2110101 : การทำโปรแกรมคอมพิวเตอร์

ประเภทของข้อมูล ตัวแปร และตัวปฏิบัติการพื้นฐาน

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

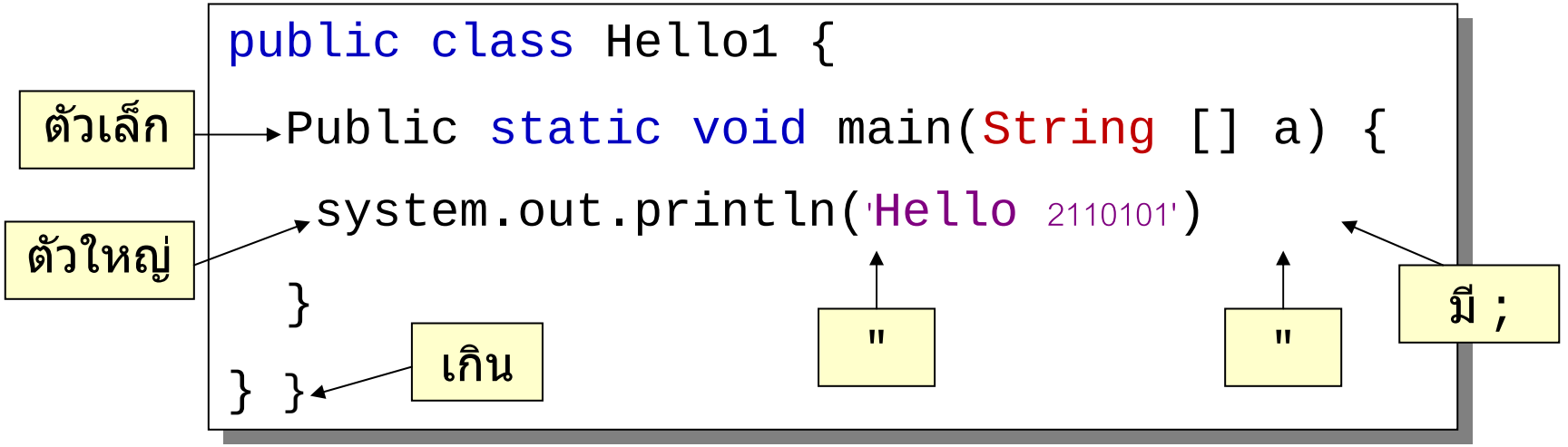
- สิ่งที่ต้องรู้ก่อนทำโปรแกรม
- ประเภทข้อมูล
  - จำนวนเต็ม
  - จำนวนจริง
- ตัวแปร
  - การประกาศ
  - กฎการตั้งชื่อ
- ตัวปฏิบัติการ + – * / %
- นิพจน์

การศึกษาภาษาการทำโปรแกรม

- ไวยากรณ์ของภาษา
  - เขียนอย่างไร ให้ถูกกฎ ตัวแปลภาษาเข้าใจ
- คำสั่ง
  - มีอะไรให้ใช้ ให้นำมาประกอบกัน (ตามไวยากรณ์) เพื่อแก้ไขปัญหาที่ต้องการ
- ประเภทข้อมูล
  - เลือกใช้ประเภทข้อมูลให้เหมาะสมกับงาน เพื่อประสิทธิภาพ และความถูกต้องในการทำงาน

ไวยากรณ์ของภาษาการทำโปรแกรม

- ไม่ซับซ้อน
- ไม่กำกวม
- แต่ผิดนิดผิดหน่อยไม่ได้ (ต้องเขียนถูก 100%)



compiler จะแจ้งตำแหน่งและสาเหตุของความผิดพลาด

คำสั่งและคลังคำสั่งของจาวา

- keywords : มีทั้งสิ้น 47 ตัว (เช่น do, while, if, else, switch, ...)
- operators : มีวิธีปฏิบัติการทางคณิตศาสตร์และตรรกศาสตร์ทั้งสิ้น 40 วิธี (เช่น +, –, *, /, %, +=, ...)

60%

- library : มีคลังคำสั่งขนาดใหญ่มโหฬารให้เรียกใช้ได้เป็นหมื่น (เช่น sqrt, cos, sort, clone, getTimeZone, getLocale, getWordInstance, ...)

1%

ประเภทข้อมูล

- ประเภทพื้นฐาน
  - มี 8 ประเภท
    - จำนวนเต็ม : **`byte`**, **`short`**, **`int`**, **`long`**
    - จำนวนจริง : **`float`**, **`double`**
    - อักขระ : **`char`**
    - ตรรกะ : **`boolean`**
  - ออกแบบเพิ่มเติมไม่ได้
- ประเภท class
  - มีแบบมาตรฐานเกือบสามพันแบบ
    - เช่น **`String`**, **`Text`**, **`Rectangle`**, ...
  - ออกแบบเองเพิ่มเติมได้

ประเภทข้อมูล : วัตถุประสงค์

- ระบบใช้เนื้อที่หน่วยความจำอย่างเหมาะสม
  - **byte** (1 ไบต์), **short** (2 ไบต์), **int** (4 ไบต์), **long** (8 ไบต์), **float** (4 ไบต์), **double** (8 ไบต์), **char** (2 ไบต์)
- ระบบประมวลผลอย่างมีประสิทธิภาพ
  (เช่น ประมวลผลจำนวนเต็มแบบ **int** ไม่ช้ากว่าแบบ **long**)
- compiler ตรวจสอบความถูกต้องของโปรแกรมได้
- ผู้เขียนบอกจุดประสงค์ของการใช้ข้อมูล
- คนอื่นอ่านเข้าใจโปรแกรมได้ง่าย

