Faculty of
Information Technology
King Mongkut's Institute of Technology Ladkrabang

บทที่ 3 คำสั่งทำงานซ้ำ, ขอบเขต และการแปลงชนิดข้อมูล

บรรยายโดย ผศ.ดร.ธราวิเชษฐ์ ธิติจรูญโรจน์
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

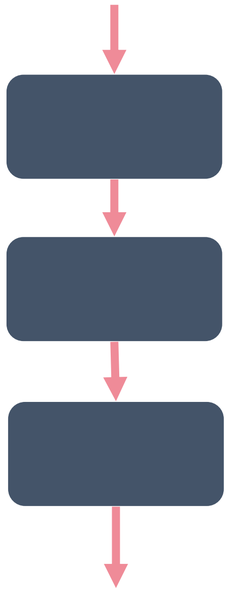
หัวข้อ

- คำสั่งทำงานซ้ำ
- ขอบเขต
- การแปลงชนิดข้อมูล

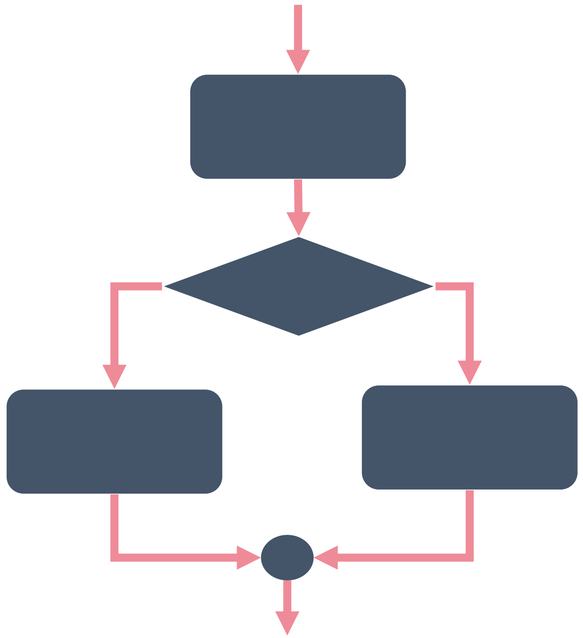
หัวข้อ

- คำสั่งทำงานซ้ำ
- ขอบเขต
- การแปลงชนิดข้อมูล

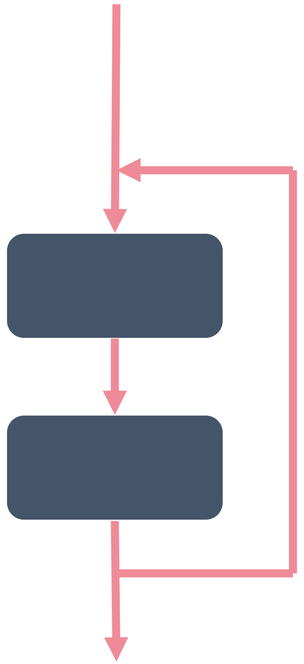
โครงสร้างการเขียนโปรแกรม



การทำงานแบบเรียงลำดับ
(Sequence)



การทำงานแบบมีเงื่อนไข
(Selection)

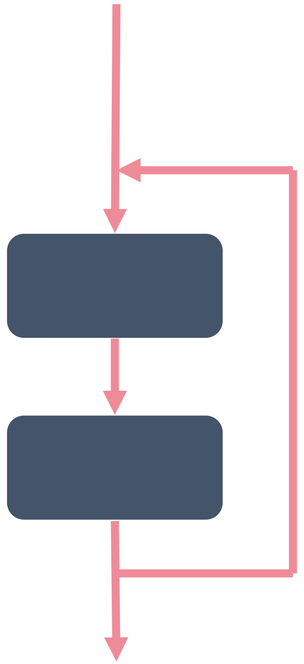


การทำงานแบบทำซ้ำ
(Iteration)

