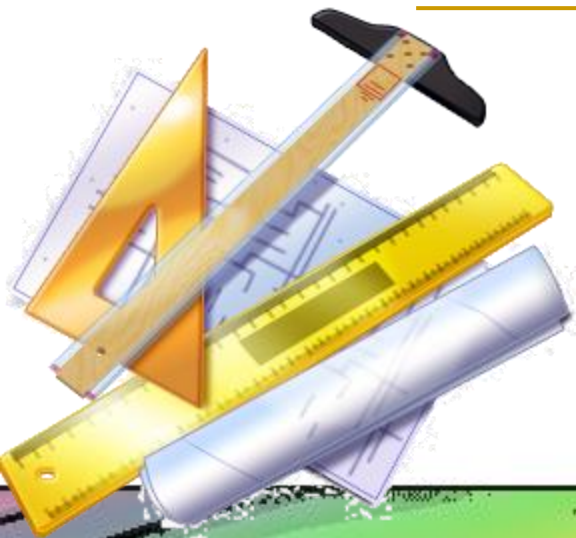


ข้อมูลชนิดพื้นฐาน

Lecture 3

Yaowadee Temtanapat

เยาวดี เต็มธนาภักดิ์



วัตถุประสงค์ของบทนี้

- เรียนรู้พื้นฐานองค์ประกอบของโปรแกรม:
 - Identifiers, ชนิดข้อมูลพื้นฐาน (Primitive), literal, เครื่องหมายการคำนวณ
- เข้าใจการใช้ข้อมูลชนิดตัวเลขและนิพจน์คณิตศาสตร์
- วิเคราะห์นิพจน์คณิตศาสตร์ร่วมกับกฎลำดับ (Precedence rules)
- อธิบายการจัดสรร หน่วยความจำสำหรับตัวแปรชนิดข้อมูลพื้นฐาน
- สามารถกำหนดค่าเริ่มต้นของตัวแปรและค่าคงที่ได้
- เรียนรู้เกี่ยวกับข้อมูลตัวอักษรและสายอักขระ
- เขียนนิพจน์คณิตศาสตร์โดยใช้ Methods ใน Math class
- เขียนโปรแกรมสำหรับการนำเข้าและส่งข้อมูลออกหน้าจอ
- เข้าใจพื้นฐานการเขียนเมทอด

กฎการตั้งชื่อ Identifier

เมื่อเริ่มต้นการโปรแกรม จำเป็นต้องตั้งชื่อสิ่งต่าง ๆ เช่น ชื่อ class, ชื่อตัวแปร, ชื่อ method ซึ่งชื่อเหล่านี้ต้องสอดคล้องกับกฎการตั้งชื่อของจาวา

■ ชื่อของ Identifier ประกอบด้วย

- ❑ ลำดับของตัวอักษร, ตัวเลข, \$ และ _ โดยที่ตัวแรกของชื่อจะต้องไม่เป็นตัวเลข
- ❑ ต้องไม่เป็นคำสงวน (Reserved Word)
- ❑ ใช้ระบุ/ตั้งชื่อ class, object, method และอื่น ๆ
- ❑ ตัวอย่าง:

```
FunTime  
area  
getName
```

กฎการตั้งชื่อ Identifier (ต่อ)

- สามารถใช้ตัวใหญ่และตัวเล็กผสมกันได้ แต่จะถือว่าเป็นคนละตัวกัน

- ตัวอย่าง: Identifiers 3 ตัวต่อไปนี้แตกต่างกัน

mainWindow

mAinWindow

MainWindow

- ไม่อนุญาตให้มีช่องว่างในชื่อ Identifier

- ตัวอย่าง: Identifiers 2 ตัวต่อไปนี้ ไม่ถูกต้อง

~~Sample Program~~

~~My First Application~~

ตัวอย่างคำสงวนในจาวา

abstract assert boolean break byte
case catch char class const
continue default do double else
enum 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

■ หมายเหตุ

- ❑ const และ goto เป็นคำสงวนที่ไม่มีที่ใช้ในภาษาจาวา
- ❑ true, false, null เป็น literal value

รูปแบบการตั้งชื่อของจาวา (Java Naming Convention)

- ตัวแรกของชื่อ Class ให้ขึ้นต้นด้วยตัวพิมพ์ใหญ่
- ตัวแรกของชื่อ object และ method ให้ขึ้นต้นด้วยตัวพิมพ์เล็ก
- ชื่อที่ประกอบขึ้นจากคำหลายคำให้ใช้ตัวพิมพ์ใหญ่ช่วยแยกคำ
- ชื่อของ class, object ควรเป็นคำนาม
- ชื่อของ method ควรเป็นคำกริยา
- ชื่อของค่าคงที่ (constant) ให้ใช้ตัวพิมพ์ใหญ่ทั้งหมด และใช้ _ เป็นตัวช่วยแยกคำ

รูปแบบการตั้งชื่อของจาวา (ต่อ)

■ ตัวอย่าง:

□ Class

MainWindow Customer

□ Object

mainWindow jack

□ Method

show () getName ()

□ Constant

HOST_ADDRESS PI

ใช้มาตรฐานรูปแบบการ
ตั้งชื่อ ในการเขียน
โปรแกรมจาวา เพื่อให้ง่าย
ในการอ่านและการ
บำรุงรักษา

