บทที่ 2

คำสั่งควบคุมและโครงสร้างแบบซ้อน

บรรยายโดย ผศ.ดร.ธราวิเชษฐ์ ธิติจรูญโรจน์

หัวข้อ

01 โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

02 โครงสร้างการเขียนโปรแกรมแบบมีเงื่อนไข

03 ตัวดำเนินการทางเปรียบเทียบและตรรกศาสตร์

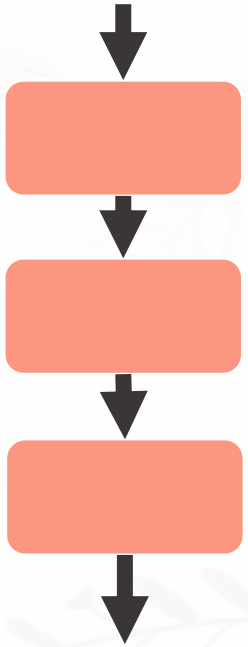
หัวข้อ

01 โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

02 โครงสร้างการเขียนโปรแกรมแบบมีเงื่อนไข

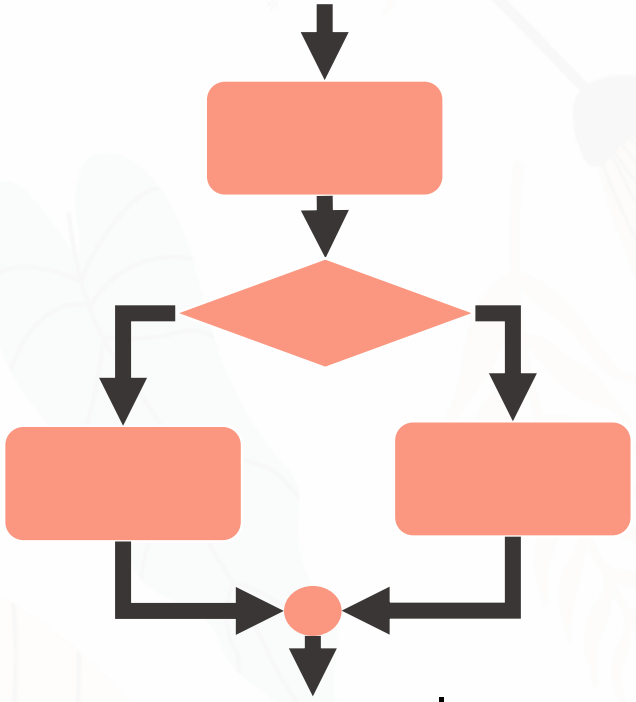
03 ตัวดำเนินการทางเปรียบเทียบและตรรกศาสตร์

โครงสร้างการเขียนโปรแกรม



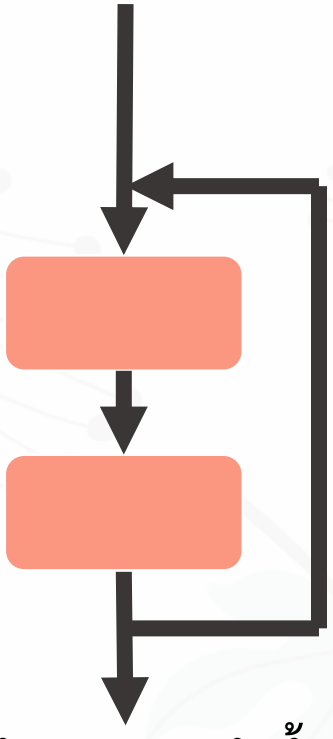
การทำงานแบบเรียงลำดับ

(Sequence)



การทำงานแบบมีเงื่อนไข

(Selection)



การทำงานแบบทำซ้ำ

(Iteration)

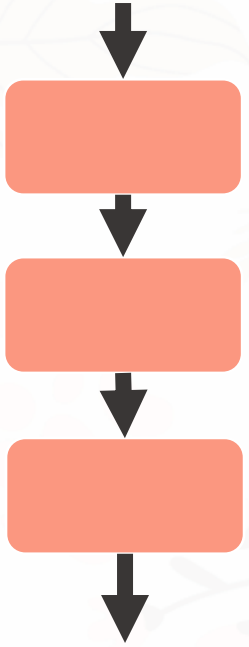
หัวข้อ

01 โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

02 โครงสร้างการเขียนโปรแกรมแบบมีเงื่อนไข

03 ตัวดำเนินการทางเปรียบเทียบและตรรกศาสตร์

โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ



input

จงเขียนโปรแกรมรับค่าปี พ.ศ. จากผู้ใช้งาน จากนั้นโปรแกรมจะแปลงจากปี พ.ศ. ไปเป็นปี ค.ศ. และแสดงผลทางจอภาพ

output

โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

-> พ.ศ.-543

จงเขียนโปรแกรมรับค่าปี พ.ศ. จากผู้ใช้งาน จากนั้นโปรแกรมจะแปลงจากปี พ.ศ. ไปเป็นปี ค.ศ. และแสดงผลทางจอภาพ

โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

1 รับปี พ.ศ.

2 คำนวณ ค.ศ -> พ.ศ.-543

3 แสดง ค.ศ.

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        int yearBC, yearAC;
        Scanner sc = new Scanner(System.in);
1       yearBC = sc.nextInt();

2       yearAC = yearBC - 543;