ประเภทจำนวนเต็ม (integer)

- จำนวนเต็ม (ไม่มีจุดทศนิยม)

  เช่น 1023, 76289, 0, –10, +67392
- มีสี่ขนาด ขึ้นกับช่วงของจำนวนที่ใช้
  - **`byte`** (-128 ถึง 127)
  - **`short`** (-32678 ถึง 32767)
  - **`int`** (-2147483648 ถึง 2147483647)
  - **`long`** (-2^{63} ถึง 2^{63} - 1)

โดยทั่วไปใช้ **`int`**

ประเภทจำนวนจริง (real number)

- จำนวนที่มีจุดทศนิยมได้

  เช่น 102.0, 3.14159, 0.0, -10.8, +0.3333

- จาวาเก็บจำนวนจริงแบบ floating point

  10000000 เก็บ 0.1×10^{8} , 0.0000012 เก็บ 0.12×10^{-5}

- มีสองขนาด ขึ้นกับความละเอียดของจำนวนที่เก็บ
  - **`float`** เก็บประมาณ 6-9 ตำแหน่งหลังจุดทศนิยมในช่วง

    (-3.4×10^{38} ถึง -1.4×10^{-45} ศูนย์ และ 1.4×10^{-45} ถึง 3.4×10^{38})
  - **`double`** เก็บประมาณ 15-17 ตำแหน่งหลังจุดทศนิยมในช่วง

    (-1.8×10^{308} ถึง -4.9×10^{-324} ศูนย์ และ 4.9×10^{-324} ถึง 1.8×10^{308})

โดยทั่วไปใช้ `double`

ทำไมต้องมีจำนวนเต็ม จำนวนจริงน่าจะพอ ?

- **`float`** เก็บข้อมูลในช่วงที่กว้างกว่า **`int`** มากๆ
- **`int`** เก็บข้อมูลได้ทุกๆ ตัวที่เป็นไปได้ในช่วง
- **`float`** เก็บได้บางตัวเท่านั้น (เพราะเก็บจำนวนตัวเลขหลังจุดทศนิยมได้จำกัด)
- ตัวอย่าง
  - **`int`** เก็บค่า 2147483647 ได้
  - **`float`** เก็บค่า 2147483647 ไม่ได้
    ถ้ายัดเยียดให้ จะเก็บเป็น 2.14748365×10^9

ประเภทข้อมูลพื้นฐาน ประมวลผลได้เร็วมาก เพราะมีขนาดจำกัด และฮาร์ดแวร์รองรับการทำงานโดยตรง

ประเภทอักขระ (character)

- ข้อมูลแบบ **char** คือตัวอักขระหนึ่งตัว
- อักขระ (character) คือ
  - ตัวอักษร (ในภาษาต่างๆ)
    เช่น 'a', 'b', 'A', 'B', 'Ω', 'Ж', 'ก', 'ข'
  - ตัวเลข 0-9 (ในภาษาต่างๆ)
    เช่น '1', '2', '๑', '๒'
  - สัญลักษณ์พิเศษ (ในภาษาต่างๆ)
    เช่น '#', '$', '฿', '๛'
  - รหัสควบคุมอุปกรณ์
    เช่น `'\n'` ตัวขึ้นบรรทัดใหม่ `'\t'` ตัว tab

ประเภทสายอักขระ (`String`)

- **`String`** คือลำดับของ **`char`** จำนวนศูนย์ตัวขึ้นไป
  - `"engineering"`
  - `"webboy@universe.org"`
  - `"วิศวกรรมศาสตร์"`
  - `"Hello How are you?"`
  - `""`

`"1004"` ไม่เหมือนกับ 1004

`""` มีความหมาย แต่ `"` ไม่มี

ประเภทตรรกะ (boolean)

- **`boolean`**
- มีได้แค่สองค่าคือ **`true`** และ **`false`** เท่านั้น
- 1 ไม่ได้แทน **`true`** 0 ไม่ได้แทน **`false`**

ตัวแปร (Variable)

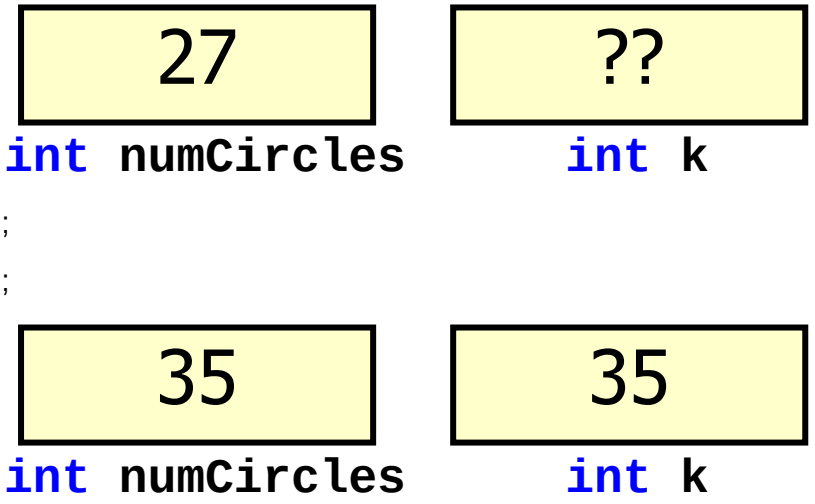
- ตัวแปรเป็นที่เก็บข้อมูลชั่วคราวของโปรแกรม
- ตัวแปรทุกตัวมีชื่อและประเภทข้อมูลกำกับ
- ต้องประกาศตัวแปรก่อนที่จะนำมาใช้

```
public class Zombie {

  public static void main(final String[] args ) {
    int       counter = 0;
    double    radius = 5.225;
    boolean   success = false;
    String    name = "somchai";

    ...
  }
}
```

