Decisions

การทำงานแบบทางเลือก

This chapter

- If statements: if, if-else, nested if
- ตัวกระทำแบบเงื่อนไข (conditional operator)
- การเปรียบเทียบจำนวน สตริง และ วัตถุ
- Dangling else
- Short-circuit Evaluation
- Switch statement
- ทดสอบโปรแกรม (Testing)

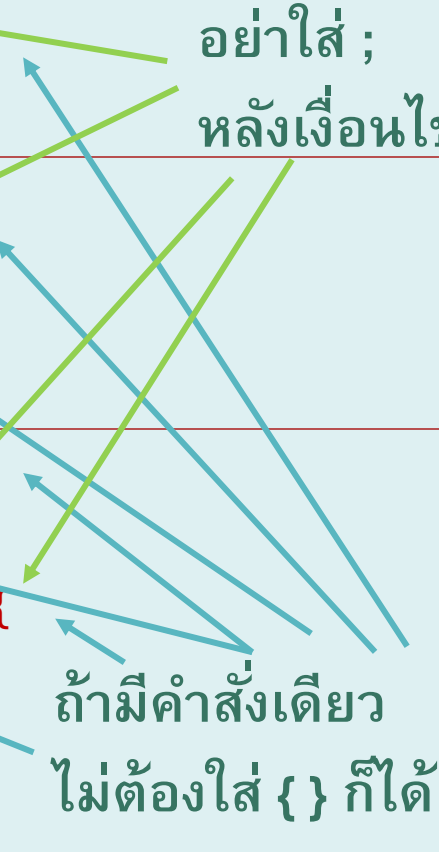
If Structures ในไพธอนและจาวา

Python	Java

`if cond :`
`  ...`
	`if ( cond ) {`
	`  ...`
	`}`

`if cond :`
`  ...`
`else:`
`  ...`
	`if (cond ) {`
	`  ...`
	`} else {`
	`  ...`
	`}`

`if cond1 :`
`  ...`
`elif cond2:`
`  ...`
`else:`
`  ...`
	`if (cond1 ) {`
	`  ...`
	`} else {`
	`  if (cond2 ) {`
	`  ...`
	`  } else {`
	`  ...`
	`  }`
	`}`