3       System.out.println(yearAC);
    }
}
```

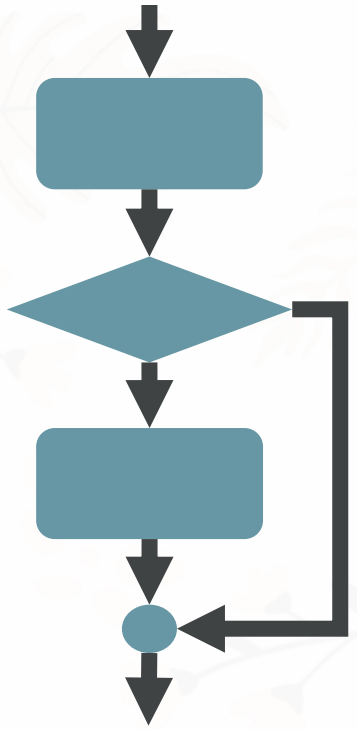
หัวข้อ

01 โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

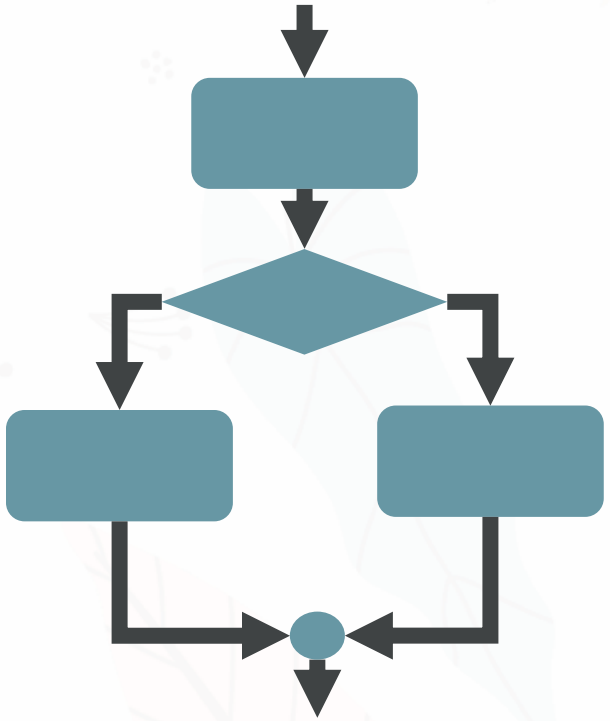
02 โครงสร้างการเขียนโปรแกรมแบบมีเงื่อนไข

03 ตัวดำเนินการทางเปรียบเทียบและตรรกศาสตร์

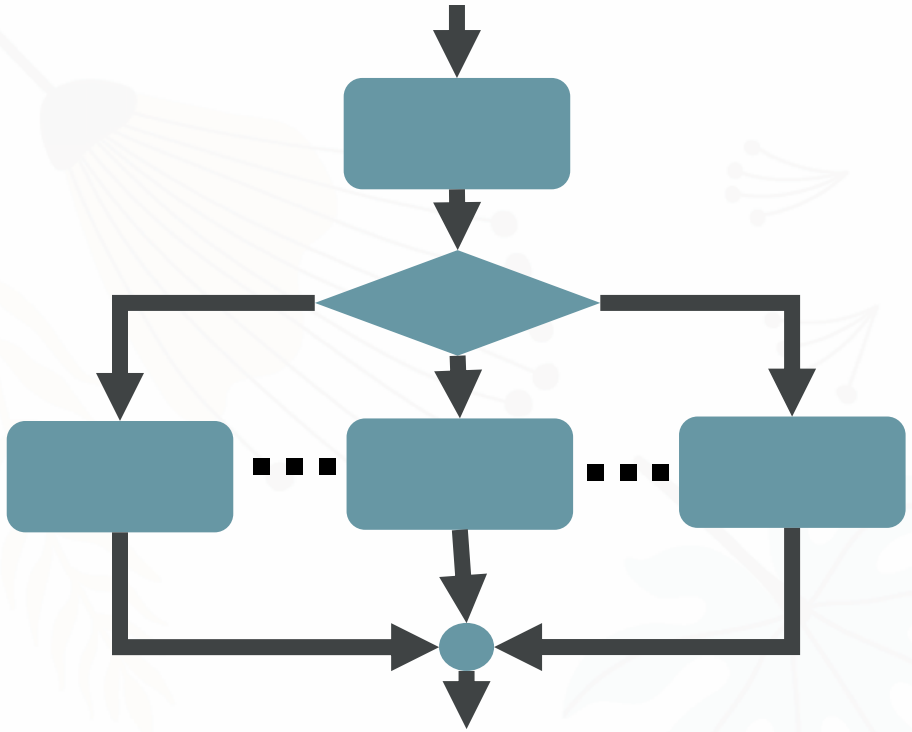
การทำงานแบบมีเงื่อนไข (Selection)



(Single Selection)

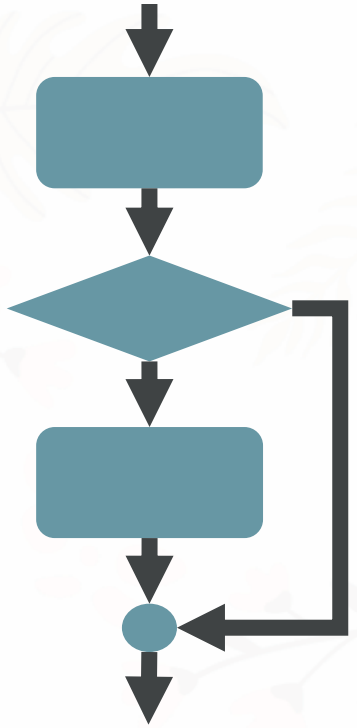


(Double Selection)



(Multiple Selection)

การทำงานแบบมีเงื่อนไข (Selection)

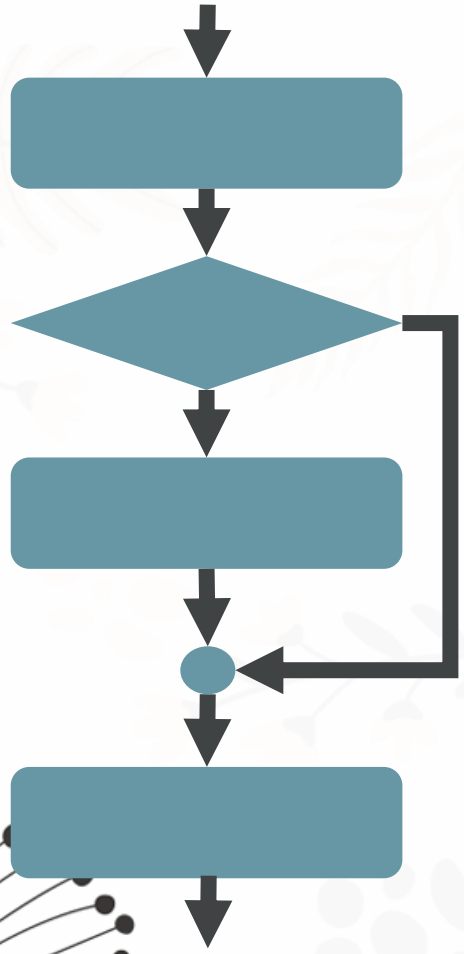


(Single Selection)

(Double Selection)

(Multiple Selection)

การทำงานแบบมีเงื่อนไข Single Selection



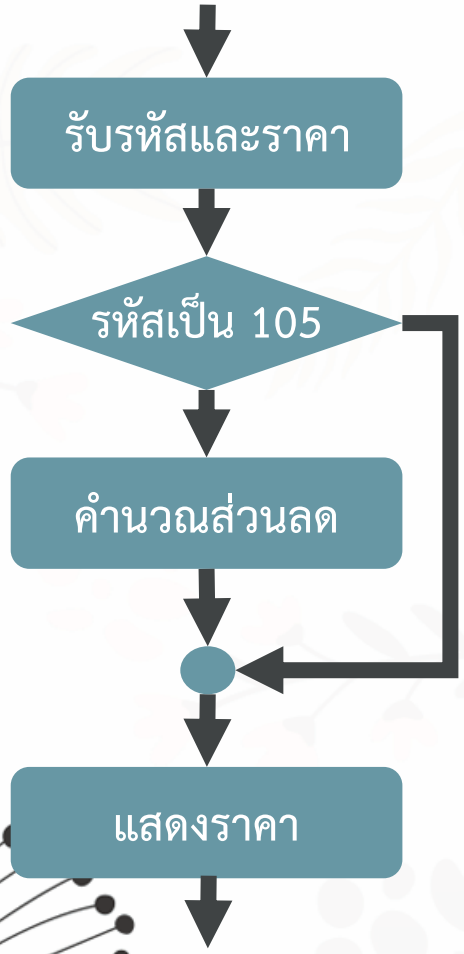
output

ให้นักศึกษาเขียนโปรแกรมเพื่อคำนวณและแสดงราคาสินค้าให้กับลูกค้า โดยที่ลูกค้าที่มีรหัสสมาชิกเป็น 105 จะได้รับส่วนลดราคาสินค้าลง 10 % แต่ถ้าลูกค้าคนอื่น ๆ จะคิดราคาเท่าเดิม โดยจะรับค่ารหัสลูกค้าและราคาสินค้าผ่านทางคีย์บอร์ด

process

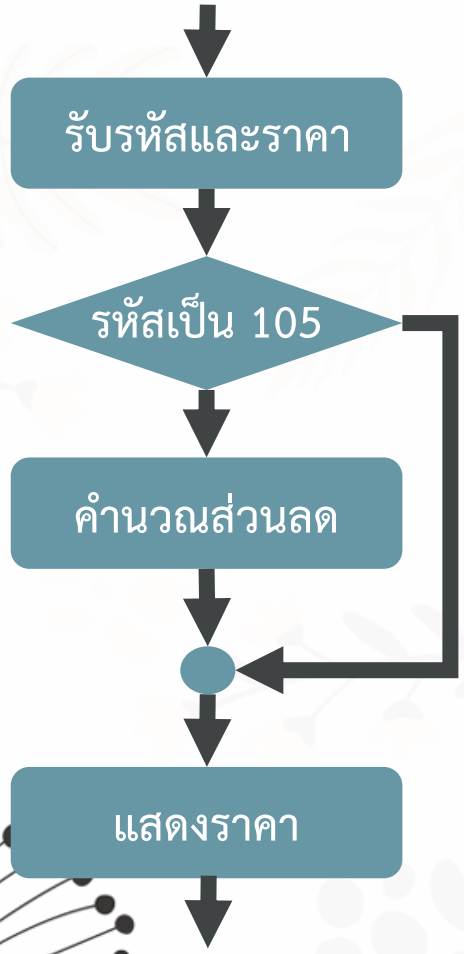
Input

การทำงานแบบมีเงื่อนไข Single Selection



ให้นักศึกษาเขียนโปรแกรมเพื่อคำนวณและแสดงราคาสินค้าให้กับลูกค้า โดยที่ลูกค้าที่มีรหัสสมาชิกเป็น 105 จะได้รับส่วนลดราคาสินค้าลง 10 % แต่ถ้าลูกค้าคนอื่น ๆ จะคิดราคาเท่าเดิม โดยจะรับค่ารหัสลูกค้าและราคาสินค้าผ่านทางคีย์บอร์ด

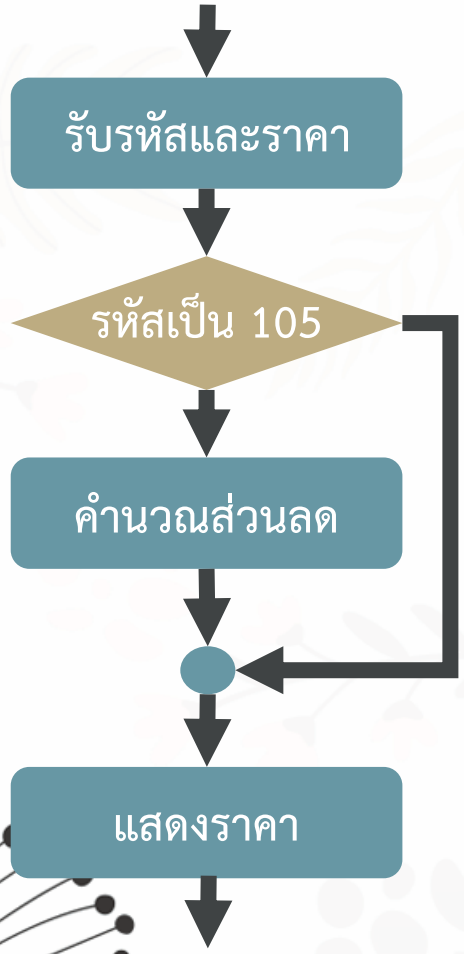
การทำงานแบบมีเงื่อนไข Single Selection



ให้นักศึกษาเขียนโปรแกรมเพื่อคำนวณและแสดงราคาสินค้าให้กับลูกค้า โดยที่ลูกค้าที่มีรหัสสมาชิกเป็น 105 จะได้รับส่วนลดราคาสินค้าลง 10 % แต่ถ้าลูกค้าคนอื่น ๆ จะคิดราคาเท่าเดิม โดยจะรับค่ารหัสลูกค้าและราคาสินค้าผ่านทางคีย์บอร์ด

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        int itemCode;
        double itemPrice;

        Scanner sc = new Scanner(System.in);
        itemCode = sc.nextInt();
        itemPrice = sc.nextDouble();

        if (itemCode == 105){
            itemPrice = 0.9 * itemPrice;
        }

        System.out.println(itemPrice);

    }
}
```

