2110101 : การทำโปรแกรมคอมพิวเตอร์

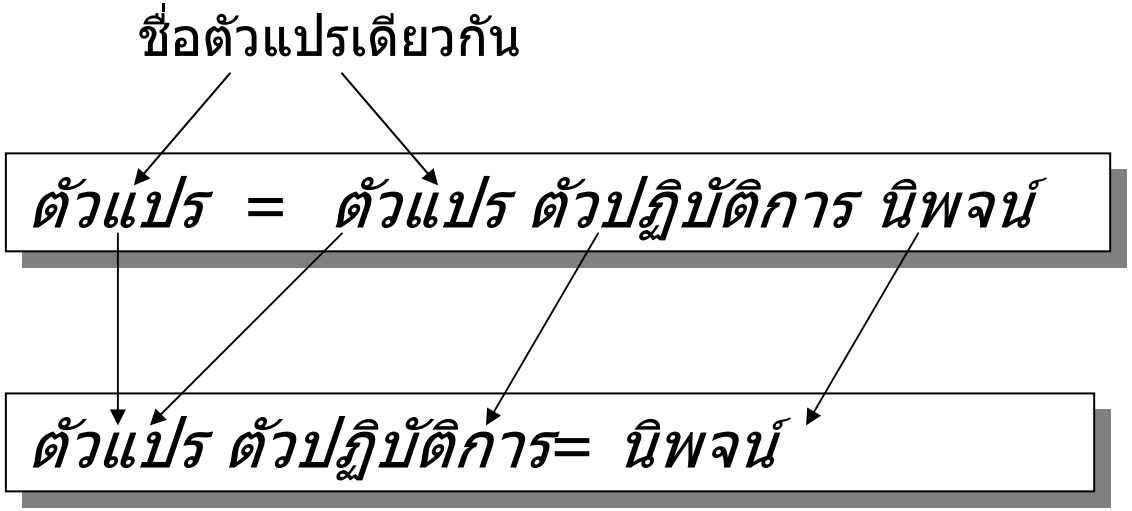
ตัวปฏิบัติการพื้นฐาน
การเปลี่ยนประเภทข้อมูล

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

- ตัวปฏิบัติการ =
- ตัวปฏิบัติการเพิ่ม และลดค่าข้อมูลอีกหนึ่ง
- การเขียนค่าคงตัวของจำนวนเต็มและจำนวนจริง
- การเปลี่ยนประเภทข้อมูล

วิธีเขียนลัด




```
x = x - 5.0;
x = 3.0 * x;
x = x % (y + z * a);
x = x / (2.0 * x);
```

```
x -= 5.0;
x *= 3.0;
x %= (y + z * a);
x /= (2.0 * x);
```