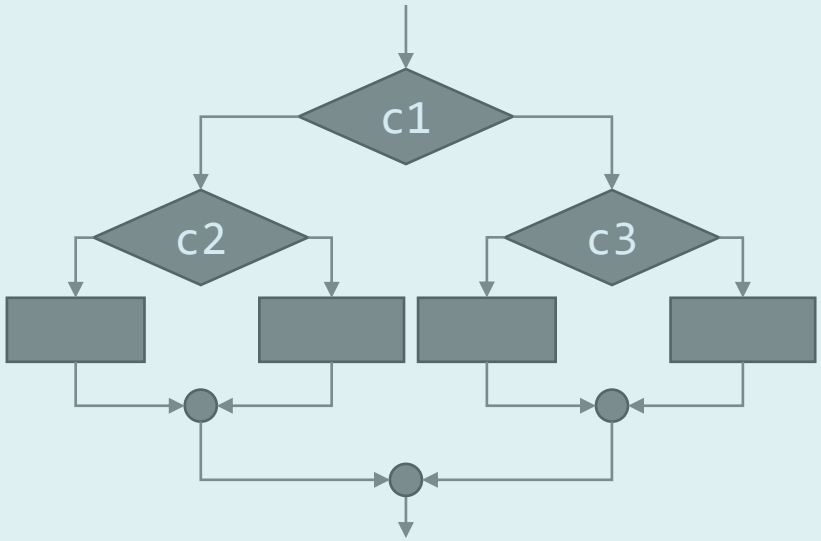


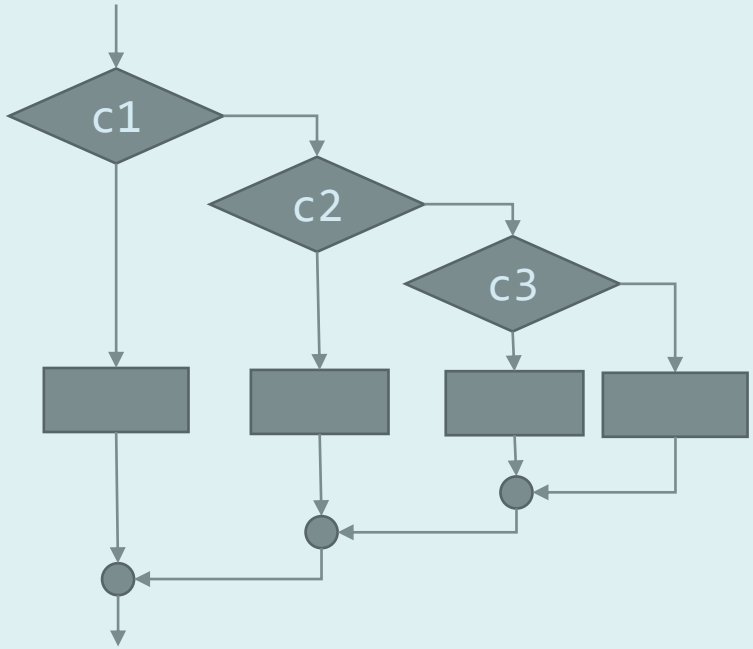
Examples

```
if (amount <= balance)
   balance = balance - amount;
else
   balance = balance - OVERDRAFT_PENALTY;

if (amount <= balance) {
   new_balance = balance - amount;
   balance = new_balance;
}
else {
   new_balance = balance - OVERDRAFT_PENALTY;
   balance = new_balance;
}
```

Nested If v.s. if-else-if ladder






```
if (c1) {
   if (c2) { ... }
   else { ... }
} else {
  if (c3) { ... }
  else { ... }
}
```

```
if (c1) {
   if (c2) { ... }
   else {
      if (c3) { ... }
      else { ... }
   }
}
```

ตัวกระทำแบบเงื่อนไข (conditional operator)

conditional operator

condition ? value1 : value2

- คืน *value1* ถ้า *condition* เป็นจริง
- คืน *value2* ถ้า *condition* เป็นเท็จ

ตัวอย่าง

```
int x,y,d,max;

d   = x>y ? x-y : y-x;  // absolute value of (x-y)
max = x>y ? x : y;      // maximum of x and y
```

conditional operator

```
d = x>y ? x-y : y-x;
```

เหมือนกับ

```
if (x>y)
   d = x-y;
else:
   d = y-x;
```

ไม่ใช่

```
d=if (x>y) x-y; else y-x;
```

```
max = x>y ? x : y;
```

เหมือนกับ

```
if (x>y)
   max = x;
else:
   max = y;
```

ไม่ใช่

```
max=if (x>y) x; else y;
```

การเปรียบเทียบจำนวน สตริง และ วัตถุ

Relational Operators

Operators	Description
>	มากกว่า
>=	มากกว่าหรือเท่ากับ
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากับ
==	เท่ากับ
!=	ไม่เท่ากับ

ใช้กับ primitive data types
ไม่ใช้กับ objects

ลำดับความสำคัญ (precedence)

- Relational operators มีลำดับความสำคัญน้อยกว่า Arithmetic operators
  - ทำ Arithmetic operators ก่อน Relational operators
- Relational operators มีลำดับความสำคัญมากกว่า logical operators
  - ทำ relational operators ก่อน logical operators

Logical or Boolean Operators

Operators	Description
&&	logical and
||	Logical or
!	Logical not

ลำดับความสำคัญ (precedence)