การทำงานแบบมีเงื่อนไข Single Selection



การทำงานแบบมีเงื่อนไข Single Selection ในภาษาจาวาต้องอาศัยคำสั่ง if

true

```
if (เงื่อนไข)
{
    statement;
}
```

เงื่อนไข (Condition) คืออะไร

ในการเขียนโปรแกรม “**เงื่อนไข (Condition)**” ในคำสั่งตัดสินใจจะมีความหมายว่า

- “นิพจน์ที่คืนค่าเป็น boolean” หรือสามารถกล่าวได้
- “นิพจน์ที่คืนค่าเป็นจริง (true) หรือเท็จ (false)”

ซึ่งในการเขียนโปรแกรม “**เงื่อนไข (Condition)**” สามารถนิพจน์ได้ 4 รูปแบบ ได้แก่

- ค่าคงที่
- ตัวแปร (Variable)
- ตัวดำเนินการ (Operation)
- เมธอด (Method)

นิพจน์ที่คืนค่าความจริง

ซึ่งในการเขียนโปรแกรม “**เงื่อนไข (Condition)**” สามารถนิพจน์ได้ 4 รูปแบบ ได้แก่

- **รูปแบบที่** 1 ค่าคงที่

```
public class Main {
   public static void main(String args[]) {

      if (true){
            statement
      }

   }
}
```

นิพจน์ที่คืนค่าความจริง

ซึ่งในการเขียนโปรแกรม “**เงื่อนไข (Condition)**” สามารถนิพจน์ได้ 4 รูปแบบ ได้แก่

- **รูปแบบที่ 2** ตัวแปร (Variable)

```
public class Main {
   public static void main(String args[]) {

      boolean isPrimeNumber = true;
      if (isPrimeNumber){
            statement
      }

   }
}
```

```
public  class Main {
  public static void main(String[] args) {
      int correctedPassword = 12345;
      int givenPassword = 12345;
      boolean isMatch = (correctedPassword == givenPassword);

      // The “if (isMatch)” flag not that useful.
      // It is more appropriate than “if (isMatch == true)”.

      if (isMatch){
          System.out.println(”Match");
      }
   }
}
```

นิพจน์ที่คืนค่าความจริง

ซึ่งในการเขียนโปรแกรม “**เงื่อนไข (Condition)**” สามารถนิพจน์ได้ 4 รูปแบบ ได้แก่

- **รูปแบบที่ 3** ตัวดำเนินการ (Operation)

```
public class Main {
    public static void main(String args[]) {
        int salary = 20000;
        if ((salary > 0) && (salary <= 30000)){
            statement
        }

    }
}
```

- **รูปแบบที่ 4** เมธอด (Method)

```
public class Main {
    public static boolean isPositive(int num){
        if (num > 0)
                return true;
        else
                return false;
    }
    public static void main(String[] args){
        int num = 7;
        if (isPositive(num)){
                statement
        }
    }
}
```

หัวข้อ

01 โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

02 โครงสร้างการเขียนโปรแกรมแบบมีเงื่อนไข

03 ตัวดำเนินการทางเปรียบเทียบและตรรกศาสตร์

ตัวดำเนินการในภาษาจาวา

- เปรียบเทียบ (Comparison Operator)

```
< > <= >= == !=
```

- ตรรกะศาสตร์ (Logical Operator)

```
&& || & | !
```

ตัวดำเนินการเชิงเปรียบเทียบ

เครื่องหมาย	ความหมาย	ตัวอย่าง	ผลลัพธ์
<	น้อยกว่า	3<4	true
<=	น้อยกว่าหรือเท่ากับ	3<=4	true
>	มากกว่า	3>4	false
>=	มากกว่าหรือเท่ากับ	3>=4	false
==	เท่ากับ	3==4	false
!=	ไม่เท่ากับ	3!=4	true

ใช้ในการเปรียบเทียบสองค่า ตัวดำเนินการเหล่านี้จะกำหนดความสัมพันธ์ระหว่างค่าและส่งกลับผลลัพธ์เป็นค่าจริงหรือเท็จตามการเปรียบเทียบ

ตัวดำเนินการเชิงเปรียบเทียบ

input	operator	input		output
char int double	< > <= >=	char int double	->	boolean

input	operator	input		output
char int double boolean	== !=	char int double boolean	->	boolean

ตัวอย่างโปรแกรม

ตัวดำเนินการเชิงเปรียบเทียบ

```
public  class Main {
    public static void main(String[] args) {

        System.out.println(1 == 1);

        System.out.println('a' > 64);
        // เปรียบเทียบรหัส ASCII Code

        System.out.println('a' != 'b');

        System.out.println(true == false);

        int x = 5, y = 4;
        boolean b1 = (x!=y);
        System.out.println(b1);
    }
}
```

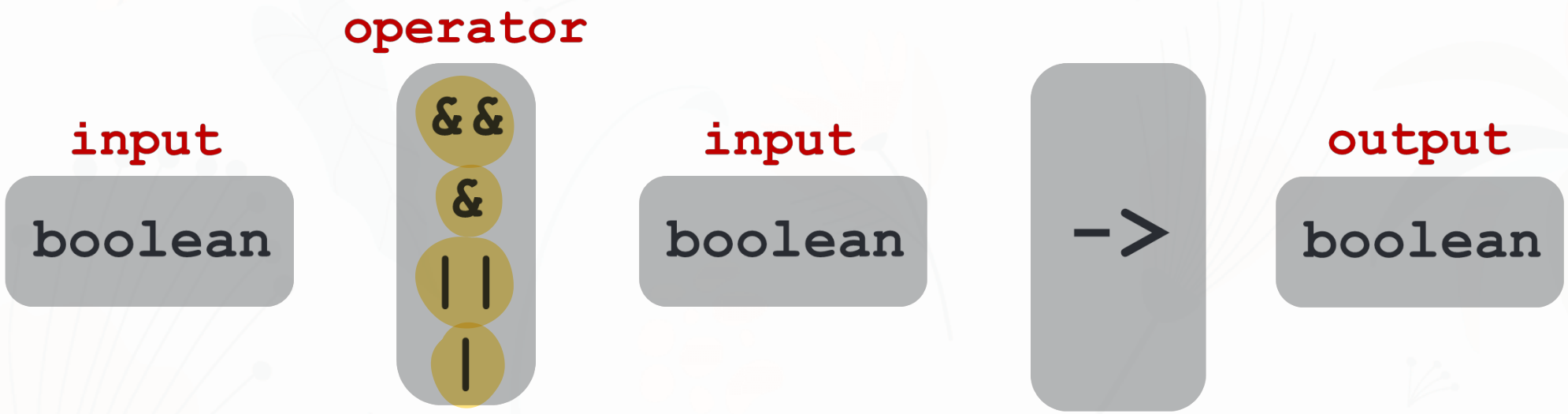
```
output:
true
true
true
false
true
```

ตัวดำเนินการทางตรรกศาสตร์

เครื่องหมาย	ความหมาย
!	กลับค่าทางตรรกะ
&& หรือ &	AND ค่าทางตรรกะ
|| หรือ |	OR ค่าทางตรรกะ
^	Exclusive-OR ค่าทางตรรกะ