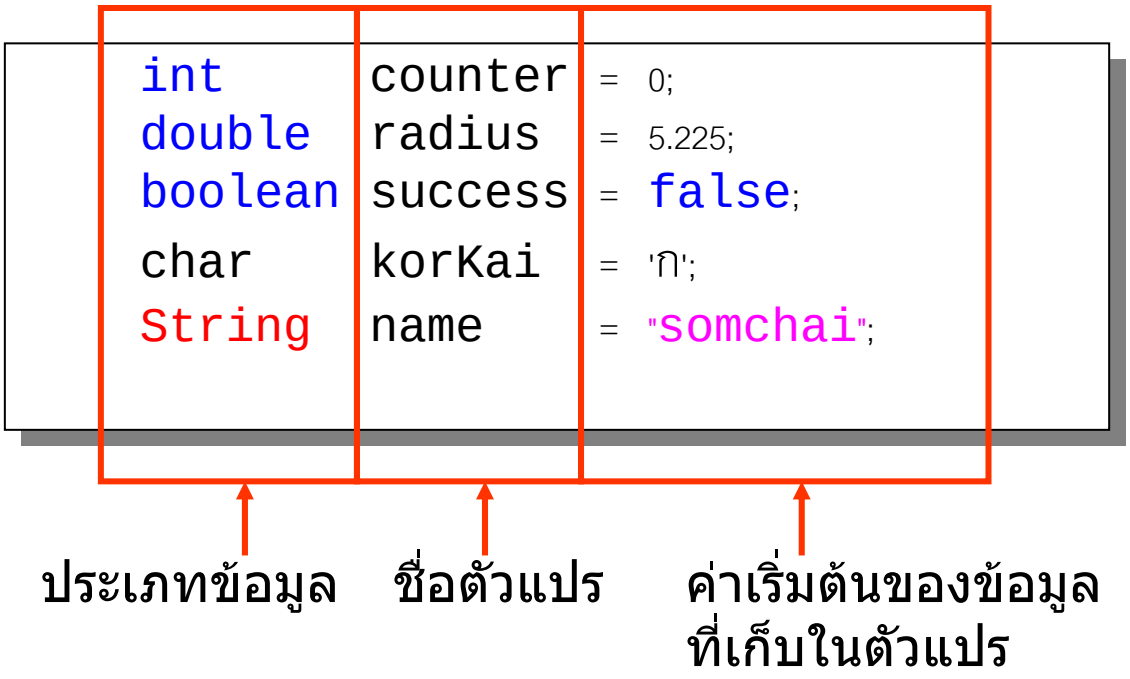
ตัวแปร

- ตัวแปรเป็นที่เก็บข้อมูลชั่วคราว
- การนำข้อมูลที่เก็บในตัวแปรมาใช้ ไม่ทำให้ข้อมูลที่เก็บในตัวแปรสูญหาย
- การนำข้อมูลใหม่เก็บใส่ตัวแปร เป็นการแทนที่ข้อมูลเดิม (ระบบไม่เก็บสำเนาข้อมูลเดิมไว้)

```
int numCircles = 27, k;

System.out.println( numCircles );
System.out.println( numCircles );

k = 35;
numCircles = k;
```



การประกาศตัวแปร



ประกาศบรรทัดละหลายตัว หรือจะไม่ใส่ค่าเริ่มต้นก็ได้

```
int      i = 100, j = 200;
double   x0, y0, z0;
double   x1, y1 = 4.5, z1 = 7.8;
```

ตัวอย่างการประกาศตัวแปรที่ผิด

ไม่ได้ปิดท้ายบรรทัดด้วย ;

```
int     counter
```

ใส่ค่าเริ่มต้นที่ผิดประเภท

```
int     counter = 0.75;
double  radius  = "15.25";
boolean success = 1;
```

นิยามประเภทข้อมูลที่ไม่รู้จัก

```
integer counter;
doublE  radius  = "15.25";
string  title = "WWW";
```

ตั้งชื่อตัวแปรผิดกฎ

```
int     boolean;
double  public = 12.7;
String  12X = "WWW";
```

กฎการตั้งชื่อ

- ชื่อประกอบด้วยตัวอักษร ตัวเลข ตัว $ หรือ _ ก็ได้
- ชื่อห้ามขึ้นต้นด้วยตัวเลข
- ชื่อยาวๆ ได้ไม่เป็นไร
- ตัวอักษรตัวใหญ่ไม่เหมือนตัวเล็ก
- ต้องไม่ซ้ำกับคำสงวนของภาษาจาวา
- ต้องไม่ซ้ำกับชื่ออื่นๆ ของโปรแกรม (เช่น ชื่อ class)

ตัวอย่างถูก

```
int17  butterCup  Public  int2String  day_of_week
```

ตัวอย่างผิด

```
7zean  I.love.you  public  ohOH!  ed-edd-n-eddy
```

คำสงวนในภาษาจาวา

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
extends	false	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	null	package	private	protected
public	return	short	static	strictfp
super	switch	synchronized	this	throw
throws	transient	true	try	void
volatile	while			