- ! มีลำดับความสำคัญมากกว่า &&
- && มีลำดับความสำคัญมากกว่า ||

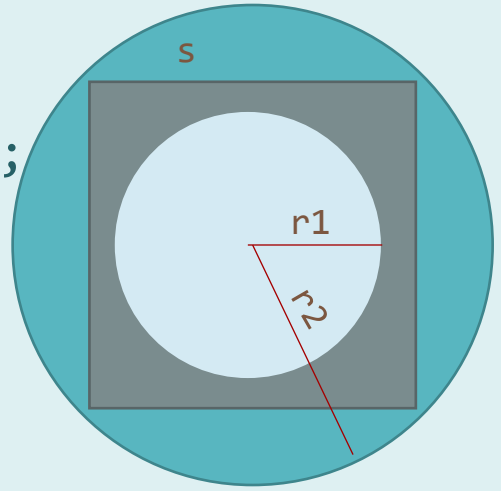
Examples

```
final double PI = 3.14159;
double r = 2.1, s = 3.46;

if (PI*r*r <= s*s)
   System.out.println("The square is bigger.");
else
   System.out.println("The circle is bigger.");
```

```
final double PI = 3.14159;
double r1 = 2.1, r2 = 4.12, s = 3.46;

if (2*r1<=s && s<=r2/Math.sqrt(2)) {
   System.out.println("small circle in square.");
   System.out.println("square in big circle.");
}
else
   System.out.println("cannot stack");
```



Examples

```
final double PI = 3.14159;
double r = 2.1, s = 3.46;

if (PI*r*r <= s*s) {
   System.out.println("The square is bigger."); }
else {
   System.out.println("The circle is bigger."); }
```

ใส่ { }ได้

```
final double PI = 3.14159;
double r1 = 2.1, r2 = 4.12, s = 3.46;

if (2*r1<=s && s<=r2/Math.sqrt(2)) {
   System.out.println("small circle in square.");
   System.out.println("square in big circle.");
}
else
   System.out.println("cannot stack");
```

เขียน
2*r1<=s<=r2/Math.sqrt(2)
ไม่ได้

Boolean variables

- กำหนดตัวแปรชนิด Boolean เพื่อใช้ในเงื่อนไขได้

```
final double PI = 3.14159;
double r1 = 2.1, r2 = 4.12, s = 3.46;
boolean c1insq, sqinc2;

c1insq = 2*r1<=s; sqinc2 = s<=r2/Math.sqrt(2);

if (c1insq && sqinc2) {
   System.out.println("small circle in square.");
   System.out.println("square in big circle.");
}
else
   System.out.println("cannot stack");
```

ระวัง

Roundoff Error

```
Math.sqrt(2)*Math.sqrt(2)
>> 2.0000000000000004

Math.pow(2,54)
>> 1.8014398509481984E16

Math.pow(2,54)-1
>> 1.8014398509481984E16
```

ไม่เทียบว่า floating-point numbers 2 ค่า **เท่ากัน** หรือไม่

Example

```
double x=2.0;

if (Math.pow(Math.sqrt(x),2)==2.0)    // (√2)² = 2 ?
   System.out.println("x is 2");
else
   System.out.println("x is not 2");


Output:
x is not 2
```

Comparing floating-point numbers

Equality test for two floating-point numbers x and y

- $|x - y| \leq \varepsilon$

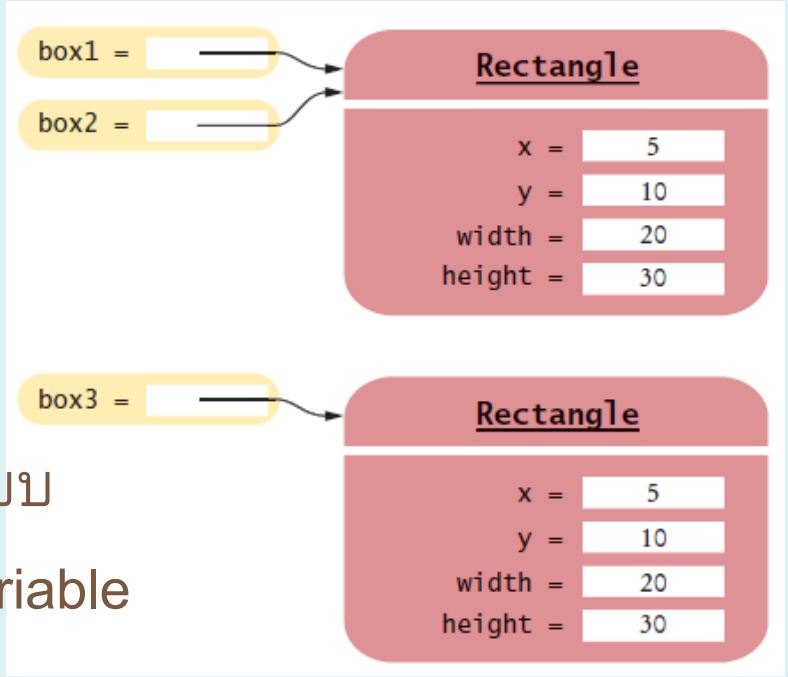
```
double x=2.0;
final double EPSILON = 1E-14;

if (Math.abs(Math.pow(Math.sqrt(x),2)-2.0)<=EPSILON )
   System.out.println("x is 2");
else
   System.out.println("x is not 2");
```

```
Output:
x is 2
```