ตัวแปร (Variable)

- ตัวแปร = ชื่อที่ใช้ในการเก็บข้อมูล
 - จอ่งเนื้อที่ในหน่วยความจำหลักเพื่อใช้ในการเก็บค่าตามชนิดของข้อมูล
 - ใช้เป็นชื่อในการอ้างถึงเนื้อที่ในหน่วยความจำหลักนั้น ๆ
- การตั้งชื่อตัวแปรเป็นไปตามกฎการตั้งชื่อของ Identifier
- Syntax: รูปแบบการประกาศตัวแปร

`<data type> <variable>;`

`<data type>` = ชนิดของตัวแปร (คลาสหรือชนิดข้อมูลพื้นฐาน)

`<variables>` = กลุ่มลำดับของตัวแปรแยกจากกันโดย “,”

□ ตัวอย่าง: `int positionX, positionY;`

ชนิดข้อมูลพื้นฐาน (Primitive Data Type)

■ จาวากำหนดชนิดตัวเลข ไว้ 6 ชนิด

`byte, short, int, long, float, double`

■ และชนิดพื้นฐานอื่นอีก 2 ชนิด

`char, boolean`

ชนิดข้อมูล	Content	ค่าปริยาย	ค่าต่ำสุด	ค่าสูงสุด	ขนาด
byte	Integer	0	-2^7	2^7-1	8 bits
short	Integer	0	-2^{15}	$2^{15}-1$	16 bits
int	Integer	0	-2^{31}	$2^{31}-1$	32 bits
long	Integer	0	-2^{63}	$2^{63}-1$	64 bits
float	Real	0.0	-3.40282347E+38	3.40282347E+38	32 bits
double	Real	0.0	-1.79769...E+308	1.79769...E+308	64 bits
char	character	'\u0000'			16 bits
boolean	boolean	false			1 bit

Literal ของเลขจำนวนเต็ม (byte, short, int, long)

■ สามารถกำหนดค่าตัวเลขได้ 4 รูปแบบ:

■ รูปเลขฐานสิบ (Decimal)

□ 12 เลขฐานสิบที่มีค่า สิบสอง

■ รูปเลขฐานแปด (Octal)

□ 062 เลขฐานแปดขึ้นต้นด้วยเลขศูนย์ แสดงค่า (62) ฐาน 8 = 50

■ รูปเลขฐานสิบหก (Hexadecimal)

□ 0xA2 เลขฐานสิบหกขึ้นต้นด้วยเลขศูนย์และ "X" แสดงค่า (A2) ฐาน 16 = 162

■ รูปเลขฐานสอง (Binary) (เริ่มใน version 7)

□ 0b11 เลขฐานสองขึ้นต้นด้วยเลขศูนย์และ "B" แสดงค่า (11) ฐาน 2 = 3

■ การกำหนดตัวเลขที่เป็นชนิด long ให้ใช้ตัวอักษร “L” หรือ “l” เติมท้าย (12L, 062L, 0xA2L)

Literal ของเลขจำนวนจริง

- กำหนดข้อมูลชนิดตัวเลขจำนวนจริงโดยใช้ “.” หรือใส่:
 - E หรือ e เพื่อแสดง exponential
 - $2.3E+2$ ซึ่งมีค่าเท่ากับ 230
 - F หรือ f เพื่อแสดง float
 - $3.14F$ เป็น float ซึ่งมีค่าเท่ากับ 3.14
 - D หรือ d เพื่อแสดง double
 - $1.65E3D$ เป็น double ซึ่งมีค่าเท่ากับ 1650
- เลขจำนวนจริงมีค่าปริยาย (default) เป็น double
- กำหนด float โดยใช้ตัวอักษร “F” หรือ “f” ($12.0f$, $12.0F$)

Underscores ใน Literal ตัวเลข

- จาवाตั้งแต่ version 7 ให้ใช้ “_” คั่นเพื่อแสดงค่าตัวเลขให้ชัดเจนได้
 - 3_100 มีค่าเท่ากับ 3100
 - long citizenID = 1_2345_67890_12_3L;
 - double pi = 3.14_15;
- โดยให้วาง “_” ไว้ระหว่างตัวเลขและไม่ให้ไว้
 - ที่ตำแหน่งแรกหรือสุดท้ายของตัวเลข (~~12~~, ~~13~~)
 - ข้างจุดทศนิยม (ไม่ว่าหน้าหรือหลังจุด) (~~3.14~~, ~~3.14~~)
 - ก่อนตัว D, F หรือ L suffix (~~4.4F~~, ~~44L~~)
 - ในตำแหน่งที่คาดหวังตัวเลข (~~0x45~~, ~~0x45~~) แต่ 0_45 (45ฐาน 8) ได้

คำสั่งประเภทการให้ค่า (Assignment)

- เป็นการกำหนดค่าให้กับตัวแปร

- Syntax:

<variable> = <expression>;

<variable> = ตัวแปร

<expression> = นิพจน์คณิตศาสตร์

- ตัวอย่าง:

`myIntNumber = y - 25;`

- หมายเหตุ การทำ assignment ต่างจาก การเปรียบเทียบ “เท่ากับ” ทางคณิตศาสตร์

~~`25 * 12 = x;`~~
~~`x + y = y + x;`~~

นิพจน์คณิตศาสตร์ (Arithmetic Expression)

■ นิพจน์คณิตศาสตร์ ประกอบด้วย

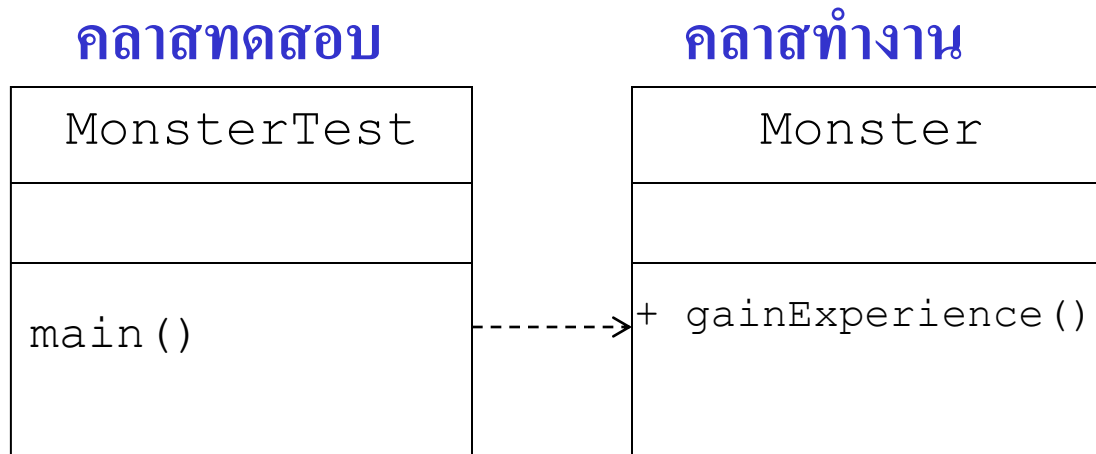
- ตัวแปร (เช่น x, y, z) หรือ ค่าตัวเลข (เช่น $1, 2$) ที่เรียกว่า Operand
- สัญลักษณ์ที่ใช้ในการคำนวณ เรียกว่า Operator (เช่น $+, -, *, /, \%$)
- ตัวอย่าง

$$x + y / z$$

ตัวอย่างนิพจน์คณิตศาสตร์

Operation	Java Operator	ตัวอย่าง	Value (x=10, y=7, z=2.5)
บวก (Addition)	+	$x + y$	17
ลบ (Subtraction)	-	$x - y$	3
คูณ (Multiplication)	*	$x * y$	70
หาร (Division)	/	x / y	1
		x / z	4.0
เศษเหลือ (Modulo division)	%	$x \% y$	3

MonsterTest



```
public class MonsterTest {  
    public static void main(String args[]) {  
        Monster m = new Monster();  
        m.gainExperience();  
    }  
}
```


การ implement Class อย่างง่าย

```
// Comment  
  
import statement  
  
public class ClassName  
{  
    <access modifier> instance variables  
  
    method block  
  
    . . .  
  
    method block  
  
}
```

Monster.java

```
public class Monster{
```

ตัวแปรวัตถุ

ลักษณะ

```
private boolean isSleep;  
private int positionX, positionY;  
private double eyeSight;  
private long expPoint;
```

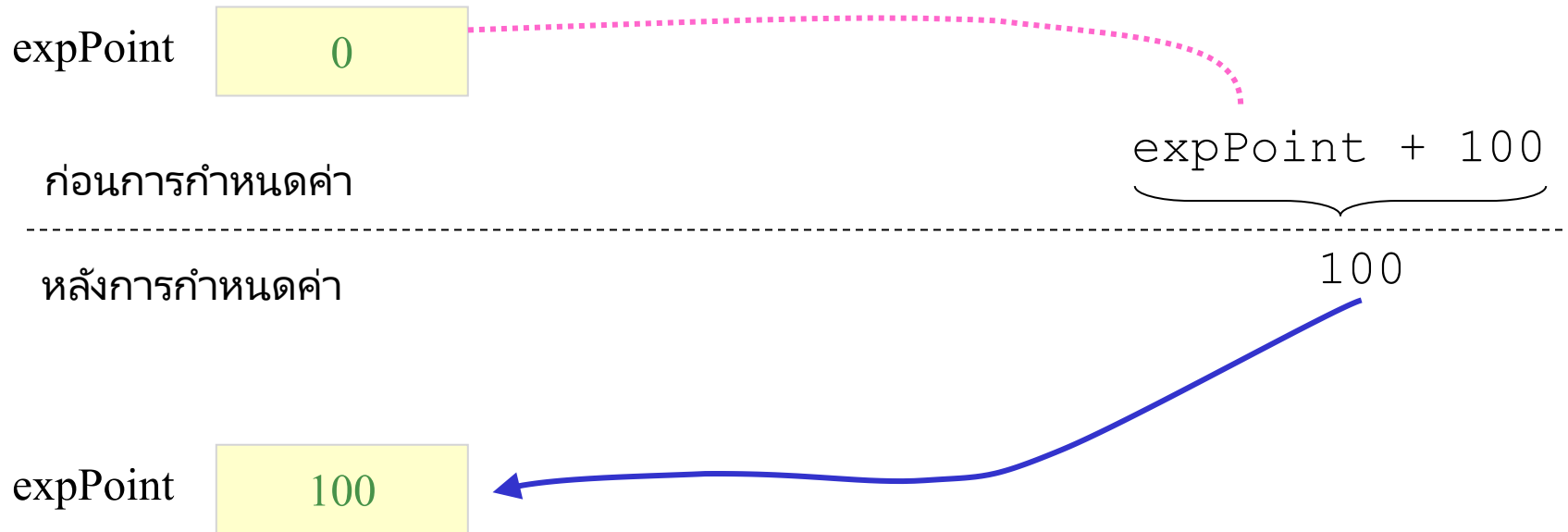
```
public void gainExperience() {  
    expPoint= expPoint + 100;  
    eyeSight = 5.0 + expPoint/1000.0;  
}
```