ห้ามนำคำสงวนมาตั้งเป็นชื่อตัวแปร

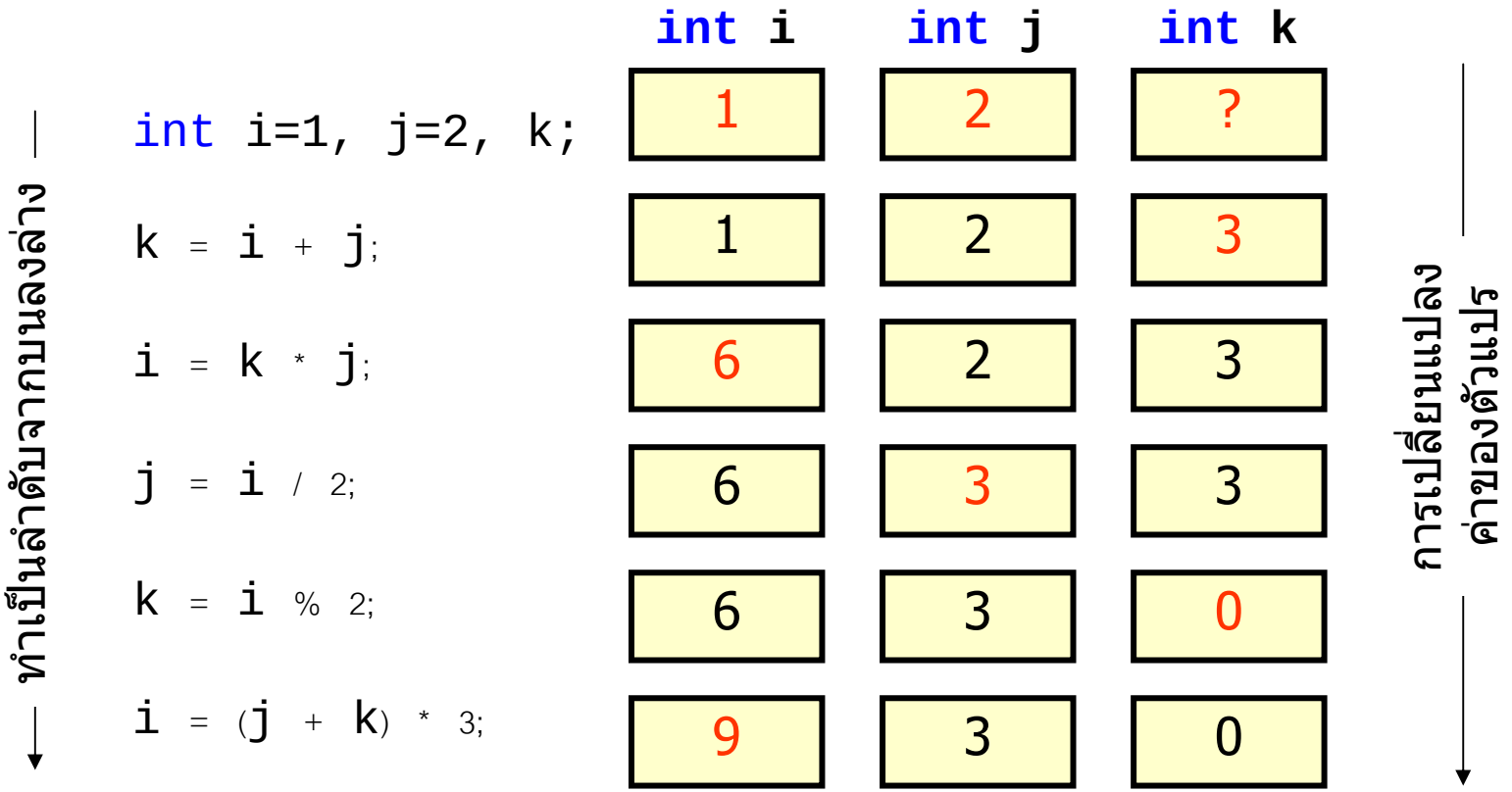
ธรรมเนียมนิยมในการตั้งชื่อ

- ตัวแปร : ขึ้นต้นด้วยตัวอักษรตัวเล็ก
- class : ขึ้นต้นด้วยตัวอักษรตัวใหญ่
- หลีกเลี่ยงการใช้ _ และ $
- สื่อความหมาย
- ถ้าชื่อประกอบด้วยคำหลายคำ เขียนแบบ "หลังอูฐ"

```
numStudents   dayOfWeek     imageBuffer

tvChannel     customerID    initialVelocity
```