ใช้เพื่อสร้างเงื่อนไขที่มีความซับซ้อนมากขึ้น โดยการรวมนิพจน์บูลีนตั้งแต่ 2 รายการขึ้นไปเข้าด้วยกัน ผลลัพธ์ของชุดค่าผสมดังกล่าวคือค่าบูลีน (จริงหรือเท็จ)

ตัวดำเนินการทางตรรกศาสตร์



การรวมเงื่อนไขย่อย (Sub-condition)

ในจาวา "การรวมเงื่อนไขย่อย" หมายถึง การสร้างเงื่อนไขที่มีความซับซ้อนจากการประกอบจากหลายเงื่อนไขย่อยร่วมกันในคำสั่งเดียว โดยอาศัยตัวดำเนินการเชิงตรรกะ ซึ่งช่วยให้การตัดสินใจในโค้ดมีความซับซ้อนมากขึ้นโดยการพิจารณาจากหลายเกณฑ์พร้อมกัน การรวมกันของเงื่อนไขมักใช้ในคำสั่ง เช่น if, else if, while และ for เพื่อกำหนดโฟลว์ของโปรแกรมตามปัจจัยหลายประการ

Condition 1 | Logical Operator | Condition 2 | Logical Operator | Condition 3

การรวมเงื่อนไขย่อย (Sub-condition)

```
public class Main {
    public static void main(String args[]) {
        int x = 15
        if ((x >= 10) && (x <= 20)) {
            System.out.println("correct");
        }

        String userRole = "admin";
        if (userRole.equals("admin") || userRole.equals("mod")) {
            System.out.println("User has special privileges.");
        }
    }
}
```

$\boxed{10 \leq x} \leq 20$

↓

true $\leq$ 20 → เทียบไม่ได้

$(10 \leq x)$ && $(x \leq 20)$

↓

true && true

↓

true

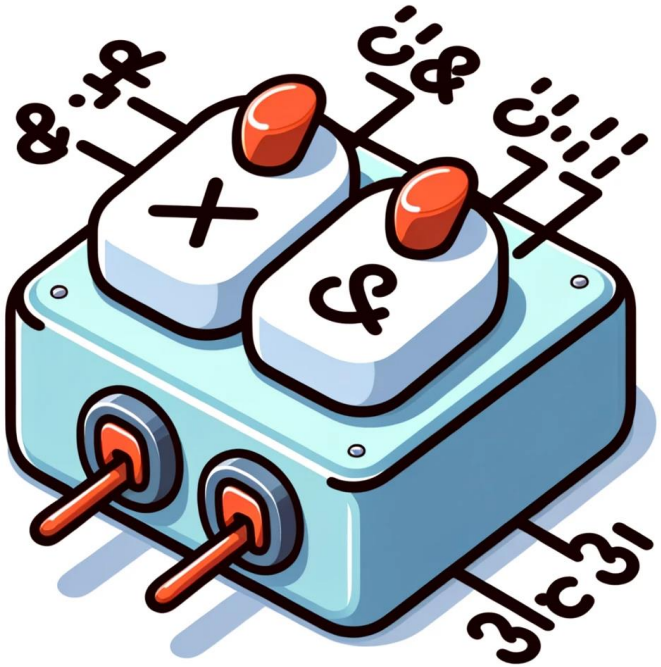
ตัวดำเนินการทางตรรกศาสตร์

input1	input2	and	or	XOR	not
P	Q	P && Q	P || Q	P ^ Q	! P
true	true	true	true	false	false
true	false	false	true	true	false
false	true	false	true	true	true
false	false	false	false	false	true

ตัวดำเนินการทางตรรกศาสตร์

Ignore (ไม่ประมวลผลเงื่อนไขที่ 2)

ตัวดำเนินการลัดวงจร (Short Circuit Operators) หมายถึง ตัวดำเนินการทางตรรกศาสตร์ที่สามารถหยุดการประมวลผล โดยจะไม่ประมวลผลเงื่อนไขย่อยที่เหลือจากเงื่อนไขทั้งหมด (เงื่อนไขประกอบ Combination of conditions) ก็ต่อเมื่อจาวาสามารถระบุผลลัพธ์ของเงื่อนไขได้จากการประมวลผลตัวถูกดำเนินการเพียงตัวแรก ซึ่งจะเกิดกับ && (Logical AND) และ || (Logical OR)



ตัวดำเนินการทางตรรกศาสตร์

หลักการทำงานของ Short Circuit Operators

- **สำหรับ && (AND)**

หากตัวถูกดำเนินการทางด้านซ้ายเป็นเท็จ **ผลลัพธ์จะเป็นเท็จ** โดยไม่คำนึงถึงค่าของตัวถูกดำเนินการทางขวามือ ทำให้จาวาไม่ประมวลผลตัวถูกดำเนินการทางขวา เนื่องจากไม่จำเป็น

- **สำหรับ || (OR)**

หากตัวถูกดำเนินการทางซ้ายเป็นจริง **ผลลัพธ์จะเป็นจริง** โดยไม่คำนึงถึงค่าของตัวถูกดำเนินการทางขวามือทำให้จาวาไม่ประมวลผลตัวถูกดำเนินการทางขวา เนื่องจากไม่จำเป็น

```
public  class Main {
  public static void main(String[] args) {
      int numA = 10;
      boolean numB = ((numA < 11) | (++numA >= 11));
      System.out.println(numA + " with " + numB);

      numB = ((numA < 11) || (++numA >= 11));
      System.out.println(numA + " with " + numB);

      numB = ((numA > 11) & (++numA >= 11));
      System.out.println(numA + " with " + numB);

      numB = ((numA > 11) && (++numA >= 11));
      System.out.println(numA + " with " + numB);

    }
}
```

true true 11 with true

true true 12 with true

false true 13 with false

true true 14 with true

หัวข้อ

01 โครงสร้างการเขียนโปรแกรมแบบเรียงลำดับ

02 โครงสร้างการเขียนโปรแกรมแบบมีเงื่อนไข

03 ตัวดำเนินการทางเปรียบเทียบและตรรกศาสตร์

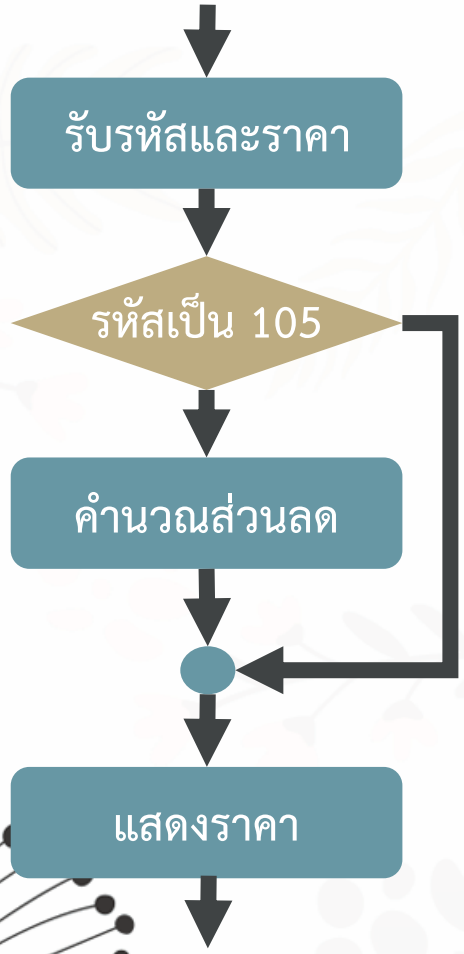
นิพจน์ที่คืนค่าความจริง

ซึ่งในการเขียนโปรแกรม “**เงื่อนไข (Condition)**” สามารถนิพจน์ได้ 4 รูปแบบ ได้แก่

- **รูปแบบที่ 3** ตัวดำเนินการ (Operation)

```
public class Main {
    public static void main(String args[]) {
        int salary = 20000;
        if ((salary > 0) && (salary <= 30000)){
            statement
        }

    }
}
```

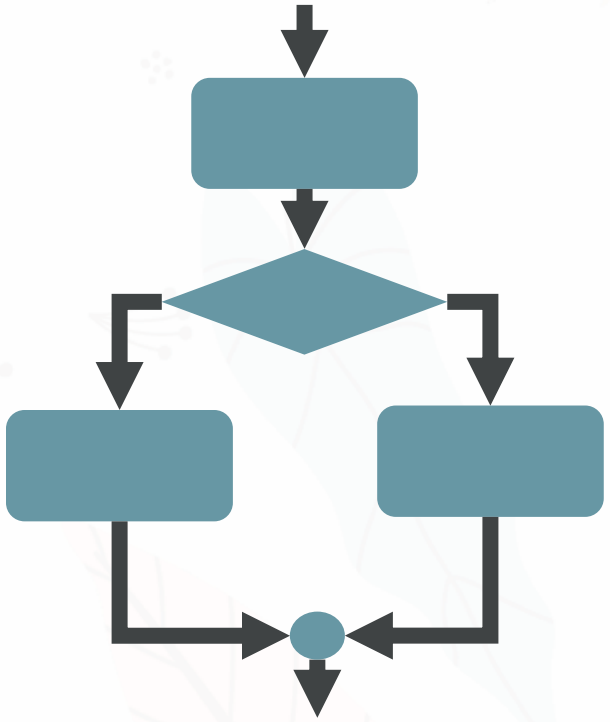
การทำงานแบบมีเงื่อนไข Single Selection



การทำงานแบบมีเงื่อนไข Single Selection ในภาษาจาวาต้องอาศัยคำสั่ง if

```
if (เงื่อนไข)
{
    statement;
}
```