ตัวปฏิบัติการกับ = ต้องติดกัน

โปรแกรมหาค่าเฉลี่ย

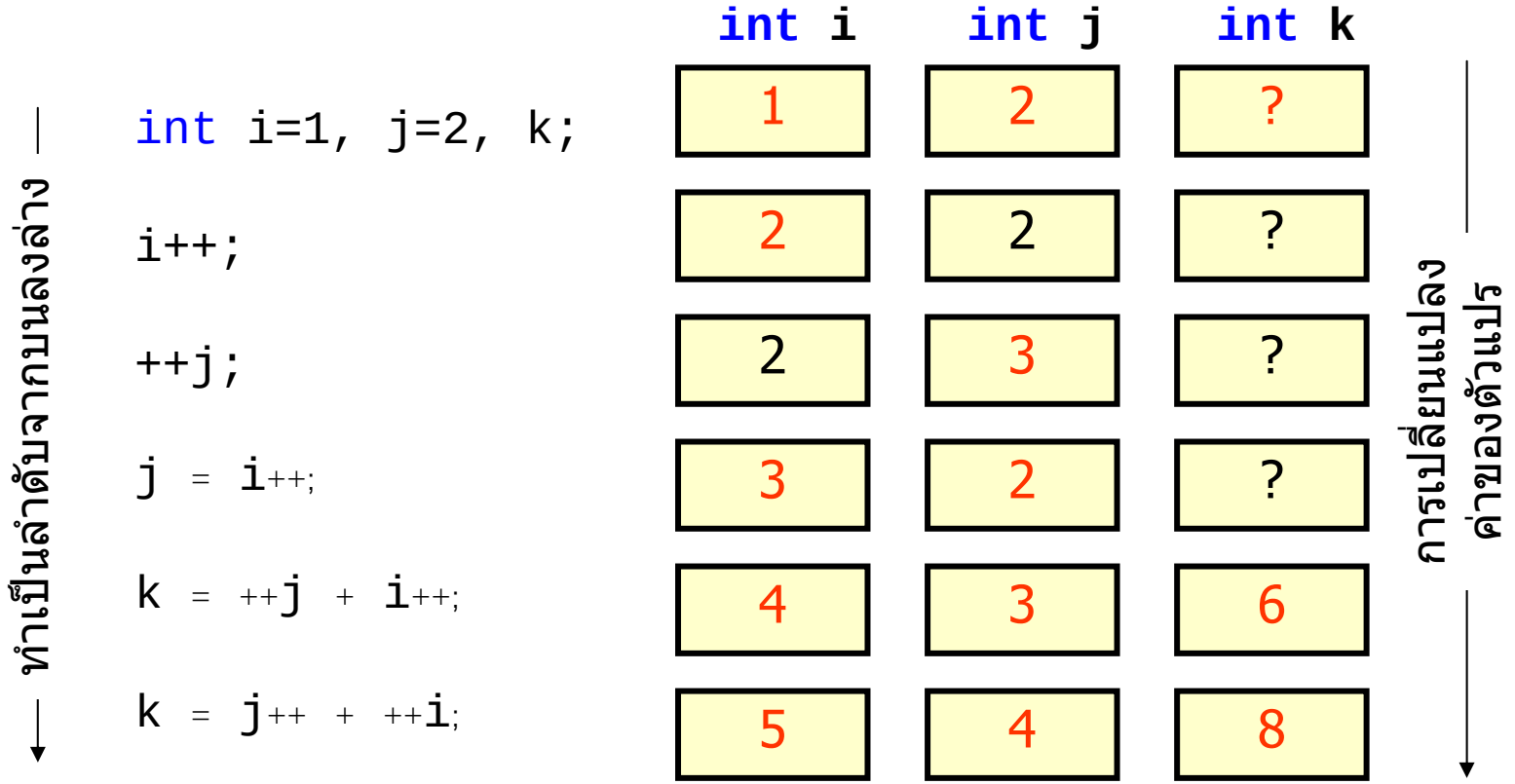
```
import jlab.JLabIO;

public class Mean {
  public static void main(String [] args) {
    double x = 0.0;
    x += JLabIO.readDouble("data #1 : ");
    x += JLabIO.readDouble("data #2 : ");
    x += JLabIO.readDouble("data #3 : ");
    x += JLabIO.readDouble("data #4 : ");
    x += JLabIO.readDouble("data #5 : ");
    System.out.println("mean = " + (x / 5.0) );
  }
}
```

```
JLab>java Mean
data #1 : 12
data #2 : 35.5
data #3 : 16.8
data #4 : 19.0
data #5 : 2.2
mean = 17.1
JLab>
```