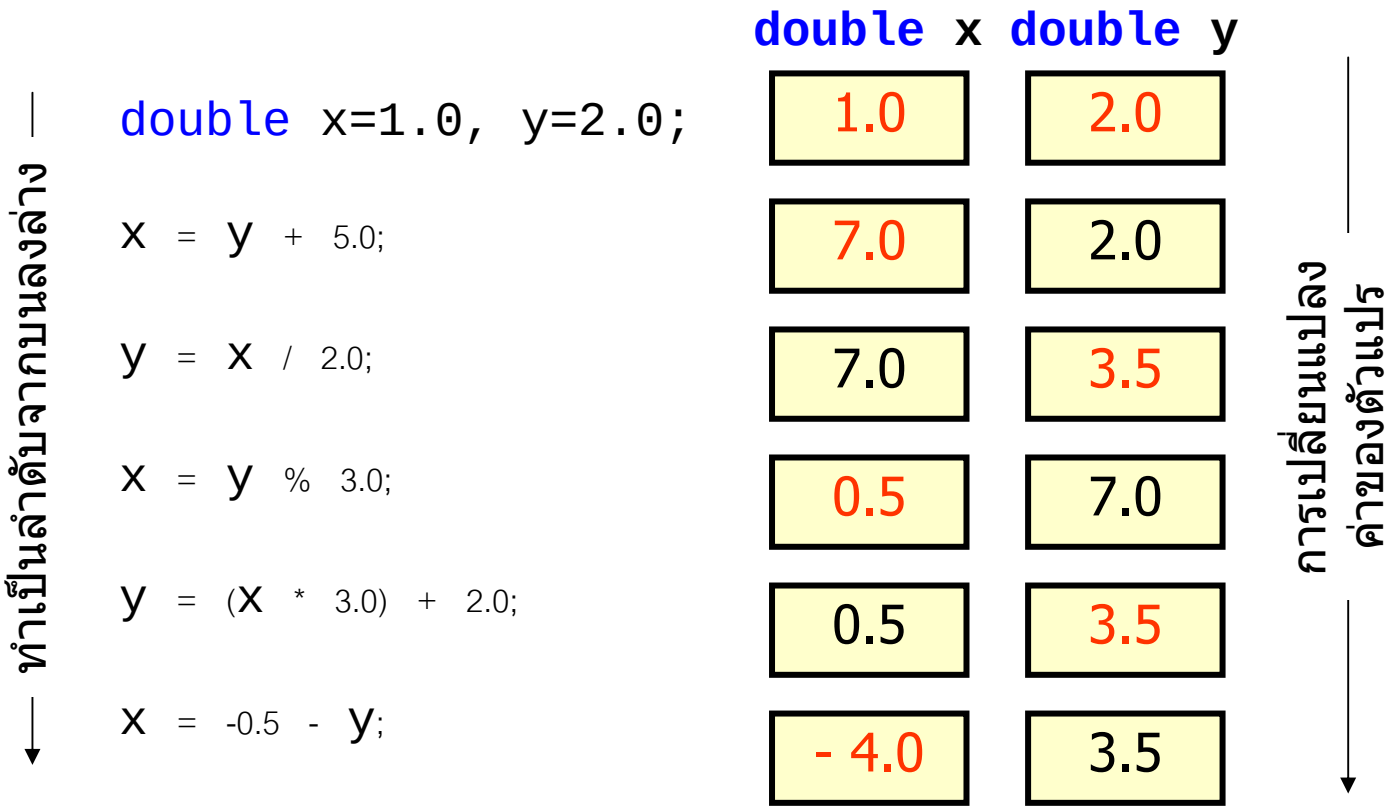
ตัวปฏิบัติการพื้นฐาน : จำนวนเต็ม

- บวก ลบ คูณ หาร และเศษเหลือจากการหาร (+ – * / %)



ตัวปฏิบัติการพื้นฐาน : จำนวนจริง

- บวก ลบ คูณ หาร และเศษเหลือจากการหาร (+ – * / %)



การให้ค่าตัวแปร (Assignment)

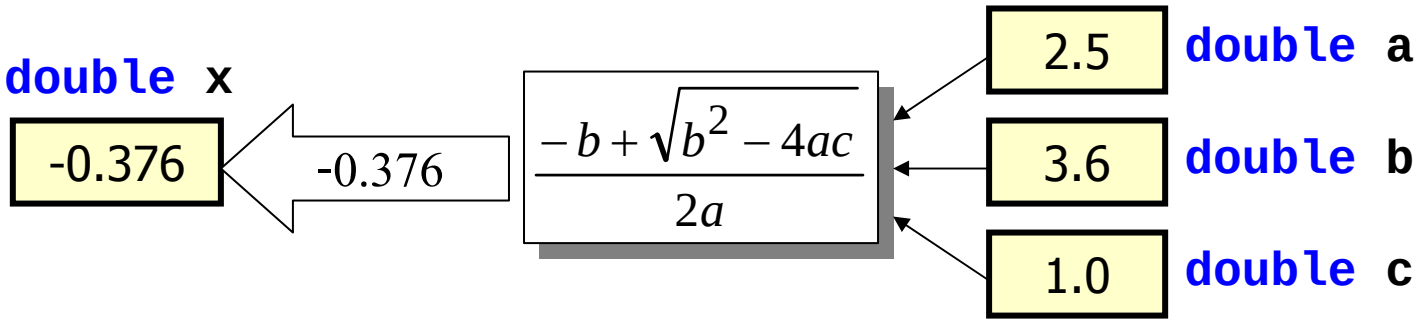
- ใช้เครื่องหมาย =

```
ตัวแปร = นิพจน์ ;
```

ชื่อตัวแปรที่รับค่า

นิพจน์ ที่ให้เป็นผลลัพธ์ซึ่งมีประเภทข้อมูลเดียวกับตัวแปรที่รับค่า

```
x = (-b + Math.sqrt(b*b - 4*a*c)) / (2.0*a);
```



นิพจน์ (Expression)

- ค่าคงตัว ตัวแปร ฟังก์ชัน (หรือ method) หรือ
- การรวมกันของค่าคงตัว ตัวแปร ฟังก์ชัน นิพจน์ย่อย ด้วยตัวปฏิบัติการ และเครื่องหมายวงเล็บปิดเปิด ที่สามารถคำนวณค่าได้

$$\frac{9c}{5} + 32$$

```
((9.0 * c) / 5.0) + 32.0
(((9.0) * (c)) / (5.0)) + (32.0)
9.0*c/5.0+32.0
```

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

```
(-b + Math.sqrt(b*b - 4.0*a*c)) / (2.0*a)
```