โครงสร้างการเขียนโปรแกรม

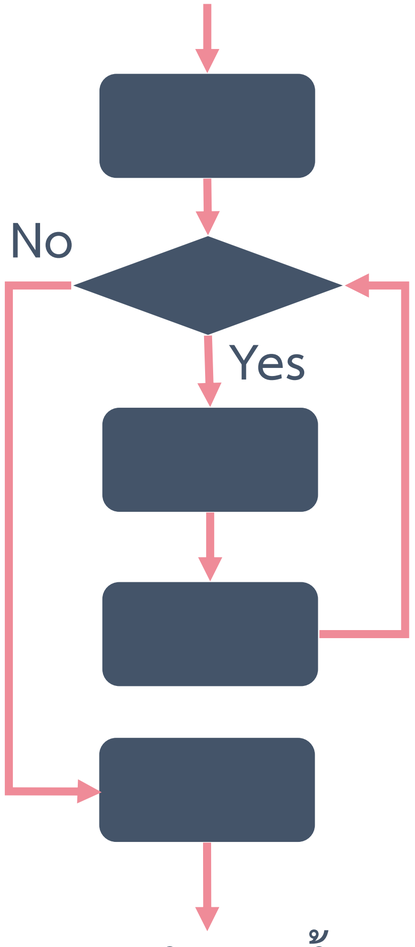
การทำงานแบบเรียงลำดับ
(Sequence)

การทำงานแบบมีเงื่อนไข
(Selection)



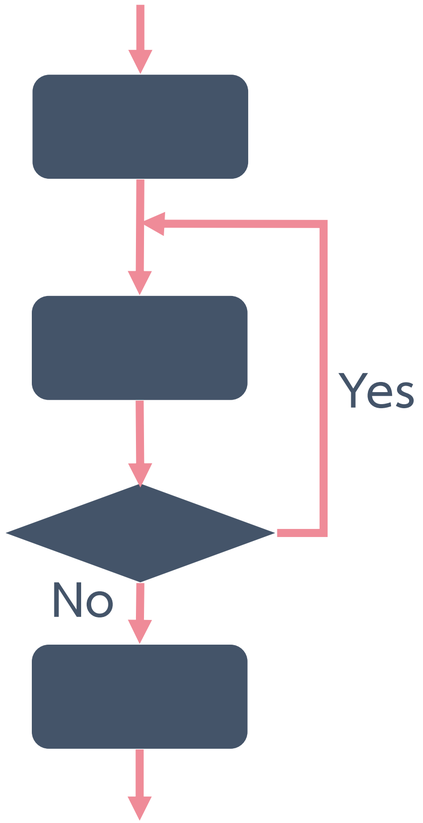
การทำงานแบบทำซ้ำ
(Iteration)

การทำงานแบบทำซ้ำ(Iteration)



การทำงานซ้ำ

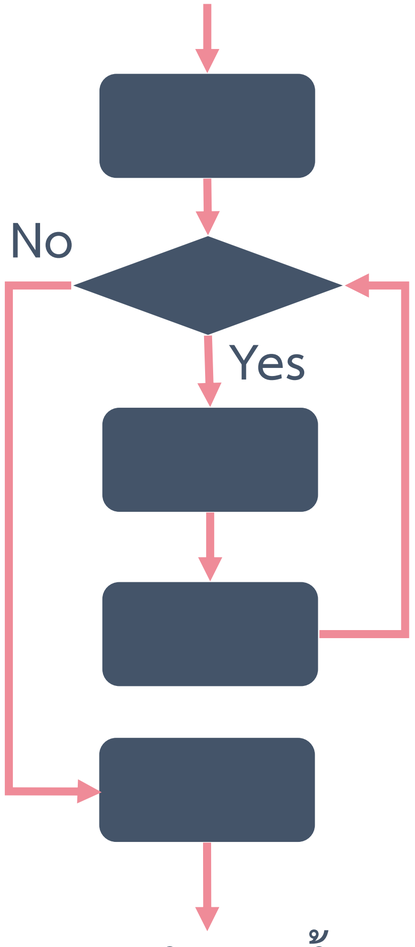
แบบ<u>ตรวจสอบเงื่อนไขก่อน</u>



การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขหลัง</u>

การทำงานแบบทำซ้ำ(Iteration)



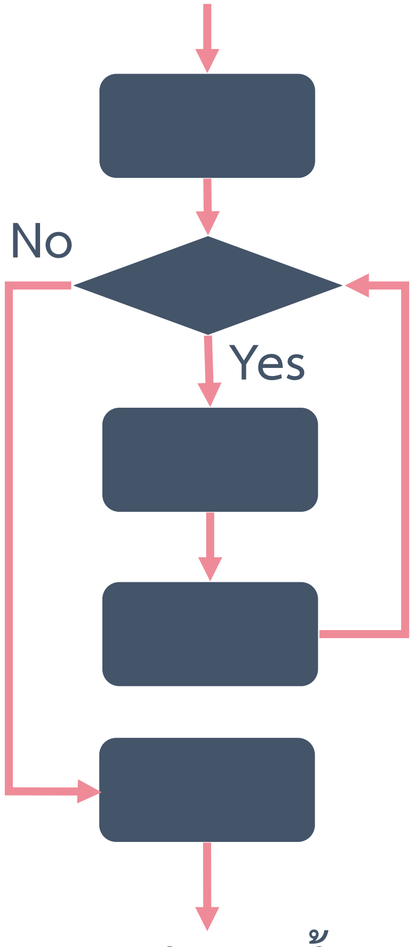
การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขก่อน</u>

การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขหลัง</u>

การทำงานแบบทำซ้ำ(Iteration)



รู้จำนวนรอบ

ไม่รู้จำนวนรอบ

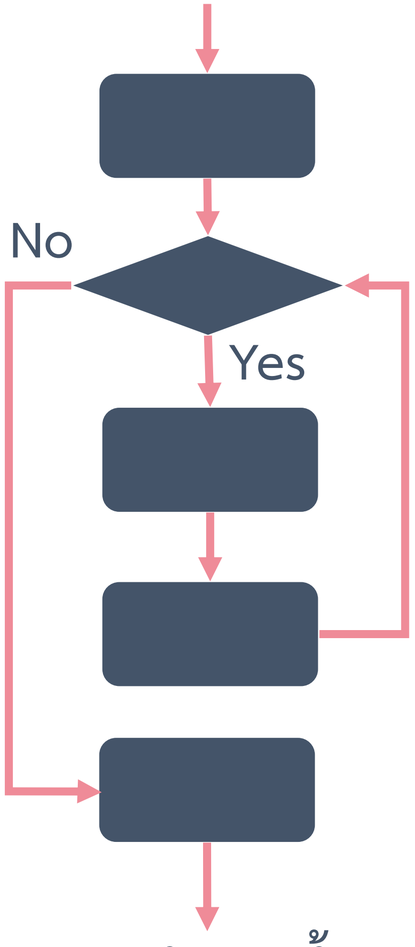
การทำงานซ้ำ

แบบตรวจสอบเงื่อนไขก่อน

การทำงานซ้ำ

แบบตรวจสอบเงื่อนไขหลัง

การทำงานแบบทำซ้ำ(Iteration)



รู้จำนวนรอบ

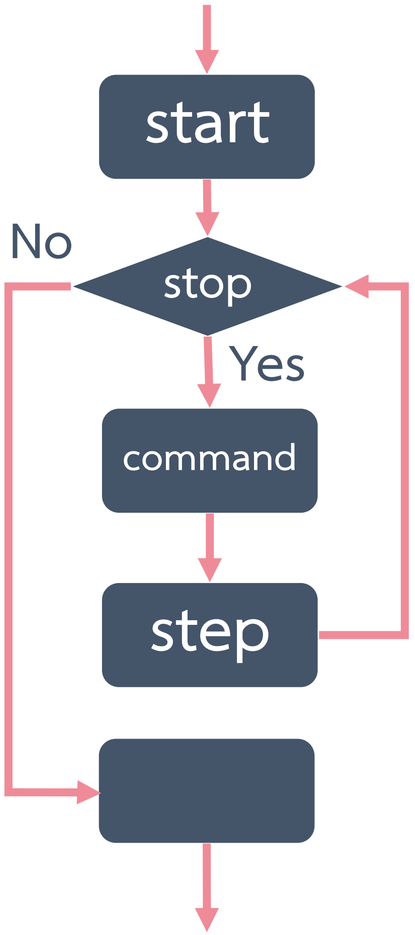
การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขก่อน</u>

การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขหลัง</u>

การทำงานแบบทำซ้ำ(Iteration)