พฤติกรรม หรือ
ความสามารถ

```
}
```

การทำงานของ Assignment

`expPoint = expPoint + 100`



ลำดับของ operator (Precedence Rules)

Order	Group	Operator	กฎ
High ↓ Low	subexpression	()	ทำเป็นอันดับแรก จากในสุด ออกมาข้างนอก
	unary operator	−, +	ทำจาก ขวา ไป ซ้าย
	multiplicative op.	*, /, %	ทำจาก ซ้าย ไป ขวา
	additive operator	+, −	ทำจาก ซ้าย ไป ขวา

a * (b + − (c / d) / e) * (f − g % h)

The diagram illustrates the evaluation order of the expression $a * (b + -(c / d) / e) * (f - g \% h)$ based on operator precedence. The operations are numbered 1 through 8, indicating the sequence from highest to lowest precedence:

- 1: c / d (multiplicative)
- 2: $-(c / d)$ (unary)
- 3: $b + -(c / d)$ (additive)
- 4: $(b + -(c / d)) / e$ (multiplicative)
- 5: $g \% h$ (multiplicative)
- 6: $f - g \% h$ (additive)
- 7: $a * ((b + -(c / d)) / e)$ (multiplicative)
- 8: $a * ((b + -(c / d)) / e) * (f - g \% h)$ (multiplicative)

การแปลงชนิด (Casting Conversion)

- การเปลี่ยนชนิดของข้อมูล (data type) เมื่อทำ arithmetic operation ของชนิดข้อมูลต่างกัน
- Casting Conversion เป็นในลักษณะ
 - implicit หรือ numeric promotion เป็นการแปลงโดยอัตโนมัติ
 - explicit เป็นการแปลงโดยการใช้ type cast operator
 - ตัวอย่าง: ถ้า x เป็น short

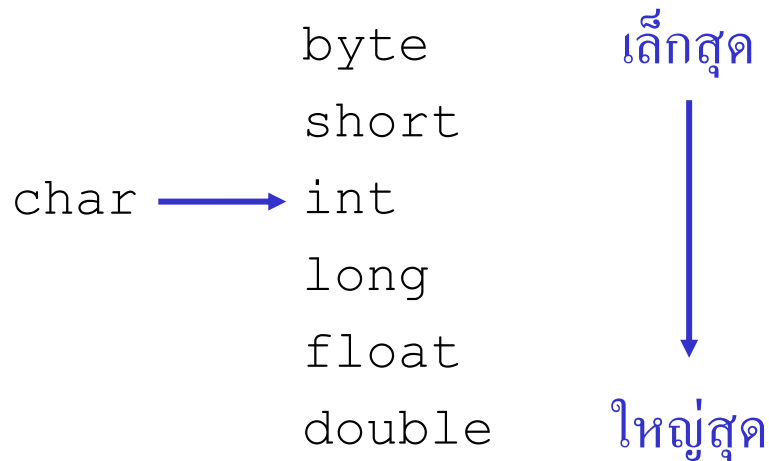
```
x + 3           // implicit casting to int
(float) x / 3    // explicit casting to float
```

การแปลงชนิดโดยนัย (Implicit Arithmetic Promotion)

- Unary Operator (เครื่องหมายที่มี operand ตัวเดียว)
 - ผลลัพธ์เป็นชนิดเดียวกับ operand
 - ยกเว้น กรณี operand ชนิด byte หรือ short จะได้ int
- Binary Operator
 - ถ้า operand ตัวใดตัวหนึ่งเป็นชนิด double, ผลลัพธ์เป็นชนิด double
 - ถ้า operand ตัวใดตัวหนึ่งเป็นชนิด float, ผลลัพธ์เป็นชนิด float
 - ถ้า operand ตัวใดตัวหนึ่งเป็นชนิด long, ผลลัพธ์เป็นชนิด long
 - ถ้าไม่เช่นนั้น ให้ผลลัพธ์เป็นชนิด int

Assignment Conversion

- การ assign arithmetic expression ให้กับตัวแปรที่มีชนิดต่างกัน จะทำให้เกิด implicit conversion เช่นกัน
 - การแปลงทำโดยอัตโนมัติ ถ้าแปลงแล้วชนิดข้อมูลเป็นชนิดที่มีค่าขนาดใหญ่ขึ้น