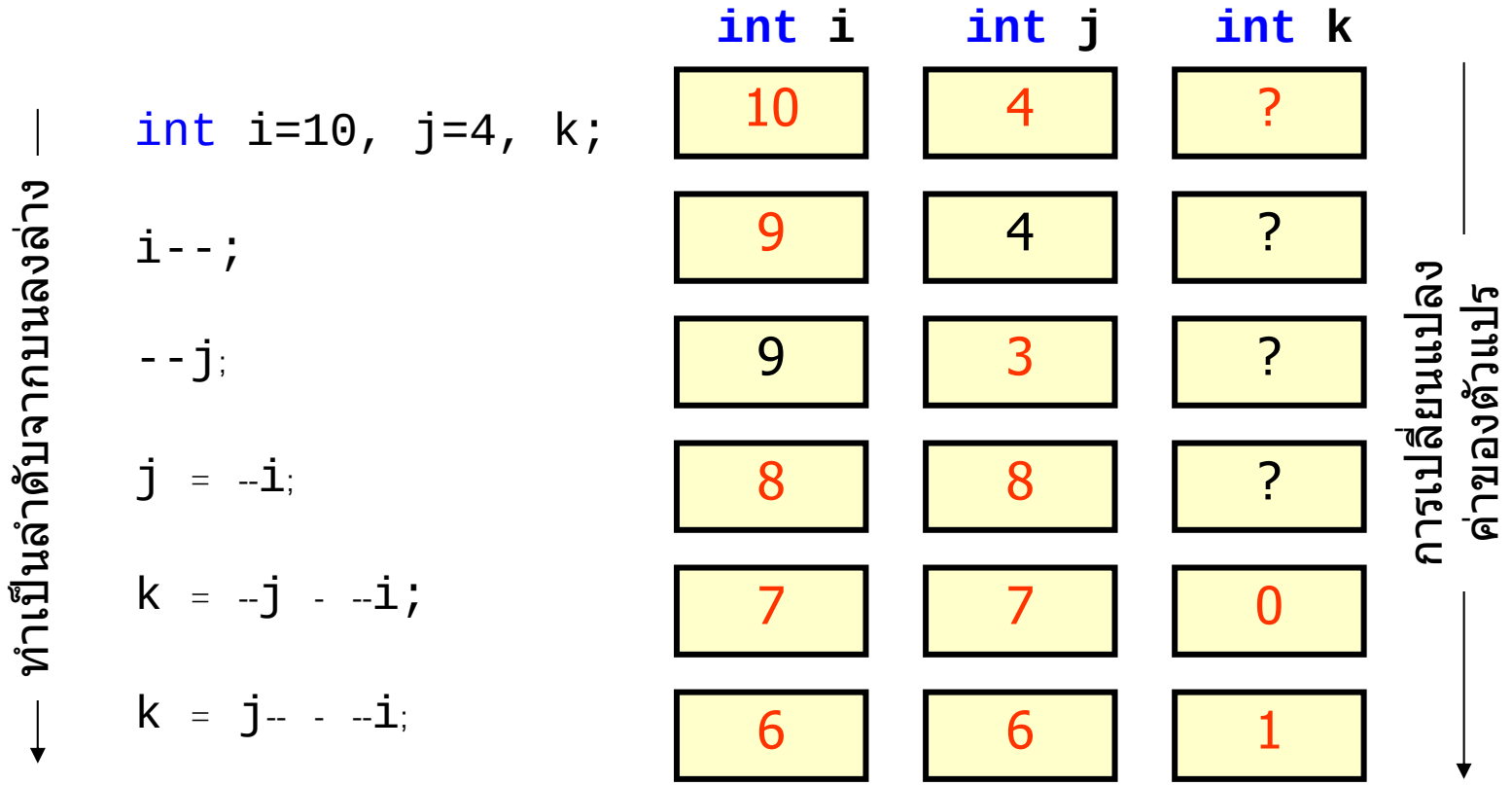
ตัวปฏิบัติการพื้นฐาน : จำนวนเต็มและจริง (ต่อ)

- การเพิ่มค่าทีละหนึ่ง มีสองแบบ
  - **a++** นำค่าของ **a** ไปใช้ แล้วค่อยเพิ่มค่าของ **a** ขึ้นอีกหนึ่ง
  - **++a** เพิ่มค่าของ **a** ขึ้นอีกหนึ่ง แล้วค่อยนำค่าไปใช้

