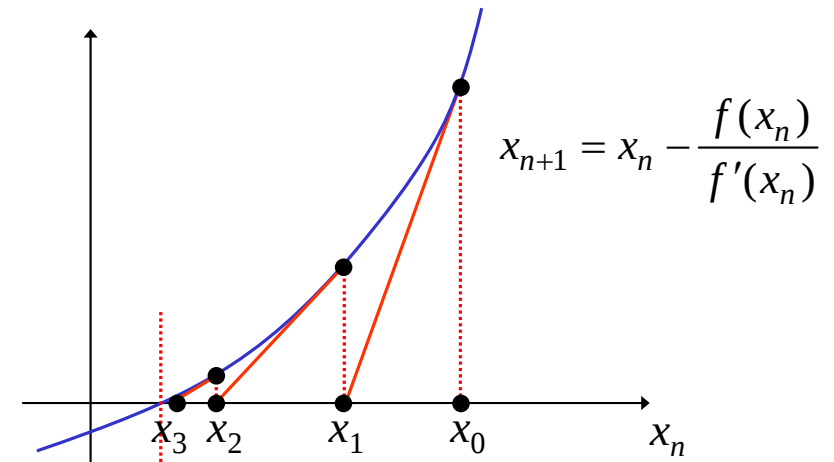
Newton's Method

- ต้องการหารากของสมการ $f(x) = 2x^2 - 11x + 5$
- หาอนุพันธ์เตรียมไว้ก่อน $f'(x) = 4x - 11$
- เริ่มด้วยการให้ค่าเริ่มต้นกับ x_0
- คำนวณ $x_1, x_2, x_3, \dots, x_n, x_{n+1}, \dots$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- วนคำนวณจนค่า x_n และ x_{n+1} เปลี่ยนแปลงน้อยมาก ๆ

Newton's Method



ตัวอย่าง : โปรแกรมการหารากโดยวิธีของนิวตัน

```
import java.util.Scanner;
public class Newton {
    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("initial x = ");
        double x0 = kb.nextDouble();
        double err = 1.0;
        while (err > 1e-10) {
            System.out.println(x0);
            double x1 = x0 - f(x0) / df(x0);
            err = Math.abs(x1 - x0);
            x0 = x1;
        };
        System.out.println(x0);
    }
    private static double f(double x) {
        return 2*x*x - 11*x + 5;
    }
    private static double df(double x) {
        return 4*x - 11;
    }
}
```

```
JLab>java Newton
initial x = 2
2.0
-1.0
0.19999999
0.48235294
0.49993133
0.49999999
0.5
0.5
```

```
JLab>java Newton
initial x = 10
10.0
6.724137931
5.373999551
5.026653141
5.000156016
5.000000005
5.0
5.0
```

หมายเหตุ โปรแกรมนี้หา
ได้เฉพาะรากจำนวนจริง

$$f(x) = 2x^2 - 11x + 5 = (2x - 1)(x - 5)$$

211 29

เมท็อดที่รับอาร์เรย์

```
public static double sum(double[] d) {
    ...
}

public static int search(int[] d, int x) {
    ...
}

public static void sort(int[] d) {
    ...
}
```

2110101 วิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์ฯ (10/06/52)

30

การส่งอาร์เรย์ไปยังเมท็อด

- ต้องการส่งอาร์เรย์ใดให้เมท็อด
- ส่งเฉพาะชื่อ ไม่ต้องมี []

```
int[] a = { 1, 3, 4, 0, 2, 5 };
int k = search(a, 0);    // ส่ง a ให้ search
sort(a);                // ส่ง a ให้ sort
int m = max(a[0], a[1]); // ไม่ได้ส่งอาร์เรย์
```

```
public static int search(int[] d, int x) {
    ...
}
public static void sort(int[] d) {
    ...
}
public static int max(int x, int y) {
    if (x > y) return x;
    return y;
}
```