ตัวอย่าง implicit conversion

```
int j;  
long k;  
j = 88;  
k = j; // valid upward cast
```

```
k = 88L;  
j = k; // invalid cast
```

```
short a, b, c;  
a = 1;  
b = 2;  
c = a + b; // invalid cast
```

a + b ผลลัพธ์เป็น int

Declaration & Assignment พร้อมกัน

```
double y = 1.24F; // valid  
float x = 1.25; // invalid
```

```
int smallVal = 99L;  
//invalid  
long bigVal = 6; // legal
```

default type เป็น double

Type Cast Operator

- ทำให้ทั้งที่การแปลงเป็นชนิดที่มีค่าขนาดใหญ่ขึ้นหรือเล็กลง
- Syntax: **(<data type> <expression>**
- type cast เป็น unary operator ที่มี precedence เหนือกว่า binary operator อื่น ๆ

```
short a = 1, b = 2;  
short c;  
c = a + b; // invalid cast  
c = (short) (a+b); // ok
```

```
int c;  
long bigVal = 6; // legal  
c = bigVal; // invalid  
c = (int)bigVal; // valid
```

แบบฝึกหัด

พิจารณานิพจน์ (expression) ต่อไปนี้ อันใดที่ถูกต้อง (valid)

```
int    i = 1,    j = 2;  
float  x = 4.0f, y = 2.0f;  
double u = 5.0,  v = 3.0;
```

```
i = x;  
x = u + y;  
x = 23.4 + j * y;  
v = (int) x;  
y = j / i * x;
```

แบบฝึกหัด (ต่อ)

จากนิพจน์คณิตศาสตร์ที่ให้ หลังการให้ค่าตัวแปรด้านซ้ายมีค่าเท่าไร

```
int    i = 1,    j = 2;  
float  x = 4.0f, y = 2.0f;  
double u = 5.0,  v = 3.0;
```

```
i = (int) x;  
x = (float) u + y;  
x = 23.4F + j * y;  
v = (int) x;  
y = i / j * x;
```

ชนิด char

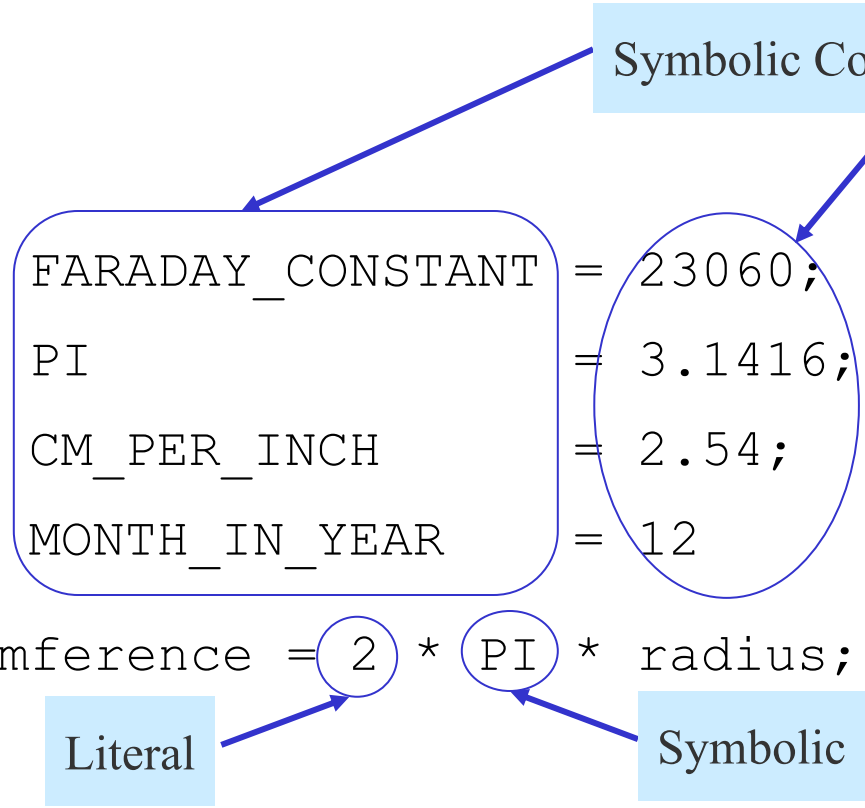
- จาว่าใช้ unicode character: 1 อักษรใช้ 2 bytes
- ค่า literal ของตัวอักษรคร่อมด้วย ' (single quote)
 - 'a', 'b', 'c'
- สามารถให้ค่า char กับตัวแปรชนิด int
 - `int x = 'a';` // ให้ค่า 97 กับตัวแปร x
- สามารถให้ค่าตัวเลขจำนวนเต็ม (integer literal) กับตัวแปรชนิด char ได้
 - `char c = 97;` // ให้ค่า 'a' กับตัวแปร c
- สามารถทำ arithmetic +, - บนตัวแปรชนิด char ได้
 - `char ch = 'a';`
 - `int x = ch + 1;` // ให้ค่า 98 กับตัวแปร x

ค่าคงที่ (Constant)