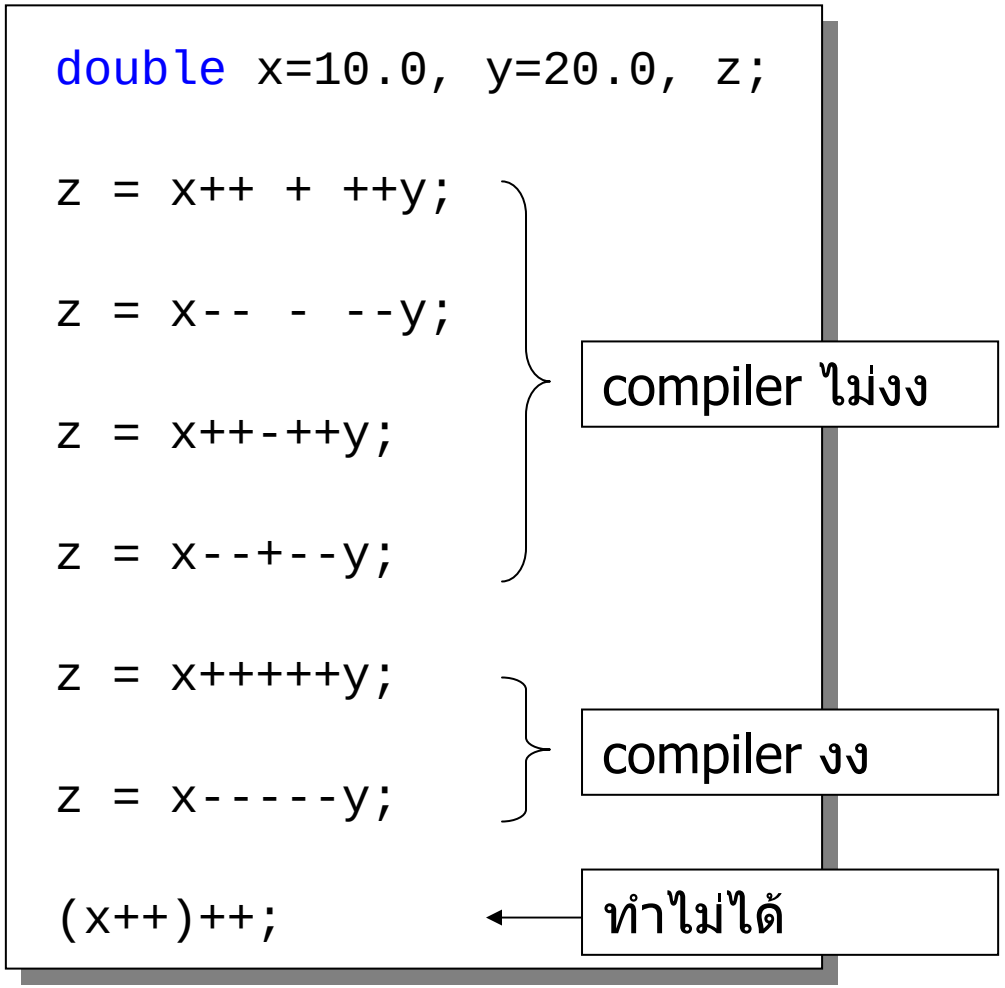


ตัวปฏิบัติการพื้นฐาน : จำนวนเต็มและจริง (ต่อ)

- การลดค่าทีละหนึ่ง มีสองแบบ
  - **a--** นำค่าของ **a** ไปใช้ แล้วค่อยลดค่าของ **a** ลงหนึ่ง
  - **--a** ลดค่าของ **a** ลงหนึ่ง แล้วค่อยนำค่าไปใช้



ตัวอย่างงงงง


double x=10.0, y=20.0, z;
z = x++ + ++y;
z = x-- - --y;
z = x++-++y;
z = x--+--y;
z = x+++++y;
z = x-----y;
(x++)++;
compiler ไม่งง
compiler งง
ทำไม่ได้