2110101 วิชา

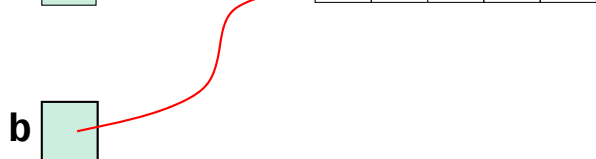
31

ตัวแปรหลายตัวอ้างอิงอาร์เรย์เดียวกันได้

```
int[] a = new int[5];
```



b



```
int[] b = a;
```

a กับ b มีค่าเท่ากัน หมายความว่า อ้างอิงอาร์เรย์เดียวกัน
ดังนั้นผลการเปรียบเทียบ `a == b` ได้ค่า true และ
`a[i]` กับ `b[i]` อ้างอิงที่เก็บข้อมูลที่เดียวกัน

2110101 วิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์ฯ (10/06/52)

32

ส่งอาร์เรย์ = ส่งตำแหน่งของอาร์เรย์

- การส่งอาร์เรย์ไม่ได้ส่งข้อมูลของทั้งอาร์เรย์ไป เป็นเพียงการส่งค่าของตัวแปรอาร์เรย์
- ตัวแปรอาร์เรย์เก็บตำแหน่งอาร์เรย์ (อ้างอิงอาร์เรย์)
- ผู้เรียกเมทอด กับภายในเมทอด ใช้อาร์เรย์เดียวกัน

```
public static void main(String[] args) {
    double[] data = {5, -2, 7, 4, 18};
    ...
    double a = sort( data );
}
```

data

```
public static void sort(double[] x) {
    ...
}
```

x

5 -2 7 4 18

ตัวอย่างเมทอดที่รับอาร์เรย์

```
public static double sum(double[] d) {
    double sum = 0;
    for (int i = 0; i < d.length; i++) {
        sum = sum + d[i];
    }
    return sum;
}
```

```
public static int search(double[] d, int x) {
    for (int i = 0; i < d.length; i++) {
        if (d[i] == x) return i;
    }
    return -1; // หาไม่พบ คืน -1
}
```

ตัวอย่างเมทอดที่รับอาร์เรย์

```
public static double max(double[] d) {
    double max = d[0];
    for (int i = 1; i < d.length; i++) {
        if (d[i] > max) max = d[i];
    }
    return max;
}
```

```
public static int maxIndex(double[] d) {
    int maxIndex = 0;
    for (int i = 1; i < d.length; i++) {
        if (d[i] > d[maxIndex]) maxIndex = i;
    }
    return maxIndex;
}
```

การเรียงลำดับข้อมูลแบบเลือก

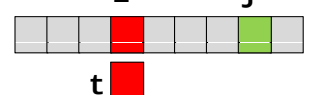
0	1	2	3	4	5	6
23	2	3	14	5	99	9
23	2	3	14	5	9	99
9	2	3	14	5	23	99
9	2	3	5	14	23	99
5	2	3	9	14	23	99
3	2	5	9	14	23	99
2	3	5	9	14	23	99

เมท็อดการเรียงลำดับข้อมูลแบบเลือก

```
public static int maxIndex(double[] d, int k) {
    int maxIndex = 0;
    for (int i = 1; i <= k; i++) {
        if (d[i] > d[maxIndex]) maxIndex = i;
    }
    return maxIndex;
}

public static void swap(double[] d, int i, int j) {
    double t = d[i];
    d[i] = d[j];
    d[j] = t;
}

public static void sort(double[] d) {
    for (int lastIndex = d.length-1; lastIndex >= 1; lastIndex--) {
        int maxIndex = maxIndex(d, lastIndex);
        swap(d, maxIndex, lastIndex);
    }
}
```

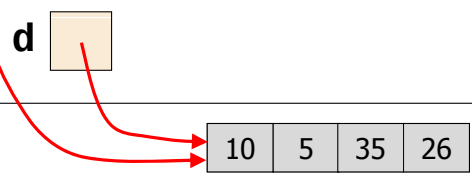


เมท็อดที่คืนอาร์เรย์

สิ่งที่คืนคือตำแหน่งอ้างอิงอาร์เรย์

```
public static void main(String[] args) {
    int[] x = randomIntArray(4);
    ... X
}

public static int[] randomIntArray(int n) {
    int[] d = new int[n];
    for (int i = 0; i < d.length; i++) {
        d[i] = (int)(100 * Math.random());
    }
    return d;
}
```