Comparing objects

- สตริงเป็นวัตถุ (object)
- reference variable ของวัตถุเก็บ object reference ที่ชี้ไปที่วัตถุ
- ถ้าใช้ relational operator เปรียบเทียบ reference variable จะเป็นการเปรียบเทียบ object reference ที่เก็บใน reference variable
- ในตัวอย่างนี้
  - box1 เท่ากับ box2 เพราะอ้างถึง object เดียวกัน
  - แต่ box1 ไม่เท่ากับ box3 ถึงแม้ instance variable ทุกตัวมีค่าเท่ากัน



Example


```
if ("test"=="test")
   System.out.println("\"test\"==\"test\"");
else
   System.out.println("\"test\"!=\"test\"");
                                    Output:  "test"=="test"
```


```
String s = "test";
if (s=="test")
   System.out.println("s==\"test\"");
else
   System.out.println("s!=\"test\"");
                                    Output:  s=="test"
```


```
String s = "test";
if (s=="testimony".substring(0, 4))
   System.out.println("s==substring of \"testimony\"");
else
   System.out.println("s!=substring of \"testimony\"");
                          Output:  s!=substring of "testimony"
```

เมทอด equals สำหรับสตริง

```
if ("test".equals("test"))
   System.out.println("\"test\"==\"test\"");
else
   System.out.println("\"test\"!=\"test\"");
                          Output:  "test"=="test"
```

```
String s = "test";
if (s.equals("test"))
   System.out.println("s==\"test\"");
else
   System.out.println("s!=\"test\"");
                          Output:  s=="test"
```

```
String s = "test";
if (s.equals("testimony".substring(0, 4)))
   System.out.println("s==substring of \"testimony\"");
else
   System.out.println("s!=substring of \"testimony\"");
                           Output:  s==substring of "testimony"
```