การเขียนค่าคงตัวที่เป็นจำนวนเต็ม

- ต้องไม่มี comma
- long ต้องปิดท้ายด้วย l หรือ L (แอลเล็กหรือใหญ่)
- ฐานสิบ : ห้ามเขียนเริ่มด้วย 0 (ยกเว้นค่า 0)
- ฐานแปด : ต้องเริ่มด้วย 0
- ฐานสิบหก : ต้องเริ่มด้วย 0x


```
int base10 = 2545;
int base8 = 04761;
int base16 = 0x9F1;
```

```
long base10 = 2545L;
long base8 = 04761L;
long base16 = 0x9F1L;
```

การเขียนค่าคงตัวที่เป็นจำนวนจริง

- float ต้องปิดท้ายด้วย f หรือ F
- double ปิดท้ายด้วย d หรือ D หรือไม่มีก็ได้
- เขียนในรูปแบบย่อ ที่มีการคูณด้วย 10 ยกกำลังได้ เช่น
  - 1230000000.0 เขียนเป็น
    - **123E7** (ซึ่งแทน 123 x 10^7)
    - **12.3E8** (ซึ่งแทน 12.3 x 10^8)
    - **0.0123E11** (ซึ่งแทน 1.23 x 10^{11})
    - ...
  - -0.000567 เขียนเป็น
    - **-567E-6** (ซึ่งแทน -567 x 10^{-6})
    - **-5.67E-4** (ซึ่งแทน -5.67 x 10^{-4})
    - **-0.567E-3** (ซึ่งแทน -0.567 x 10^{-3})
    - ...

ใช้ตัว e เล็กก็ได้

เลขชี้กำลังต้องเป็นจำนวนเต็ม

การเปลี่ยนประเภทของข้อมูล

- บางครั้งเราอาจต้องการเปลี่ยนข้อมูลประเภทหนึ่งไปยังอีกประเภทหนึ่ง เช่น
  - นิพจน์ที่ใช้มีค่าคงตัวและตัวแปรหลากหลายประเภท
  - ใช้ method ของระบบที่รับข้อมูลคนละประเภทกับที่เรามี
  - มีการนำจำนวนเต็มมาหารกัน แต่ต้องการผลซึ่งมีเศษด้วย
  - ผลที่ได้จากการคำนวณมีเศษ แต่ต้องการปัดทิ้ง
  - ...

```
int dx, dy, halfDist;
...
halfDist = 0.5 * Math.sqrt(dx*dx + dy*dy);
```

$$d = \left\lfloor \frac{\sqrt{\Delta x^2 + \Delta y^2}}{2} \right\rfloor$$

Type Checking

- การให้ค่าผิดประเภทกับตัวแปรอาจแปลไม่ผ่าน

```
public class TypeChecking {
  public static void main(String[] args) {
    double x = 1.0;
    short k = 100000;
    float f = 1.2;
  }
}
```

```
JLab>javac TypeChecking
TypeChecking.java:4: [#1102] [#1155] อาจสูญเสียความ
แม่นยำของข้อมูลได้ พบ int แต่ต้องการ short, column:15
TypeChecking.java:5: [#1102] [#1155] อาจสูญเสียความ
แม่นยำของข้อมูลได้ พบ double แต่ต้องการ float, column:15
JLab>
```