การทำงานแบบมีเงื่อนไข (Selection)

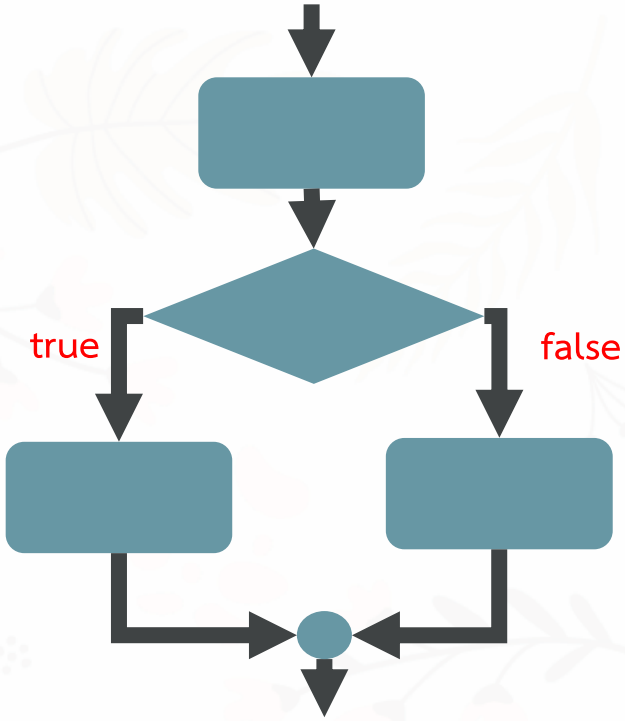


(Double Selection)

(Single Selection)

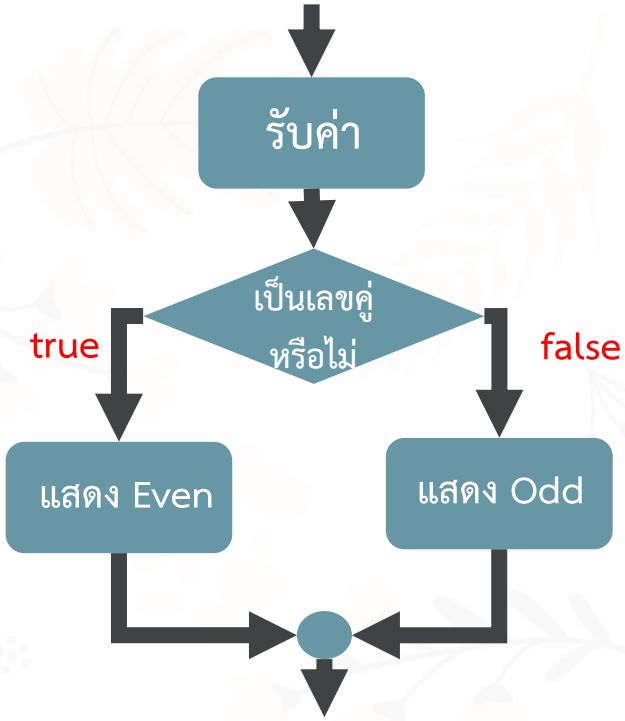
(Multiple Selection)

การทำงานแบบมีเงื่อนไข Double Selection



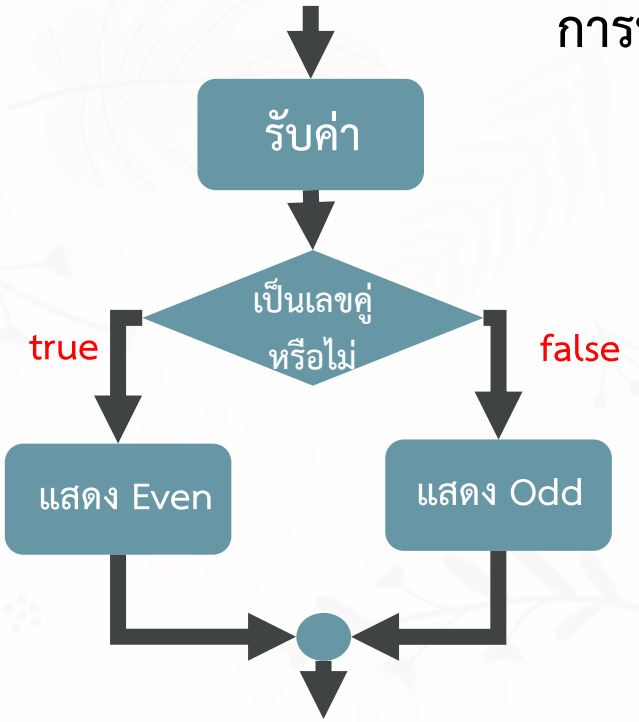
ให้นักศึกษาเขียนโปรแกรมรับค่าผ่านคีย์บอร์ดจำนวน 1 ค่า ถ้ามีค่าเป็นเลขคู่ ให้แสดงออกทางหน้าจอว่า “Even” แต่ถ้าเป็นเลขคี่ให้แสดงออกทางหน้าจอว่า “Odd”

การทำงานแบบมีเงื่อนไข Double Selection



ให้นักศึกษาเขียนโปรแกรมรับค่าผ่านคีย์บอร์ดจำนวน 1 ค่า ถ้ามีค่าเป็นเลขคู่ ให้แสดงออกทางหน้าจอว่า “Even” แต่ถ้าเป็นเลขคี่ให้แสดงออกทางหน้าจอว่า “Odd”

การทำงานแบบมีเงื่อนไข Double Selection



การทำงานแบบมีเงื่อนไข Double Selection ต้องอาศัยคำสั่ง if-else หรือ if-else if

* สุดท้ายจะเข้าช่องใดช่องหนึ่ง

```
if (เงื่อนไข){
    statement;     // นิพจน์ใน if จะทำก็ต่อเมื่อเงื่อนไขของ if เป็น true
} else {
    statement;     // นิพจน์ใน else จะทำก็ต่อเมื่อเงื่อนไขของ if เป็น false
}
```

1

* มีโอกาสไม่เข้าทั้งคู่

```
if (เงื่อนไข){
    statement;     // นิพจน์ใน if จะทำก็ต่อเมื่อเงื่อนไขของ if เป็น true
} else if (เงื่อนไข){{
    statement;     // นิพจน์ใน else if จะทำก็ต่อเมื่อเงื่อนไข else if เป็น true
}
```

2

การทำงานแบบมีเงื่อนไข Double Selection

1. **คำสั่ง if-else**

   เป็นการตัดสินใจแบบ 2 ทางเลือก โดยมีเพียงเงื่อนไขที่ขึ้นกับ if อย่างเดียว โดยที่ นิพจน์ใน if จะทำก็ต่อเมื่อเงื่อนไขของ if เป็นจริง ขณะที่ นิพจน์ใน else จะทำก็ต่อเมื่อเงื่อนไขของ if เป็นเท็จ ซึ่งเหมาะสมกับสถานการณ์ที่เราออกแบบไว้เพียง 2 ทางเลือกเท่านั้น

2. **คำสั่ง if-else if**

   เป็นการตัดสินใจแบบหลายทางเลือก ที่มีได้หลายเงื่อนไขที่จะตรวจสอบเงื่อนไขเป็นลำดับจากบนลงล่าง ซึ่งเงื่อนไขใน else if เป็น additional conditions ที่จะตรวจสอบก็ต่อเมื่อเงื่อนไขของ if หรือ else if ก่อนหน้าเป็น false นอกจากนี้ else if สามารถมีได้หลายอัน

ความแตกต่างระหว่าง (1) และ (2)