- การประกาศค่าคงที่ที่ทำได้เช่นเดียวกับการประกาศ variable แต่เพิ่ม modifier “final”

- ตัวอย่างเช่น

```
final short FARADAY_CONSTANT = 23060; // cal/volt
final double PI = 3.1416;
final double CM_PER_INCH = 2.54;
final int MONTH_IN_YEAR = 12
```



```
double circumference = 2 * PI * radius;
```

Monster.java (Revisited)

```
public class Monster {  
    private final double BASE_EYE_SIGHT = 5.0;  
    private boolean isSleep;  
    private int positionX, positionY;  
    private double eyesight;  
    private long expPoint;  
  
    public void gainExperience() {  
        expPoint= expPoint + 100;  
        eyeSight = BASE_EYE_SIGHT + expPoint/1000.0;  
    }  
}
```

Math class (รายละเอียดใน Java Lang. Spec.)

- Math class เป็น class ใน java.lang package โดยจะมีฟังก์ชันทางคณิตศาสตร์ (mathematical functions) ต่าง ๆ

□ ตัวอย่าง เพื่อคำนวณ

$$\frac{1}{2} \sin\left(x - \frac{\pi}{\sqrt{y}}\right)$$

`(1.0/2.0) * Math.sin(x - Math.PI / Math.sqrt(y))`

- Syntax: การเรียกใช้

Math.<method name>

Math.<class constant>

ตัวอย่าง methods ใน Math class

Methods	
Modifier and Type	Method and Description
static double	<code>abs(double a)</code> Returns the absolute value of a double value.
static float	<code>abs(float a)</code> Returns the absolute value of a float value.
static int	<code>abs(int a)</code> Returns the absolute value of an int value.
static long	<code>abs(long a)</code> Returns the absolute value of a long value.
static double	<code>acos(double a)</code> Returns the arc cosine of a value; the returned angle is in the range 0.0 through π .
static double	<code>asin(double a)</code> Returns the arc sine of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$.
static double	<code>atan(double a)</code> Returns the arc tangent of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$.
static double	<code>atan2(double y, double x)</code> Returns the angle θ from the conversion of rectangular coordinates (x, y) to polar coordinates (r, θ).
static double	<code>cbrt(double a)</code> Returns the cube root of a double value.
static double	<code>ceil(double a)</code> Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
static double	<code>copySign(double magnitude, double sign)</code> Returns the first floating-point argument with the sign of the second floating-point argument.
static float	<code>copySign(float magnitude, float sign)</code> Returns the first floating-point argument with the sign of the second floating-point argument.
static double	<code>cos(double a)</code> Returns the trigonometric cosine of an angle.

แบบฝึกหัด

■ กำหนดให้

```
final int X = 9;
```

```
double y;
```

■ พิจารณาว่า statement ต่อไปนี้ผิดที่ใด

```
y = (1/2) * Math.sqrt(X);
```

```
y = sqrt(38.0);
```

```
y = Math.exp(2, 3);
```

```
y = math.sqrt(36);
```

```
y = Math.sin(360);
```

Roundoff Errors

```
double f = 4.35; // keep 4.349999999999999994
int n = (int) (100*f);
System.out.println(n); // print 434
```

- เนื่องจากการแสดงค่า double ในคอมพิวเตอร์นั้น เป็นแบบ binary ไม่ได้เก็บค่าจริงของ 4.35 แต่เก็บค่าที่ใกล้เคียงที่สุด เมื่อแปลงชนิดอาจทำให้ค่าผิดพลาดจากการตัดค่าส่วนหลังทิ้งได้

```
int n = (int) Math.round(100*f);
System.out.println(n); // print 435
```

String

- String เป็นคลาสในจาวา java.lang เพื่อแทนลำดับของตัวอักษร (≥ 0 ตัวอักษร)
- ค่าคงที่ (String literal) ครอบด้วยเครื่องหมาย " (double quote)
 - ถือเป็นวัตถุ
- null หรือ empty strings ไม่มีตัวอักษร (มีค่าไม่เท่ากัน)
- String ตัวเลข (Numeric string) ประกอบด้วยตัวเลขในเครื่องหมาย ""
- ความยาวของ String = จำนวนตัวอักษรใน String นั้น
 - ถ้ามได้จากเมทอด length() ของคลาส String

Non-zero value



null



0



undefined



การทำงานกับ String

■ การประกาศตัวแปร String

```
String name = "NidNoi";  
String name2 = new String("NidNoi");
```

■ การกำหนดค่าให้กับ String

```
name = "Nim";
```

■ การหาความยาวของ String โดย method length()

```
int i = name.length();  
i = "Nim".length();
```

■ การต่อ String (Concatenation)

```
name = "Ms." + name;  
name = "Ms." + name + "name length = " + i;
```

Scanner class (JDK 5.0)

ใน java.util package

Scanner

- + Scanner(InputStream)
- + nextInt() : int
- + nextDouble() : double
- + nextLine() : String

Sample Methods

Scanner (InputStream)

สร้างวัตถุ Scanner ที่อ่านข้อมูลจาก source stream (ในที่นี้เราจะใช้ System.in เป็น source)

int

nextInt()