เมทอดสำหรับเปรียบเทียบสตริงจากคลาส String

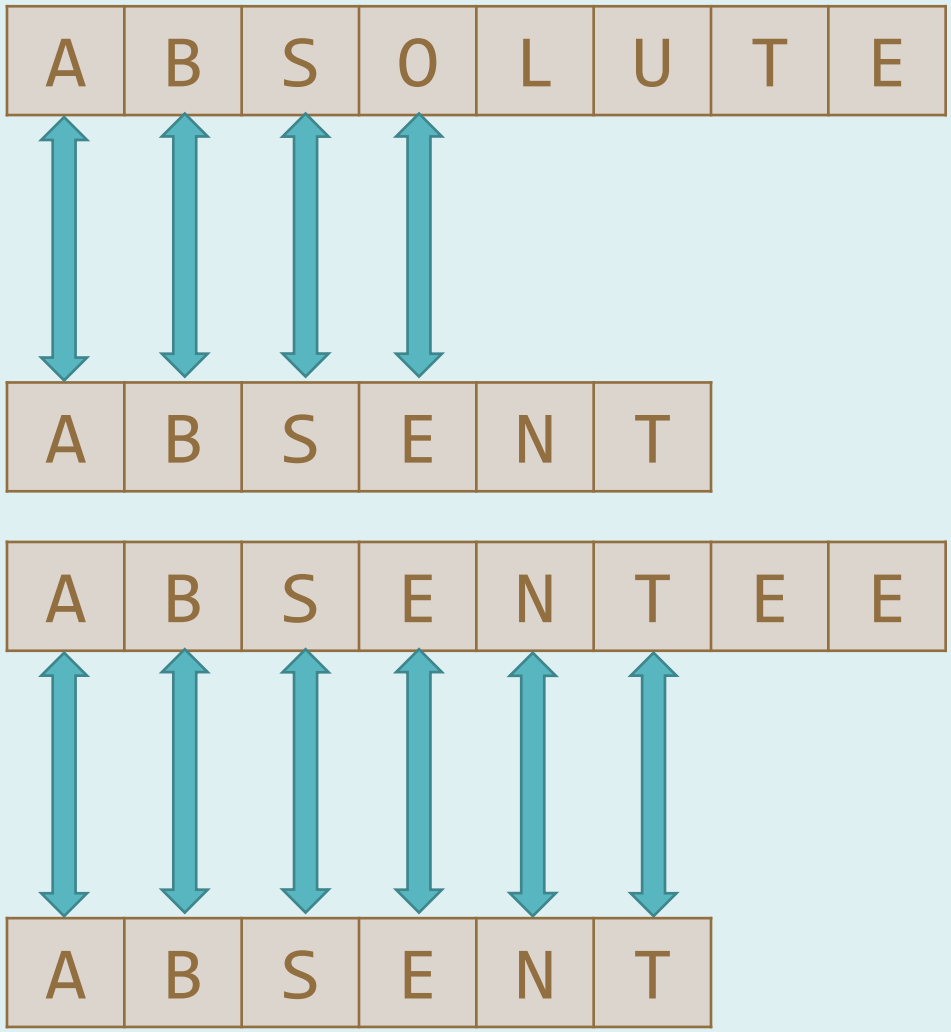
เมทอด	ตรวจสอบว่า
s1.equals(s2) return boolean	ตรวจสอบว่า s1 เก็บสตริงที่เท่ากับ s2
s1.equalsIgnoreCase(s2) return boolean	ตรวจสอบว่า s1 เก็บสตริงที่เท่ากับ s2 เมื่อถือว่าตัวอักขระใหญ่เท่ากับตัวอักขระเล็ก
s1.compareTo(s2) return int	ตรวจสอบว่า s1 กับ s2 เมื่อเทียบตัวอักขระไล่ไปทีละตัว โดยคืนค่าดังนี้ ค่า + ถ้า s1 อยู่ในลำดับหลังจาก s2 ค่า - ถ้า s1 อยู่ในลำดับก่อน s2 ค่า 0 ถ้า s1 = s2

เมทอด compareTo สำหรับสตริง

```
public int compareTo(String anotherString) {
    int len1 = value.length;
    int len2 = anotherString.value.length;
    int lim = Math.min(len1, len2);
    char v1[] = value;
    char v2[] = anotherString.value;

    int k = 0;
    while (k < lim) {
        char c1 = v1[k];
        char c2 = v2[k];
        if (c1 != c2)   return c1 - c2;
        k++;
    }
    return len1 - len2;
}
```

เมทอด compareTo สำหรับสตริง



Comparing Objects

- สร้างเมทอด equals สำหรับแต่ละคลาส

```
public class Rectangle {
   private int x, y, width, height;
   public Rectangle(int x, int y, int w, int h) {
      this.x = x;   this.y = y;
      this.width = w;   this.height = h;
   }
   ...
   public boolean equals(Object obj) {
      if (obj instanceof Rectangle) {
         Rectangle r = (Rectangle) obj;
         return ((x == r.x) && (y == r.y) &&
                 (width == r.width) && (height == r.height));
      }
      return false;
   }
   ...
}
```