- **การใช้งาน if และ else**
  - นิพจน์ใน if จะทำก็ต่อเมื่อเงื่อนไขของ if เป็นจริง
  - ขณะที่ นิพจน์ใน else จะทำก็ต่อเมื่อเงื่อนไขของ if เป็นเท็จ
- **การใช้งาน if และ else if**
  - นิพจน์ใน if จะทำก็ต่อเมื่อเงื่อนไขของ if เป็นจริง
  - ขณะที่ นิพจน์ใน else if จะทำก็ต่อเมื่อเงื่อนไขของ else if เป็นจริง

```
import java.util.*;
public class Main {
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num = sc.nextInt();

1     if ( (num%2) == 0 ) {
            System.out.print("Even");
      } else {
            System.out.print("Odd");
      }

2     if ( (num%2) == 0 ) {
            System.out.print("Even");
      } else if( (num%2) == 1 ){
            System.out.print("Odd");
      }
  }
}
```

ความแตกต่างระหว่าง if-else และ if-else if

```
public class Main {
    public static void main(String[] args){
        Scanner tube = new Scanner(System.in);
        int num = tube.nextInt();
        if ( num > 0 ) {
            System.out.print("I+");
        } else {
            System.out.print("I-");
        }
    }
}
```

1

```
public class Main {
    public static void main(String[] args){
        Scanner tube = new Scanner(System.in);
        int num = tube.nextInt();
        if ( num > 0 ) {
            System.out.print("I+");
        } else if ( num < 0 ) {
            System.out.print("I-");
        }
    }
}
```

2

การทำงานแบบเงื่อนไข

Double Selection

การเขียนโปรแกรมแบบ 2 ทางเลือก สามารถเขียนได้หลายรูปแบบ แล้วแต่บริบทและความชำนาญของผู้เขียน จากตัวอย่าง จะพบว่าการเขียนเงื่อนไขสำหรับเช็คเลขคู่หรือเลขคี่นั้นสามารถเขียนได้ 3 แบบ ดังตัวอย่าง

```
if ( (num%2) ==  0 ) {
   System.out.print("Even number");
} else {
   System.out.print("Odd number");
}
```

1

```
if ( (num%2) == 0 ) {
    System.out.print("Even number");
} else if ( (num%2) != 0 ) {
    System.out.print("Odd number");
}
```

2

```
if ( (num%2) == 0 ) {
    System.out.print("Even number");
} else if ( (num%2) == 1 ) {
    System.out.print("Odd number");
}
```

3

การเปรียบเทียบการเท่ากัน

== compare address

.equals() ใช้กับตัวแปรอ้างอิง

INT

```
int n1 = sc.nextInt();
int n2 = sc.nextInt();
if( n1 == n2 ) {
        System.out.println("==" );
} else {
        System.out.println("!=");
}
```

DOUBLE

```
double n1 = sc.nextDouble();
double n2 = sc.nextDouble();
if( Math.abs(n1 - n2) < 0.001 ) {
        System.out.println("==" );
}else {
        System.out.println("!=");
}
```

CHAR

```
char n1 = sc.next().charAt(0);
if( n1 == 'A' ) {
        System.out.println("==" );
} else {
        System.out.println("!=");
}
```

STRING

```
String n1 = sc.nextLine();
String n2 = sc.nextLine();
if( n1.equals(n2) ) {
        System.out.println("==" );
}else {
        System.out.println("!="); }
}
```

การเขียน Double Selection แบบย่อ

รูปแบบคำสั่ง

```
variable = condition ? a : b;
```

ถ้า condition เป็น true แล้ว variable จะมีค่าเท่ากับ a แต่ถ้า condition เป็น false แล้ว variable จะมีค่าเท่ากับ b โดยที่ “?” ใช้คั่นระหว่าง condition กับนิพจน์ และ “:” ใช้คั่นระหว่างนิพจน์ a และ b

ตัวอย่าง

```
int price = 99;
int withdraw = price <= 100 ? 100 : 200;
```

การใช้ if ซ้อนกันในการเขียนโปรแกรม

```
if (เงื่อนไข 1) {

    if (เงื่อนไข 2) {
        statements 1
    } else {
        statements 2
    }

} else {
    statements 3
}
```

การเขียน "nested if" หมายถึง

คำสั่ง if ที่มีอยู่ภายในคำสั่ง if อื่น ช่วยให้สามารถตัดสินใจที่ซับซ้อนได้มากขึ้น โดยการตรวจสอบเงื่อนไขต่าง ๆ ในลักษณะลำดับชั้น

ลักษณะการทำงานของ nested if มีรายละเอียดดังนี้

เมื่อเงื่อนไข if ภายนอกเป็นจริง คำสั่ง if ภายในจะถูกดำเนินการและตรวจสอบ ซึ่งจะทำให้คุณสามารถตัดสินใจโดยใช้เงื่อนไขหลายเกณฑ์ได้ โดยการตัดสินใจครั้งหนึ่งอาจขึ้นอยู่กับผลลัพธ์ของอีกเงื่อนไขหนึ่ง

การใช้ if ซ้อนกันในการเขียนโปรแกรม

มีประโยชน์มากมายดังนี้

- **การตัดสินใจที่ซับซ้อน**

การเขียนคำสั่ง **nested-if** ช่วยให้โปรแกรมเมอร์สามารถตัดสินใจตามเงื่อนไขหลาย ๆ อย่าง ซึ่งเป็นสิ่งที่มีประโยชน์เมื่อการตัดสินใจหนึ่ง ๆ ขึ้นอยู่กับการตัดสินใจอื่น ๆ

- **การอ่านโค้ดที่ง่ายขึ้น**

การเขียนคำสั่ง **nested-if** ที่โครงสร้างถูกต้องสามารถทำให้โค้ดอ่านง่ายขึ้นและชัดเจนมากขึ้นในมุมมองของการไหลของตรรกะ

- **หลีกเลี่ยงการตรวจสอบซ้ำ**

ด้วยการเขียน **nested-if** สามารถช่วยหลีกเลี่ยงการตรวจสอบที่ไม่จำเป็นได้

การใช้ if ซ้อนกันในการเขียนโปรแกรม

มีประโยชน์มากมายดังนี้

- **การแบ่งแยก**

การเขียนคำสั่ง **nested-if** สามารถแบ่งแยกในการจัดการสถานการณ์ที่ต่างจุดประสงค์กันได้

- **ประสิทธิภาพ**

ในบางกรณีการเขียนคำสั่ง **nested-if** อาจมีประสิทธิภาพมากกว่าการใช้ “การรวมเงื่อนไขย่อย” หรืออาศัยตัวดำเนินการทางตรรกะที่ซับซ้อน

- **ความยืดหยุ่น**

การเขียนคำสั่ง **nested-if** ให้ความยืดหยุ่นในการจัดการเงื่อนไขต่าง ๆ และการรวมกันของเงื่อนไข

- **การจัดการข้อผิดพลาด**

การเขียนคำสั่ง **nested-if** สามารถใช้สำหรับการจัดการข้อผิดพลาดที่มีความจำเพาะกรณีได้

ตัวอย่าง
การใช้ nested-if

```
import java.util.*;

public  class Main {

  public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       int score = sc.nextInt();
       if ( score <= 100 && score >= 0 ) {
              if ( score >= 50) {
                     System.out.print("Pass");
              } else if( score < 50 ){
                     System.out.print("Fail");
              }
       } else {
           System.out.print("Score out of range.");
       }

  }

}
```

การใช้ if ซ้อนกันในการเขียนโปรแกรม

อย่างไรก็ตาม การใช้ if ซ้อนกันมากเกินไปอาจนำไปสู่ปัญหาได้ อาทิเช่น

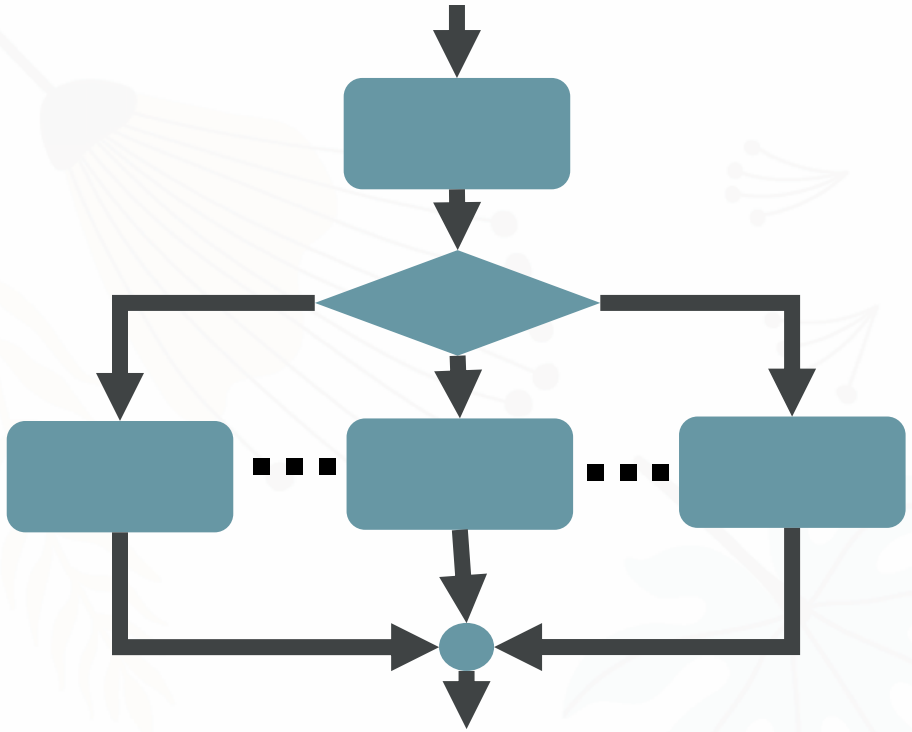
- **ความยากในการอ่าน** คือ การซ้อนเกินไปทำให้โค้ดยากต่อการอ่านและติดตาม
- **ปัญหาในการบำรุงรักษา** คือ โค้ดที่ซ้อนกันลึก ๆ อาจกลายเป็นที่ยากในการบำรุงรักษาและอัปเดต
- **เพิ่มโอกาสในการผิดพลาด** คือ การซ้อนกันที่ซับซ้อนมากขึ้นเพิ่มโอกาสในการเกิดข้อผิดพลาดทางตรรกะ