อ่านข้อมูล int จากหน้าจอ console จากผู้ใช้
คืนค่า int กลับไปยังผู้เรียก

double

nextDouble()

อ่านข้อมูล double จากหน้าจอ console จากผู้ใช้
คืนค่า double กลับไปยังผู้เรียก

String

nextLine()

อ่านข้อมูลสายอักขระ 1 บรรทัดจากหน้าจอ console จากผู้ใช้
คืนค่า String กลับไปยังผู้เรียก

ScannerTest.java

```
import java.util.Scanner;

public class ScannerTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        System.out.println("You typed: " + scanner.nextInt());

        System.out.print("Enter a double: ");
        System.out.println("You typed: " + scanner.nextDouble());
        scanner.nextLine();

        System.out.print("Enter a line of text: ");
        System.out.println("You typed: " + scanner.nextLine());
    }
}
```

PrintStream class (JDK 5.0)

PrintStream

- + printf(String format, Object... o): void
- + print(String s): void
- + println(String s): void
- + ...

Sample Methods

void

printf(String format, Object... o)

พิมพ์ข้อมูลใน list ของ Object o ออกทาง output stream โดยใช้รูปแบบที่กำหนดใน format (ในที่นี้ใช้ System.out → console) (JDK 5.0)

void

print(String s)

พิมพ์ข้อมูล s ออกทาง output stream (JDK 1.0)

void

println(String s)

พิมพ์ข้อมูล s ออกทาง output stream แล้วขึ้นบรรทัดใหม่ (JDK 1.0)

รูปแบบของชนิด (บางส่วน) สำหรับ printf

Code	Type	ตัวอย่าง
d	เลขจำนวนเต็ม	213
x	เลขฐาน 16	D5
o	เลขฐาน 8	325
f	เลขจุดลอยตัวที่มีจำนวนหลักและทศนิยมแน่นอน	21.30
e	เลขจุดลอยตัวแบบชี้กำลัง	2.13e+1
g	เลขจำนวนจุดลอยตัวทั่วไป	21.3
s	ข้อความ	Monthly Payment
n	สตริง (ขึ้นกับระบบ)	

รูปแบบของ Flag (บางส่วน) สำหรับ printf

Flag	ความหมาย	ตัวอย่าง
-	จัดตำแหน่งชิดซ้าย	213
0	แสดงศูนย์นำหน้า	000213
+	แสดงเครื่องหมาย + นำหน้า	+213
(ใส่วงเล็บครอบเลขลบ	(21.30)
,	แสดงตัวคั่น , แบบเลขทางบัญชี	21,300
^	แปลงให้เป็นตัวพิมพ์ใหญ่	2.13E+1

PrintfTest.java

```
public class PrintfTest {  
    public static void main(String[] args) {  
        System.out.printf("%10.2f %n ", 1235.12131);  
        System.out.printf("%-10.2f %n ", 1235.12131);  
        System.out.printf("%s %d %<x %c %n",  
"You have ", (int)1235.12131, '\u0007');  
    }  
}
```

การ implement Class อย่างง่าย

```
// Comment
```

```
import statement
```

```
public class ClassName
```

```
{
```

```
instance variables
```

```
Constructors
```

```
Methods
```

```
}
```

ลักษณะ

ความสามารถ

การ implement Class อย่างง่าย

Monster	
-positionX: int	
-positionY: int	
+moveTo(int, int): void	
+getPositionX(): int	

← คลาส

← ลักษณะ

← ความสามารถ

```
public class Monster {  
    private int positionX;  
    private int positionY;  
  
    public void moveTo(int x, int y) {  
        positionX = x;  
        positionY = y;  
    }  
    public int getPositionX(){  
        return positionX;  
    }  
}
```

การกำหนด เมทอด

```
<modifiers> <return type> <method name> (<parameter list>)  
{  
    <statement list>  
}
```

modifier	return type	method name	parameter
public	void	moveTo	(int x, in y)
	...		
public	int	getPositionX	()

```
public void moveTo (int x, in y) {  
    ...  
}  
public int getPositionX () {  
    return positionX;  
}
```

เมทอด (Methods)

- **วัตถุประสงค์** เพื่อรวมกลุ่มของคำสั่ง (statements) ของการทำหน้าที่อย่างใดอย่างหนึ่งไว้ด้วยกัน
- **ประโยชน์**
 - ช่วยจัดระเบียบของโค้ดที่มีตรรกะในเรื่องเดียวกันไว้ด้วยกัน
 - ทำให้สามารถเรียกใช้หน้าที่เดิมซ้ำได้อีก
 - ช่อนรายละเอียดและความซับซ้อนของการทำงานของเมทอดนั้น

```
public static void main(String[] args) {  
    Monster mon = new Monster();  
  
    mon.moveTo(4,3);  
}
```

เรียกเมทอด

คืนผลลัพธ์ void

Execute Code

```
public void moveTo(int x, int y) {  
    positionX = x;  
    positionY = y;  
}
```

พารามิเตอร์ (Parameter)