null

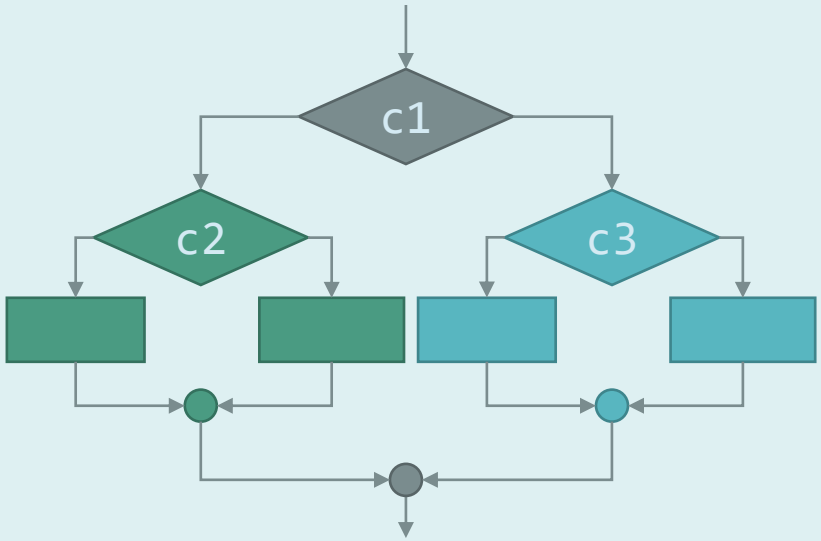
- ถ้า reference variable เก็บค่าพิเศษ คือ null หมายถึงตัวแปรไม่อ้างถึงวัตถุใด
- Null ไม่ใช่ 0, ไม่ใช่ ' ' blank, ไม่ใช่ "" สตริงว่าง
- ใช้ตรวจสอบค่าของ reference variable ได้

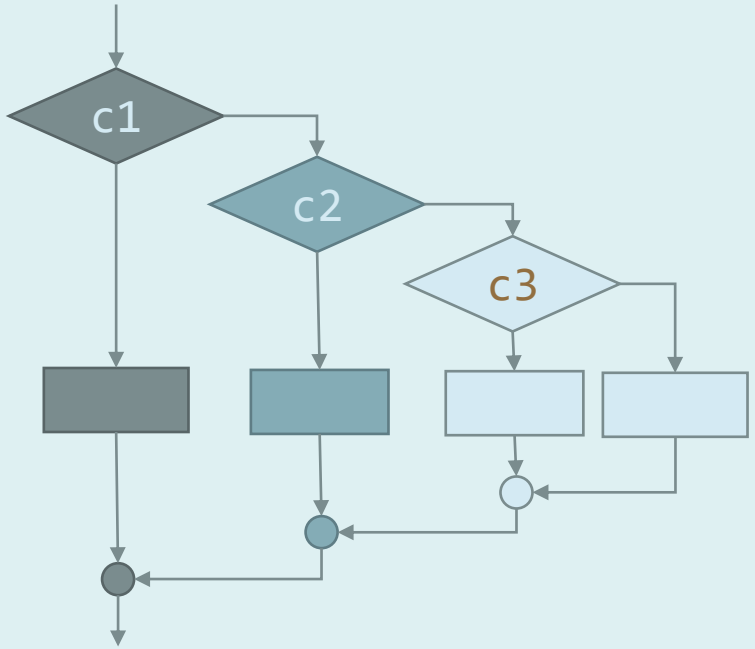
ตัวอย่าง

```
String name;
...
if (name==null) System.out.println("No object");
```

Dangling else

Nested If






```
if (c1) {
   if (c2) { ... }
   else { ... }
} else {
  if (c3) { ... }
  else { ... }
}
```

```
if (c1) {
   if (c2) { ... }
   else {
      if (c3) { ... }
      else { ... }
   }
}
```

Dangling else

- การที่มี else น้อยกว่า if คือ if บางตัวไม่มี else ที่คู่กัน
- บอกได้อย่างไรว่า else แต่ละตัวคู่กับ if ตัวใด
- จาวา ถือว่า else แต่ละตัวคู่กับ if ตัวใกล้สุดที่อยู่ก่อนหน้า คือ

```
if (c1)
    if (c2) { ... }
    else { ... }
```

- ถ้า if-else ที่ซ้อนกัน มีจำนวน if กับ else เท่ากัน ดูอย่างไรว่า else แต่ละตัวคู่กับ if ตัวใด 

Dangling else

```
if ( c1 )
if ( c2 ) { ... }
else { ... }
```

- ถ้าต้องการให้ else คู่กับ if ตัวที่อยู่ไกลกว่า ต้องกำหนดให้ชัดเจนว่า if ตัวที่อยู่ใกล้กว่าไม่มี else คู่กัน ดังนี้

```
if ( c1 ) {
   if ( c2 ) { ... }
} else { ... }
```