โดยทั่วไปจะเติมช่องว่างเพื่อให้อ่านง่าย และเติมวงเล็บเพื่อกำจัดความกำกวม

ตัวอย่าง

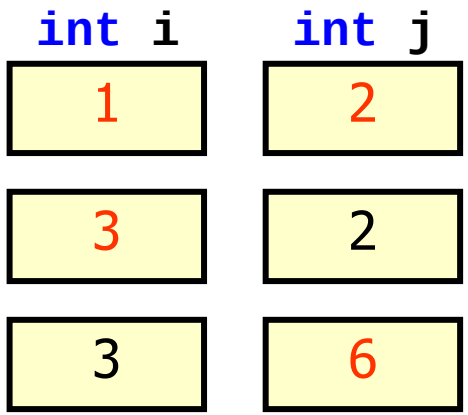
```
i = j;
```

นำค่าของตัวแปร j ไปใส่ในตัวแปร i

```
i = i + j;
```

นำค่าของตัวแปร i และ j มาบวกกัน
แล้วนำผลบวกที่ได้ไปใส่ในตัวแปร i

```
int i=1, j=2;

i = i + j;

j = j + j + j;
```



ตัวอย่าง

$$x \leftarrow -x$$

```
x = -x;
```

$$f \leftarrow \frac{9c}{5} + 32$$

```
f = ((9.0 * c) / 5.0) + 32.0;
```

$$c \leftarrow \frac{5(f - 32)}{9}$$

```
c = 5 * (f - 32.0) / 9.0;
```

$$d \leftarrow \sqrt{\Delta x^2 + \Delta y^2}$$

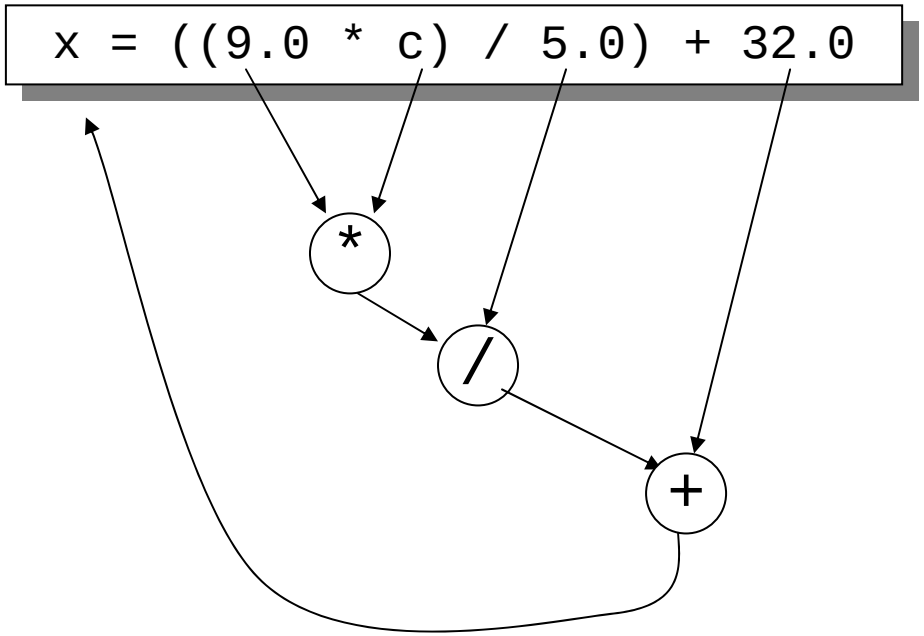
```
d = Math.sqrt(dx*dx + dy*dy);
```

$$d \leftarrow \sqrt{\Delta x(\Delta x + \Delta y)\Delta y}$$

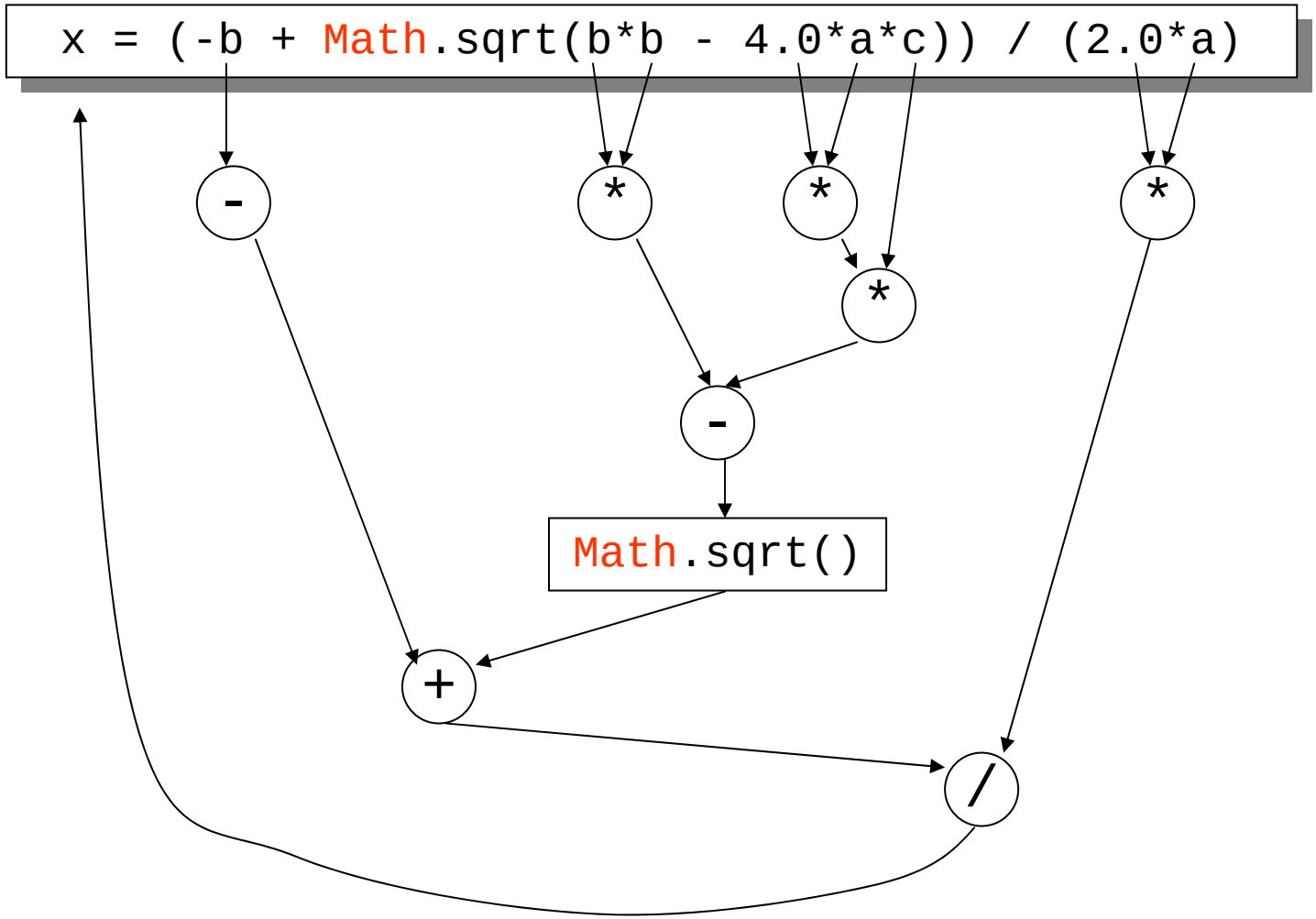
กฎการพิจารณาลำดับการคำนวณ

- ทำภายในวงเล็บทำก่อน
- ในกรณีมีตัวปฏิบัติการหลายตัว ทำตามลำดับดังนี้
  - การเรียกใช้ method
  - unary – +
  - * / %
  - + –
  - =
- ทำจากซ้ายไปขวา (ในกรณีที่ operator มีศักดิ์ศรีเท่ากัน)

วงเล็บกำหนดลำดับการคำนวณ



ตัวปฏิบัติการกำหนดลำดับการคำนวณ



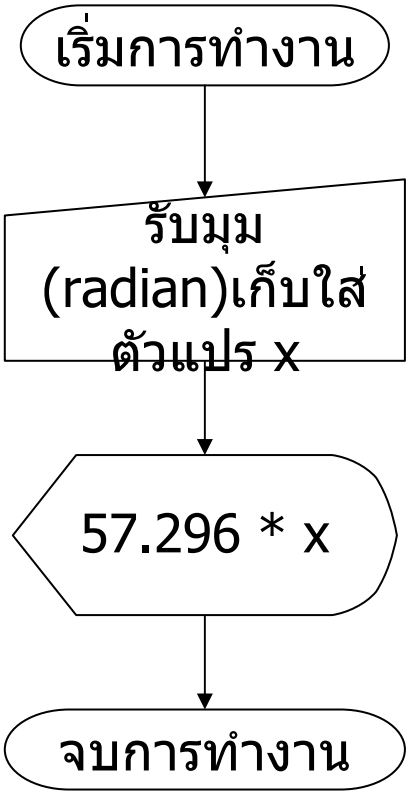
ตัวอย่างผิดๆ

```
int i, k = 9;
double x = 1.0;

k = i;                  // i is uninitialized
i = x;                  // assign double to int