เพื่อหลีกเลี่ยงปัญหาเหล่านี้ ควรใช้ if ซ้อนกันเมื่อมีประโยชน์ชัดเจนและพิจารณาการปรับเปลี่ยนโครงสร้างควบคุมอื่น ๆ

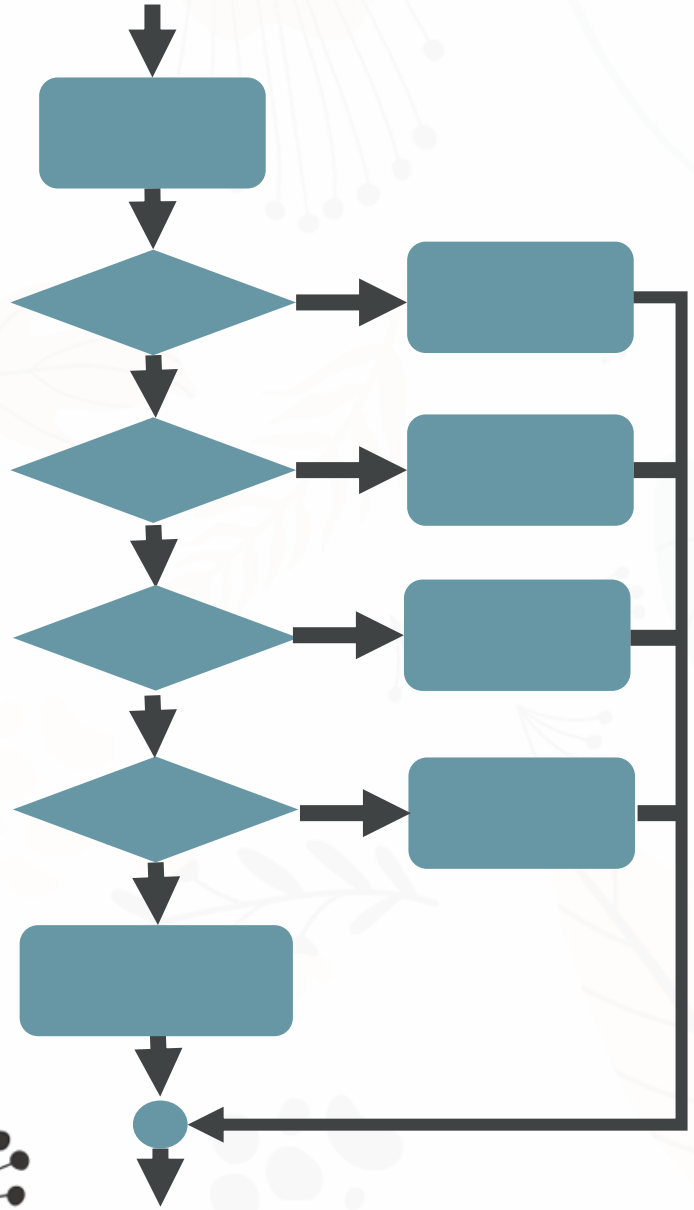
การทำงานแบบมีเงื่อนไข (Selection)

(Single Selection)

(Double Selection)



(Multiple Selection)

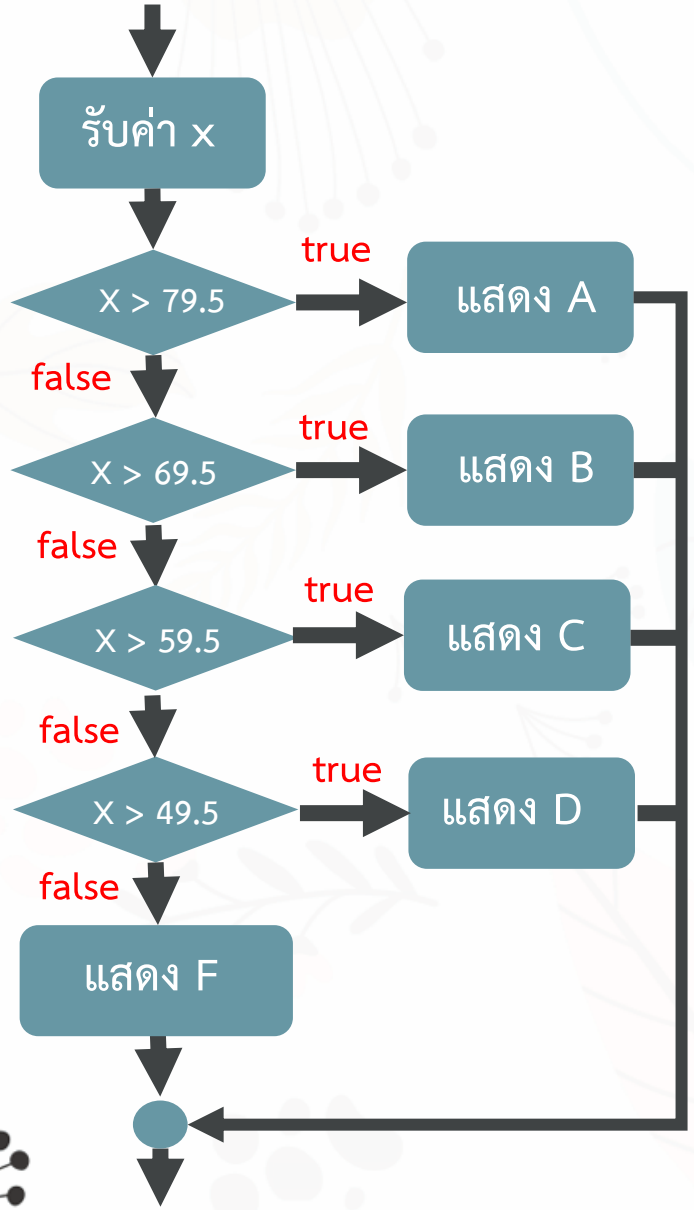


การทำงานแบบมีเงื่อนไข

Multiple Selection

ให้นักศึกษาเขียนโปรแกรมรับคะแนน (x) ผ่านคีย์บอร์ด เพื่อคำนวณหาเกรดที่นักศึกษาจะได้ และแสดงผลออกทางจอภาพ โดยที่

- A เมื่อ คะแนน > 79.5
- B เมื่อ คะแนน > 69.5
- C เมื่อ คะแนน > 59.5
- D เมื่อ คะแนน > 49.5
- F เมื่อ คะแนน < 49.5



การทำงานแบบมีเงื่อนไข

Multiple Selection

ให้นักศึกษาเขียนโปรแกรมรับคะแนน (x) ผ่านคีย์บอร์ด เพื่อคำนวณหาเกรดที่นักศึกษาจะได้ และแสดงผลออกทางจอภาพ โดยที่

- A เมื่อ คะแนน > 79.5
- B เมื่อ คะแนน > 69.5
- C เมื่อ คะแนน > 59.5
- D เมื่อ คะแนน > 49.5
- F เมื่อ คะแนน < 49.5

การทำงานแบบมีเงื่อนไข Multiple Selection

ให้นักศึกษาเขียนโปรแกรมรับคะแนน (x) ผ่านคีย์บอร์ด เพื่อคำนวณหาเกรดที่นักศึกษาจะได้ และแสดงผลออกทางจอภาพ โดยที่