- เมทอดอาจรับหรือไม่รับเข้าพารามิเตอร์ (ศูนย์หรือมากกว่า)
- พารามิเตอร์
 - เป็นข้อมูลเข้าเพื่อใช้ในการทำงานของเมทอด
 - จัดว่าเป็นตัวแปรท้องถิ่น (local variable) ในเมทอด

```
int positionX = 4;  
int positionY = 3;  
mon.moveTo(positionX, positionY);
```

positionX และ *positionY*
Actual Argument
ค่าที่ส่งให้เมทอด

4, 3

```
public void moveTo(int x, int y) {  
    positionX = x;  
    positionY = y;  
}
```

x และ *y*
Formal Parameter
ตัวแปรรับค่าจากตัวเรียก
รู้จักเพียงภายในเมทอด

instance variable

การคืนค่า (Return Value)

- เมท็อดเมื่อประมวลผลเสร็จอาจ
 - ไม่คืนค่า ประกาศชนิดที่คืนเป็น **void**
 - คืนหนึ่งค่าผ่านประโยค **return** ชนิดที่คืนต้องสอดคล้องกับค่าที่คืน
- ค่าที่ได้รับคืนแทนประโยคการเรียกเมท็อด

```
public static void main(String[] args) {  
    int monY = 3;  
    Monster mon = new Monster();  
    mon.moveTo(4, monY);  
    int x = mon.getPositionX();  
}
```

เรียกเมท็อด

คืนผลลัพธ์

Return value

Execute Code

```
public int getPositionX() {  
    return positionX;  
}
```


ตัวแปรท้องถิ่น (Local Variables)

■ ตัวแปรท้องถิ่น (Local Variables)

- ❑ ตัวแปรที่ประกาศอยู่ในเมทอด
- ❑ มีขอบเขตที่ใช้งานได้เพียงภายในเมทอด
- ❑ จาบบังคับต้องให้ค่าเริ่มต้นก่อนใช้งาน (เช่น ก่อนพิมพ์ หรืออ่านค่า)

```
public class Monnster {  
    private int positionX;  
    private int positionY;  
    . . .  
    public double getDistanceFrom(int posX, int posY){  
        int xDiff = posX - getPositionX();  
        int yDiff = posY - getPositionY();  
        double distance = Math.sqrt(Math.pow(xDiff,2)+Math.pow(yDiff,2));  
        return Math.abs(distance);  
    }  
    public int getDistance(){  
        return distance; // error  
    }  
}
```

ขอบเขตของตัวแปร (1)

- ไม่สามารถประกาศตัวแปรระดับเดียวกัน ที่มีชื่อเดียวกัน ในขอบเขตเดียวกัน
- *ตัวแปรท้องถิ่น (local)* กับ *ตัวแปรพารามิเตอร์* จัดเป็นระดับเดียวกัน
 - มีขอบเขตและใช้งานได้ไม่เกินภายในเมทอดเท่านั้น
- *ตัวแปรวัตถุ (instance variable)* มีขอบเขตและใช้งานได้ทั้งคลาส
- ประกาศตัวแปรชื่อเดียวกัน อยู่ได้ขอบเขตต่างกันได้

```
public class Monster {  
    private final double BASE_EYE_SIGHT = 5.0;  
    private Boolean isSleep;  
    . . .  
    private long expPoint; // instance variable  
  
    public void gainExperience(long expPoint) {  
        expPoint = expPoint;    // -- incorrect operation  
        eyesight = BASE_EYE_SIGHT = expPoint/1000.0;  
    }  
  
}
```

ขอบเขตของตัวแปร (2)

- ประกาศตัวแปรท้องถิ่นชื่อเดียวกัน ภายใต้ขอบเขตอันเดียวกันไม่ได้

```
public class Monster {  
    private final double BASE_EYE_SIGHT = 5.0; //instance variable  
  
    public void calculateEyeSight(double factor) {  
        double factor = expPoint/1000; // violate! duplicate variable  
        eyeSight = BASE_EYE_SIGHT + factor;  
    }  
}
```

สรุปในบทนี้

- เรียนรู้พื้นฐานองค์ประกอบของโปรแกรม:
 - Identifiers, ชนิดข้อมูลพื้นฐาน (Primitive), literal, เครื่องหมายการคำนวณ
- เข้าใจการใช้ข้อมูลชนิดตัวเลขและนิพจน์คณิตศาสตร์
- วิเคราะห์นิพจน์คณิตศาสตร์ร่วมกับกฎลำดับ (Precedence rules)
- อธิบายการจัดสรร หน่วยความจำสำหรับค่าข้อมูลพื้นฐาน
- สามารถกำหนดค่าเริ่มต้นของตัวแปรและค่าคงที่ได้
- เขียนนิพจน์คณิตศาสตร์โดยใช้ Methods ใน Math class
- เรียนรู้เกี่ยวกับข้อมูลตัวอักษรและสายอักขระ
- เขียนโปรแกรมสำหรับการนำเข้าและส่งออกข้อมูล
- เข้าใจพื้นฐานการเขียนเมทอดในคลาส
 - เมทอดรวมกลุ่มของคำสั่งเพื่อทำหน้าที่อย่างใดอย่างหนึ่งของวัตถุ
 - รับค่าเข้าผ่านพารามิเตอร์ และคืนค่าผ่าน return
 - ขอบเขตของพารามิเตอร์และตัวแปรที่ประกาศในเมทอด