i = 3k;                 // no operator
3.0 * x + 5.78;         // not a complete statement
i = 3--k;               // should be i = 3 - (-k)
i = (k + 5;             // incomplete parentheses
```

โปรแกรมแปลงหน่วยเรเดียนเป็นองศา

- Requirement
  - โปรแกรมรับมุมหน่วยเป็นเรเดียนแล้วแปลงเป็นองศา
- Analysis
  - รับมุม x เป็นจำนวนจริงผ่านทางแป้นพิมพ์
  - x เรเดียนมีค่าเท่ากับ $\frac{360x}{2\pi} \approx 57.296x$ องศา
  - แสดงมุมหน่วยเป็นองศาที่หาได้ทางจอภาพ
- Design
  - เขียนผังงาน
- Implementation
  - เขียนโปรแกรม

ผังงาน โปรแกรม และการทดสอบ




```
import jlab.JLabIO;

public class Radian2Degree {

  public static void main(String[] args) {
    double x, d;
    x = JLabIO.readDouble("input radian : ");
    d = 57.296 * x;
    System.out.println(d);
  }
}
```

```
JLab>java Radian2Degree
input radian : 3
171.888
JLab>
```

โปรแกรมแปลงหน่วยเรเดียนเป็นองศา

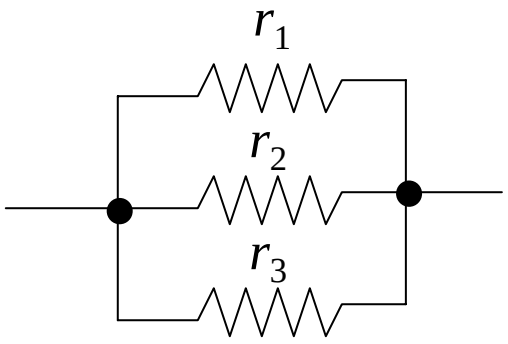
- ในจาวามีค่า π ให้ใช้
- **`Math.PI =`** 3.141592653589793238 46

```
import jlab.JLabIO;

public class Radian2Degree {

  public static void main(String[] args) {
    double x, d;
    x = JLabIO.readDouble("input radian : ");
    d = 180.0 * x / Math.PI;
    System.out.println( x + " radians = " + d + " degrees");
  }
}
```