ในไพธอน

- มีปัญหา dangling else หรือไม่
- ถ้ามีปัญหานี้ แก้อย่างไร

Short-circuit Evaluation

Lazy (or short-circuit) evaluation

`if ( a && b ) { … }`

- จาวาหาค่า (evaluate) `a`
- ถ้า `a` เป็น `false` ทำให้ `(a && b)` เป็น `false` อยู่แล้ว ดังนั้นจาวาไม่หาค่า `b`
- ถ้า `a` เป็น `true` ค่าของ `(a && b)` ขึ้นกับค่า `b` ดังนั้นจาวาหาค่า `b`

`if ( a || b ) { … }`

- จาวาหาค่า (evaluate) `a`
- ถ้า `a` เป็น `true` ทำให้ `(a || b)` เป็น `true` อยู่แล้ว ดังนั้นจาวาไม่หาค่า `b`
- ถ้า `a` เป็น `false` ค่าของ `(a && b)` ขึ้นกับค่า `b` ดังนั้นจาวาหาค่า `b`

ทำไมโปรแกรมเมอร์ต้องสนใจ

Lazy (or short-circuit) evaluation

มีการรียกเมทอดซ่อนอยู่

```
if (input != null && Integer.parseInt(input)>0) ...
```

- ถ้า `input==null` เงื่อนไขเป็น false แล้ว ดังนั้นไม่ทำ `parseInt`
- ถ้า `input!=null` เงื่อนไขเป็น true แล้ว ดังนั้นต้องทำ `parseInt`

เปรียบเทียบ

```
if (Integer.parseInt(input)>0 && input != null) ...
```

- ทำ `parseInt` ก่อน ไม่ว่า `input` จะเป็น `null` หรือไม่
  - ถ้า `input==null` ทำ `parseInt` แล้วเกิด error
  - ถ้า `input!=null` ทำ `parseInt` ได้

Switch statement

Switch structure คล้าย if-else-if ladder

int, char, String, enum type

```
switch (exp) {
    case v1: …
             break;
    case v2: …
             break;
    …
    case vn: …
             break;
    default: …
             break;
}
```

```
if (exp==v1)
{  …  }
else if (exp==v2)
{  …  }
else …

else if (exp==vn)
{  …  }
else
{  …  }
```

ตัวอย่างการใช้ switch

```
int dayNum = 3;
String dayStr;

switch (dayNum)
{  case 1:  dayStr = "Sunday";      break;
   case 2:  dayStr = "Monday";      break;
   case 3:  dayStr = "Tuesday";     break;
   case 4:  dayStr = "Wednesday";   break;
   case 5:  dayStr = "Thursday";    break;
   case 6:  dayStr = "Friday";      break;
   case 7:  dayStr = "Saturday";    break;
   default: dayStr = "";
            System.out.println("error"); break;
}
```

Enumeration Types

- ชนิดข้อมูลที่ผู้ใช้กำหนด (user-defined type) โดยค่าที่เป็นได้สำหรับข้อมูลชนิดนี้มีจำนวนจำกัด (finite) เช่น

```
public enum Gender { MALE, FEMALE }
```

- enum สร้างคลาสพิเศษที่เก็บค่าคงที่จำนวนหนึ่ง และใช้ชื่อคลาสเป็นชนิดข้อมูล เช่น

```
Gender studentGender;
```

- อ้างถึงค่าชนิดนี้โดยบอกคลาสและค่า เช่น

```
studentGender = Gender.MALE;
```

ตัวอย่างการใช้ switch กับ enum type

```
public enum Day {
   SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY }

Day today = Day.SUNDAY;  // change to other value
switch (today) {
   case SUNDAY:
   case SATURDAY:
      System.out.println("Weekend");
      break;
   case WEDNESDAY:
      System.out.println("Midweek");
      break;


   default:


      System.out.println("Weekday");
      break;
}
```

Testing

Testing

- Code coverage
  - การทดสอบ ควรครอบคลุมทุกส่วนของโปรแกรม
- มี if statement
  - การทดสอบ ครอบคลุมทุกทางที่เป็นไปได้