- A เมื่อ คะแนน > 79.5
- B เมื่อ คะแนน > 69.5
- C เมื่อ คะแนน > 59.5
- D เมื่อ คะแนน > 49.5
- F เมื่อ คะแนน < 49.5

```
public class Main {
    public static void main( String[] args ){
        int score = 52;
        if( score > 79.5 ){
                System.out.println("A");
        } else if( score > 69.5 ){
                System.out.println("B");
        } else if( score > 59.5 ){
                System.out.println("C");
        } else if( score > 49.5 ){
                System.out.println("D");
        } else {
                System.out.println("F");
        }
    }
}
```

การทำงานแบบมีเงื่อนไข Multiple Selection

กรณีเป็นการตัดสินใจแบบหลายทางเลือกที่มีได้หลายเงื่อนไขที่ จาวาจะตรวจสอบเงื่อนไขเป็นลำดับจากบนลงล่าง โดยจะต้องเริ่มตรวจสอบจากเงื่อนไขแรกด้วยคำสั่ง if เสมอ และเงื่อนไขถัดมาจะอาศัยคำสั่ง else if และเงื่อนไขสุดท้ายจะอาศัยคำสั่ง else หรือ else if ก็ได้ แล้วแต่สถานการณ์

```
if(เงื่อนไขที่ 1){
        Statement 1
} else if(เงื่อนไขที่ 2){
        Statement 2
} else if(เงื่อนไขที่ 3){
        Statement 3
} else {
        Statement 4
}
```

```
if(เงื่อนไขที่ 1){
        Statement 1
} else if(เงื่อนไขที่ 2){
        Statement 2
} else if(เงื่อนไขที่ 3){
        Statement 3
} else if(เงื่อนไขที่ 4){
        Statement 4
}
```

การทำงานแบบมีเงื่อนไข Multiple Selection

```java
public class Main {
    public static void main( String[] args ){
        int score = 52;
        if( score > 79.5 ){
                System.out.println("A");
        } else if( score > 69.5 ){
                System.out.println("B");
        } else if( score > 59.5 ){
                System.out.println("C");
        } else if( score > 49.5 ){
                System.out.println("D");
        } else {
                System.out.println("F");
        }
    }
}
```

1

```java
public class Main {
    public static void main( String[] args ){
        int score = 52;
        if( score > 79.5 ){
                System.out.println("A");
        } else if( score > 69.5 ){
                System.out.println("B");
        } else if( score > 59.5 ){
                System.out.println("C");
        } else if( score > 49.5 ){
                System.out.println("D");
        } else if( score <= 49.5 ) {
                System.out.println("F");
        }
    }
}
```

2

ตัวอย่าง

การใช้ nested-if

```
public class Main {
  public static void main(String[] args) {
   int score = new Scanner(System.in).nextInt();
   char grade;
   if((score <=100) && (score>=0)){
     if(score >= 80){
       grade = 'A';
     }else if(score >= 70){
       grade = 'B';
     }else if(score >= 60){
       grade = 'C';
     }else if(score >= 50){
       grade = 'D';
     }else{
       grade = 'F';
     }
     System.out.print("คุณได้เกรด"+grade +" !!!");
   } else {
     System.out.print("คุณใส่คะแนนไม่ถูกต้อง");
   }
  }
}
```

1

2

ตัวอย่าง
การใช้หลาย if
แบบไม่ใช้ nested-if

```
public class Main {
  public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("กรุณาใส่เลขเดือน (1-12) : ");
        int month = s.nextInt();
        System.out.print("กรุณาใส่เลขวัน (1-7): ");
        int day = s.nextInt();
        String txt_month ="“, txt_day ="";

        if(month == 1)       { txt_month = "JAN" ; }
        else if(month == 2) { txt_month = "FEB" ; }
        else if(month == 3) { txt_month = "MAR" ; }
        ……
        else if(month == 12) { txt_month = "DEC" ; }

        if(day == 1)       { txt_day = "MON" ; }
        else if(day == 2) { txt_day = "TUE" ; }
        else if(day == 3) { txt_day = "WED" ; }
        ……
        else if(day == 7) { txt_day = "SUN" ; }

        System.out.print( txt_month + " , " + txt_day);
  }
}
```

Switch-case สำหรับเงื่อนไข Multiple Selection

switch-case เป็นคำสั่ง control flow ใน Java ที่อนุญาตให้ประมวลผลโค้ดหนึ่งส่วนจากหลายตัวเลือกตามค่าของตัวแปรหรือนิพจน์ เป็นอีกทางเลือกหนึ่งแทนการใช้คำสั่ง if-else if หลายรายการ

```
switch (expression) {
    case value1:
        // code block to be executed if expression equals value1
        break;
    case value2:
        // code block to be executed if expression equals value2
        break;
      // ... more cases ...
    default:
        // code block to be executed if expression doesn't match any cases
}
```

Switch-case สำหรับเงื่อนไข Multiple Selection

ส่วนประกอบสำคัญ

- **switch** คือ คีย์เวิร์ดที่ใช้กำหนดนิพจน์ในวงเล็บ นิพจน์นี้ได้รับการประมวลผลหนึ่งครั้ง
- **case** คือ คีย์เวิร์ดที่กำหนดค่าคงที่หรือค่าตัวแปรสำหรับตรวจสอบการนิพจน์ใน switch และตามด้วยเครื่องหมายทวิภาค (“:”) ค่าจะถูกเปรียบเทียบกับนิพจน์ในswitch หากตรงกัน โค้ดหลังเครื่องหมายทวิภาคจะถูกประมวลผล
- **break** คือ คีย์เวิร์ดที่ใช้ภายใน case จะออกจากคำสั่ง switch-caseหากไม่ใส่จาวาจะประมวลผลต่อไปยัง case ถัดไปจนกว่าจะพบ break หรือ switch สิ้นสุดลง
- **default** คือ บล็อกเสริมที่จะประมวลผลหากไม่เข้า case ข้างต้น ซึ่งคล้ายกับบล็อก else ในคำสั่ง if-else (มีหรือไม่มีก็ได้)

Swi หรับเงื่อนไข Multiple Selection

นิพจน์ต้องมีชนิดข้อมูลเป็น char , byte, short หรือ int เท่านั้น

```
switch (expression) {
    case value1:
        // code block to be executed if expression equals value1
        break;
    case value2:
        // code bl xpression equals value2
        break;
      // ... m
    default:
        // code block to be executed if expression doesn't match any cases
}
```

- ชนิดข้อมูลของนิพจน์และค่าที่ 1 ถึง N ต้องเป็นชนิดเดียวกัน
- ถ้าค่าของนิพจน์ตรงกับค่าใด จะทำชุดคำสั่งของค่านั้น

- ถ้าค่าของนิพจน์ไม่ตรงกับค่าใดเลย จะทำชุดคำสั่งของ default
- default จะมีหรือไม่มีก็ได้

ตัวอย่าง

การใช้ Switch-case

```
import java.util.Scanner;
public class Main {
     public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       int x = sc.nextInt();

       switch (x) {
          case 1: System.out.print("Value is one");
                  break;
          case 2: System.out.print("Value is two");
                  break;
         default: System.out.print("Other");
       }
    }
}
```

ตัวอย่าง

การใช้ Switch-case

```
public class Main {
    public static void main(String[] args) {
        int month = 2, numDays = 0;
        switch (month) {
                case 1: case 3: case 5: case 7:
                case 8: case 10: case 12:
                        numDays = 31;
                        break;
                case 4: case 6: case 9: case 11:
                        numDays = 30;
                        break;
                case 2:
                        numDays = 28;
                        break;
                default:
                        System.out.println("Invalid");
        }
        System.out.println(numDays);
    }
}
```

1

```java
public class Main {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("กรุณาใส่เลขวัน (1-7): ");
    int day = s.nextInt();
    String txt_day = "";

    if(day == 1){
      txt_day = "จันทร์";
    }else if(day == 2){
      txt_day = "อังคาร";
    }else if(day == 3){
      txt_day = "พุธ";
    }else if(day == 4){
      txt_day = "พฤหัสบดี";
    }else if(day == 5){
      txt_day = "ศุกร์";
    }else if(day == 6){
      txt_day = "เสาร์";
    }else if(day == 7){
      txt_day = "อาทิตย์";
    }
    System.out.print("สวัสดีวัน"+ txt_day);
  }
}
```

2

```java
public class Main {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("กรุณาใส่เลขวัน (1-7): ");
    int day = s.nextInt();
    String txt_day = "";

     switch(day){
      case 1:
        txt_day = "จันทร์"; break;
      case 2:
        txt_day = "อังคาร"; break;
      case 3:
        txt_day = "พุธ"; break;
      case 4:
        txt_day = "พฤหัสบดี"; break;
      case 5:
        txt_day = "ศุกร์"; break;
      case 6:
        txt_day = "เสาร์"; break;
      case 7:
        txt_day = "อาทิตย์"; break;
    }
    System.out.print("สวัสดีวัน"+ txt_day);
  }
}
```

THANKS !