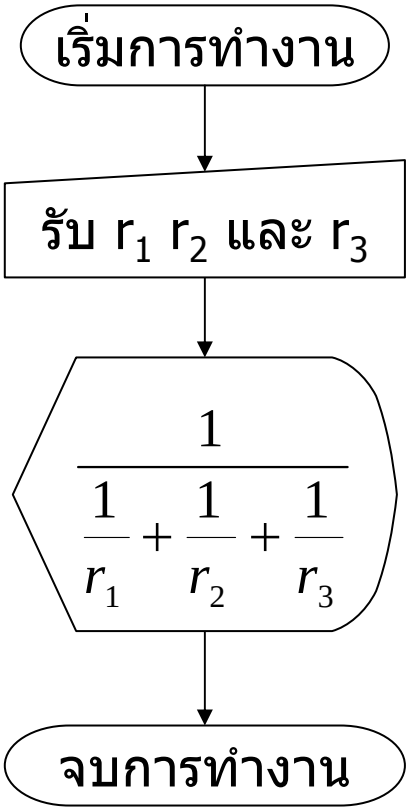
```
JLab>java Radian2Degree
input radian : 3
3.0 radians = 171.88733853924697 degrees
JLab>
```

โปรแกรมหา R จากการต่อ R 3 ตัวแบบขนาน

- Requirement
  - โปรแกรมหาความต้านรวมจากการต่อความต้านทาน 3 ตัวแบบขนาน
- Analysis
  - รับค่าความต้านทาน (K ohms) 3 ตัวจากแป้นพิมพ์
  - แสดงค่าความต่านทานรวม $\frac{1}{\frac{1}{r_1}+\frac{1}{r_2}+\frac{1}{r_3}}$ ทางจอภาพ
- Design
  - เขียนผังงาน
- Implementation
  - เขียนโปรแกรม



ผังงาน โปรแกรม และการทดสอบ



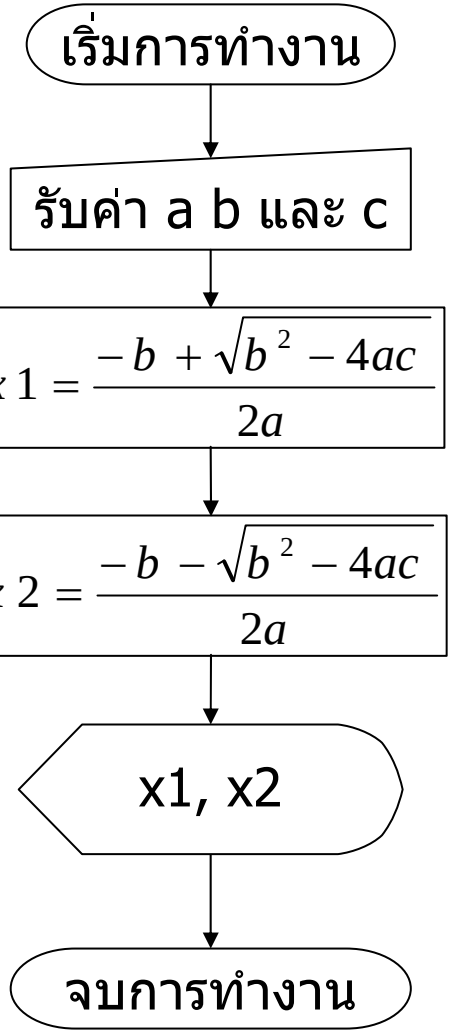

```
import jlab.JLabIO;

public class ParallelResistance {

  public static void main(String[] args) {
    double r1, r2, r3, r;
    r1 = JLabIO.readDouble("r1 (Kohms) = ");
    r2 = JLabIO.readDouble("r2 (Kohms) = ");
    r3 = JLabIO.readDouble("r3 (Kohms) = ");
    r = 1.0 / (1.0/r1 + 1.0/r2 + 1.0/r3);
    System.out.println("R (parallel) = "
                       + r + " Kohms");
  }
}
```

```
JLab>java ParallelResistance
r1 (Kohms) = 22
r2 (Kohms) = 33
r3 (Kohms) = 100
R (parallel) = 11.660777385159012 Kohms
JLab>
```

โปรแกรมหารากของสมการกำลังสอง

- Requirement
  - โปรแกรมหารากของสมการ $ax^2 + bx + c = 0$
- Analysis
  - รับจำนวนจริง a b และ c ผ่านทางแป้นพิมพ์
  - คำนวณรากสองรากจากสูตร $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
  - แสดงรากทั้งสองที่หาได้ทางจอภาพ
- Design
  - เขียนผังงาน
- Implementation
  - เขียนโปรแกรม

ผังงาน


เริ่มการทำงาน
รับค่า a b และ c
$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$
$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$
x1, x2
จบการทำงาน

โปรแกรมและการทดสอบ

```
import jlab.JLabIO;

public class QRoot {
  public static void main(String[] args) {
    double a, b, c, x1, x2;

    a = JLabIO.readDouble("a = ");
    b = JLabIO.readDouble("b = ");
    c = JLabIO.readDouble("c = ");

    double t = Math.sqrt(b * b - 4.0 * a * c);
    x1 = (-b + t) / (2.0 * a);
    x2 = (-b - t) / 2.0 / a;
    System.out.println("x1 = " + x1);
    System.out.println("x2 = " + x2);
  }
}
```

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

```
JLab>java QRoot
a = 1
b = 4
c = 3
x1 = -1.0
x2 = -3.0
JLab>
```