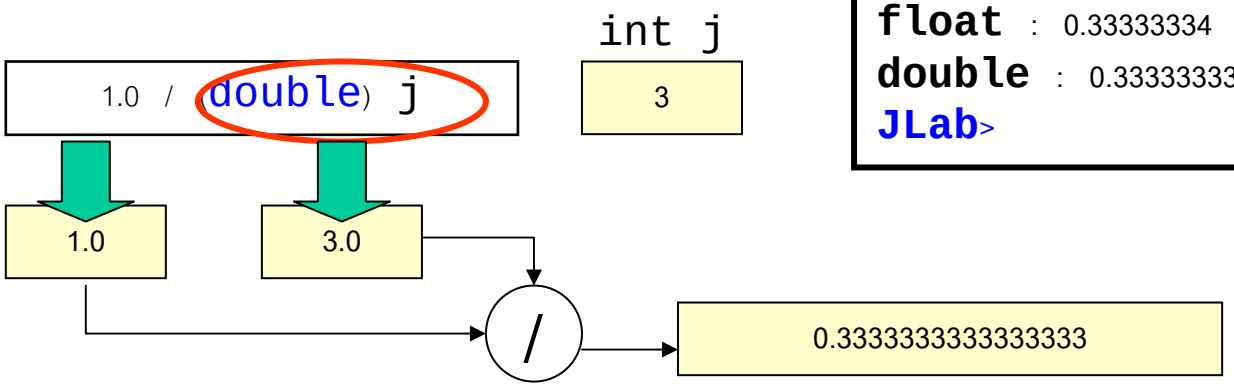
compilation error

การเปลี่ยนประเภทของข้อมูล

- cast operator : ใส่ชื่อประเภทใหม่ภายในวงเล็บไว้หน้านิพจน์ที่ต้องการให้เปลี่ยน

```
public class TypeCasting {
  public static void main(String[] args) {
    int j = 3;
    System.out.println("int : " + (1 / j));
    System.out.println("float : " + (1.0F / (float) j));
    System.out.println("double : " + (1.0 / (double) j ));
  }
}
```

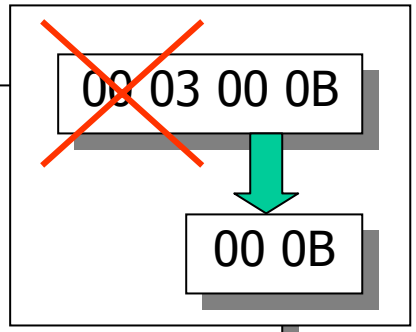
```
JLab>java TypeCasting
int : 0
float : 0.33333334
double : 0.3333333333333333
JLab>
```



ข้อมูลอาจสูญหายในการเปลี่ยนประเภทข้อมูล

- เปลี่ยนจากจำนวนจริงเป็นจำนวนเต็ม เศษจะถูกตัดทิ้ง
- เปลี่ยนไปเป็นประเภทที่รับช่วงข้อมูลได้แคบกว่า ข้อมูลอาจสูญหาย แบบแปลกๆ

```
public class LossFromTypeCasting {
  public static void main(String[] args) {
    int i = (int) 12.999;
    int k = 196620; // 0x3000B
    short tiny = (short) k;
    System.out.println("i = " + i);
    System.out.println("tiny = " + tiny);
  }
}
```




```
JLab>java LossFromTypeCasting
i = 12
tiny = 11
JLab>
```

narrowing primitive conversion

Narrowing Primitive Conversion

- **byte** เป็น **char**
- **short** เป็น **byte** หรือ **char**
- **char** เป็น **byte** หรือ **short**
- **int** เป็น **byte**, **short**, หรือ **char**
- **long** เป็น **byte**, **short**, **char**, หรือ **int**
- **float** เป็น **byte**, **short**, **char**, **int**, หรือ **long**
- **double** เป็น **byte**, **short**, **char**, **int**, **long**, หรือ **float**

ระวังเรื่องลำดับการทำงาน

- cast operator จะถูกทำก่อน + – * / %
  แต่หลัง ( ) และ unary + –

```
public class PrecedenceOfCasting {
  public static void main(String[] args) {
    double x = 1.9, y = 10.9;
    System.out.println("(int) x * y = " + ((int) x * y));
    System.out.println("(int) y * x = " + ((int) y * x));
    System.out.println("(int) (x * y) = " + ((int) (x * y)));
    System.out.println("(int) (y * x) = " + ((int) (y * x)));
  }
}
```

```
JLab>java PrecedenceOfCasting
(int) x * y = 10.9
(int) y * x = 19.0
(int) (y * x) = 20
(int) (x * y) = 20
JLab>
```