ถ้าเมท็อดต้องคืนข้อมูลมากกว่าหนึ่งตัว

- ถ้าคืนผลที่เป็นประเภทเดียวกัน
 - ให้เมท็อดสร้าง + คืนอาร์เรย์ที่เก็บผลตามช่องต่าง ๆ

```
public static int[] minmax(int a, int b) {
    if (a < b)
        return new int[] { a, b };
    else
        return new int[] { b, a };
}
```

- ให้ผู้เรียกเตรียม + ส่งอาร์เรย์ที่เก็บผลมาให้เมท็อดเติมผล

```
public static void minmax(int a, int b, int[] out) {
    if (a < b) {
        out[0] = a; out[1] = b;
    } else {
        out[0] = b; out[1] = a;
    }
}
```

- ถ้าคืนผลที่มีหลายประเภท : ใช้อ็อบเจกต์ (บทหลัง)

ตัวอย่าง : รากของ $ax^2+bx+c=0$

```
public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.print("a = ");
    double a = kb.nextDouble();
    System.out.print("b = ");
    double b = kb.nextDouble();
    System.out.print("c = ");
    double c = kb.nextDouble();
    double[] r = quadraticRoots(a, b, c);
    System.out.println(r[0] + ", " + r[1]);
}

public static double[] quadraticRoots(
    double a, double b, double c) {
    double t = Math.sqrt(b * b - 4 * a * c);
    double x1 = (-b + t) / (2 * a);
    double x2 = (-b - t) / (2 * a);
    return new double[] {x1, x2};
}
```

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

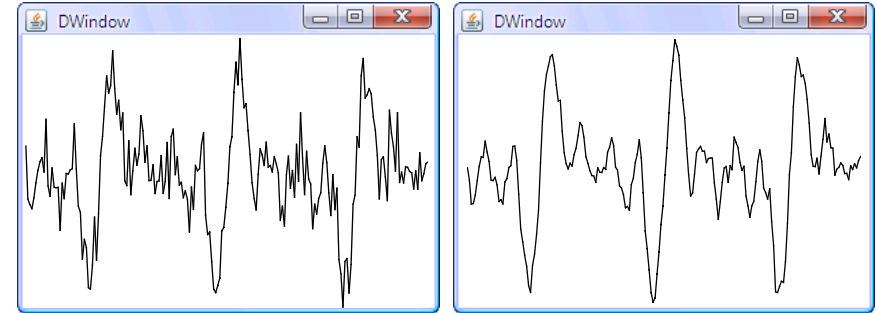
ตัวอย่าง : รากของ $ax^2+bx+c=0$

```
public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.print("a = ");
    double a = kb.nextDouble();
    System.out.print("b = ");
    double b = kb.nextDouble();
    System.out.print("c = ");
    double c = kb.nextDouble();
    double[] r = new double[2];
    quadraticRoots(a, b, c, r);
    System.out.println(r[0] + " " + r[1]);
}

public static void quadraticRoots(
    double a, double b, double c, double[] r) {
    double t = Math.sqrt(b * b - 4 * a * c);
    r[0] = (-b + t) / (2 * a);
    r[1] = (-b - t) / (2 * a);
}
```

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

ค่าเฉลี่ยเคลื่อนที่ (moving average)



in	10	12	13	12	15	10
out	11	11.67	12.33	13.33	12.33	12.5

ตัวอย่าง : ค่าเฉลี่ยเคลื่อนที่

```
public static double[] movingAverage(double[] in) {
    int n = in.length;
    double[] out = new double[n];
    for (int i = 1; i < n - 1; i++) {
        out[i] = (in[i - 1] + in[i] + in[i + 1]) / 3.0;
    }
    out[0] = (in[0] + in[1]) / 2.0;
    out[n - 1] = (in[n - 2] + in[n - 1]) / 2.0;
    return out;
}
```