การเปลี่ยนประเภทของข้อมูลอย่างอัตโนมัติ

- เมื่อให้ค่ากับตัวแปรที่รับช่วงข้อมูลได้กว้างกว่า
- + – * / หรือ % ในนิพจน์รับข้อมูลต่างประเภทกัน
  - ถ้ามีข้อมูลหนึ่งเป็น **double** จะเปลี่ยนอีกตัวเป็น **double** ด้วย
  - ไม่เช่นนั้น ถ้ามีข้อมูลหนึ่งเป็น **float** จะเปลี่ยนอีกตัวเป็น **float** ด้วย
  - ไม่เช่นนั้น ถ้ามีข้อมูลหนึ่งเป็น **long** จะเปลี่ยนอีกตัวเป็น **long** ด้วย
  - ไม่เช่นนั้น เปลี่ยนข้อมูลทั้งคู่เป็น **int**

```
public class AutoConversion {
  public static void main(String[] args) {
    int i = 7, j = 8;
    double x = 0.5F * i + j;
    System.out.println("x = " + x);
  }
}
```

```
JLab>java AutoConversion
x = 11.5
JLab>
```

widening primitive conversion

Widening Primitive Conversion

- **byte** เป็น **short**, **int**, **long**, **float**, หรือ **double**
- **short** เป็น **int**, **long**, **float**, หรือ **double**
- **char** เป็น **int**, **long**, **float**, หรือ **double**
- **int** เป็น **long**, **float**, หรือ **double**
- **long** เป็น **float** หรือ **double**
- **float** เป็น **double**

อาจทำให้หลักท้ายๆ ของจำนวนสูญหายได้ เมื่อ
- เปลี่ยน **int** หรือ **long** เป็น **float**
- เปลี่ยน **long** เป็น **double**

```
public class Widening {
  public static void main(String[] args) {
    int big = 1234567890;
    float approx = big;
    System.out.println("big = " + big);
    System.out.println("approx = " + approx);
  }
}
```

```
JLab>java Widening
big = 1234567890
approx = 1.23456794E9
JLab>
```

คำถาม

```
double x = (1/2 + 1/2);
```

x มีค่าเท่าใด ?

```
float  y = 1.0/2.0;
```

y มีค่าเท่าใด ?

```
double z = 4d + 5;
```

z มีค่าเท่าใด ?

```
double u = 0.0 + 1/2;
```

u มีค่าเท่าใด ?

```
double v = (0.0 + 1)/2;
```

v มีค่าเท่าใด ?

```
double f = 212.0;
double c = (5/9) * (f - 32);
```

c มีค่าเท่าใด ?

เรื่องต้องรู้

floating point มีขนาดจำกัด

```
public class Imprecision1 {
  public static void main(String[] args) {
    double x = 20.1;
    double y = x + 16.8;
    System.out.println(y);
  }
}
```

```
JLab>java Imprecision1
36.900000000000006
JLab>
```

ลำดับการคำนวณนั้นสำคัญ

```
public class Imprecision2 {
  public static void main(String[] args) {
    double x = 1.0E-200;
    System.out.println(x * x / x);
    System.out.println(x * (x / x));
  }
}
```

```
JLab>java Imprecision2
0.0
1.0E-200
JLab>
```

เรื่องต้องรู้

```
public class DivByZero {
  public static void main(String[] args) {
    int i = 0;
    int j = 5 / i;

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

```
JLab>java DivByZero
java.lang.ArithmeticException: / by zero
        at DivByZero.main(DivByZero.java:6)
Exception in thread "main"
JLab>
```

จำนวนเต็มในจาวาไม่มีค่า infinity
ดังนั้นการหารด้วยศูนย์จะทำให้เกิดความผิดพลาด
ขณะทำงาน (เรียกว่าเกิด exception)

เรื่องต้องรู้

```
public class DivByZero {
  public static void main(String[] args) {
    double i = 0.0;
    double j = 5.0 / i;
    System.out.println(j);
    System.out.println(0.0/i);
 }
}
```

```
JLab>java DivByZero
Infinity
NaN
JLab>
```

floating point ในจาวามีค่า infinity
และ NaN (Not a Number)
ดังนั้นการหารด้วยศูนย์จะไม่เกิด exception