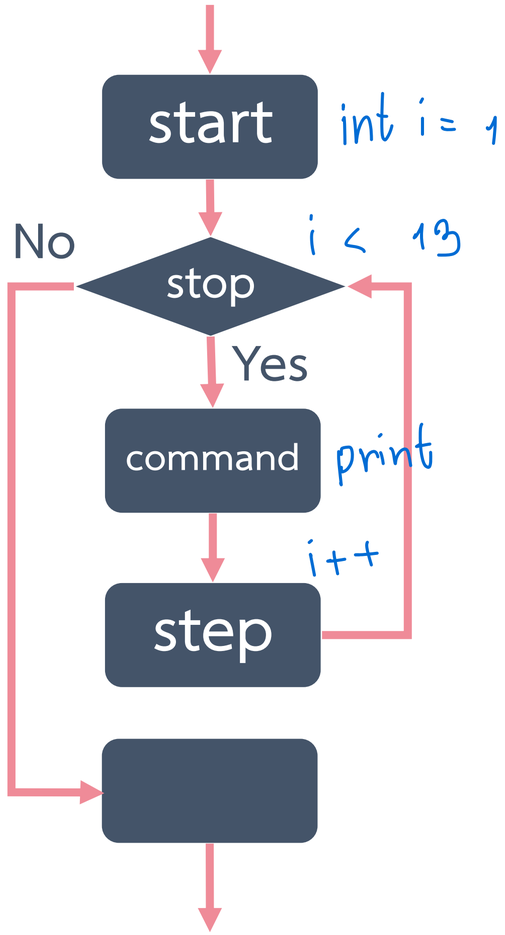
optional optional

นิพจน์ เงื่อนไข นิพจน์

```
for ( start;    stop;    step)
{
        // command
}
```

void boolean void

การทำงานแบบทำซ้ำ(Iteration)



ตัวอย่างการทำงานซ้ำแบบรู้จำนวนรอบและใช้ประโยชน์จาก running number

```
for (int i = 1; i < 13; i++){
      System.out.println("2 x "+ i + " = " + (2*i) );
}
```

ตัวอย่างการทำงานซ้ำแบบรู้จำนวนรอบและไม่ใช้ประโยชน์จาก running number

```
double balance = 100;
for (int i = -5; i < 5; i++) {
      balance *= 1.05;
}
System.out.println(balance);
```

การทำงานแบบทำซ้ำ(Iteration)

ตัวอย่าง

```
for (int i = 0; i < 5; i++){
     System.out.println(i);
}
```

นอกจากนี้ การกำหนดค่าเริ่มต้นและการเปลี่ยนแปลงค่าอาจมีมากกว่าอย่างละ 1 คำสั่งได้ ดังนั้น นักศึกษาอาศัยเครื่องหมาย ',' ในการแยกคำสั่ง

```
for (int i=0, j=0; i<4 & j < 10; i++, j+=2) {
          ...
}
```

นอกจากนี้ ขอบเขตของตัวแปรที่ประกาศในคำสั่งกำหนดค่าจะใช้ได้เฉพาะภายในบล็อกคำสั่ง for เท่านั้น

แบบฝึกหัด

```
public class Main {
    public static void main(String[] args) {
        for ( int x = __(A)__;  __(B)__; __(C)__ ){
                ....
        }
    }
}
```

x = x + ▢ x = x - ▢

x += ▢ x -= ▢

x++, ++x x--, --x

จงเขียนผลลัพธ์ลงในตารางต่อไปนี้

ค่า x ที่คาดหวัง	(A)	(B)	(C)
1, 4, 7, 10, 13, 16, 19	int x = 1	x < 20 or x <= 19	x += 3
-100, -90, -80, -70, -60, -50, -40	int x = -100	x <= -40	x += 10
-10, -5, 0, 5, 10, 15, 20	int x = -10	x < 21 ; x <= 20	x += 5

แบบฝึกหัด

```
public class Main {
    public static void main(String[] args) {
            for ( int x = 3 ;  x < 7 ; x++ ){
                System.out.print(x+2);
            }
    }
}
```

จงเขียนผลลัพธ์ลงในตารางต่อไปนี้

รอบที่	ค่าตัวแปร x	System.out.print(x+2);
1	3	5
2	4	6
3	5	7
4	6	8

แบบฝึกหัด

```
public class Main {
    public static void main(String[] args) {
            for ( int x = 10 ;  x > 5 ; x-=2 ){
                System.out.print("a-" + x);
            }
    }
}
```

จงเขียนผลลัพธ์ลงในตารางต่อไปนี้

รอบที่	ค่าตัวแปร x	System.out.print("a-" + x);
1	10	a-10
2	8	a-8
3	6	a-6

การทำงานแบบทำซ้ำ (Iteration)

```
public class VariableScope {
 public static void main(String args[]) {
    for (int i=1; i<10; i++) {
        System.out.print(i+"  ");
    }
    System.out.println("i = "+i);   //illegal
  }
}
```

โครงสร้างแบบซ้อน (Nested Structure)

เราสามารถที่จะเขียนคำสั่งโครงสร้างควบคุมใด ๆ ซ้อนอยู่ภายในได้ ซึ่งโครงสร้างควบคุมภายในและภายนอกไม่จำเป็นต้องเป็นคำสั่งชนิดเดียวกัน

ตัวอย่าง เช่น การเขียนโครงสร้างทำซ้ำแบบซ้อน (for อยู่ใน for)

```
public class NestedFor {
  public static void main(String args[]) {
      for (int i=1; i<=3; i++) {
        for (int j=1; j<=5; j++) {
           System.out.print('*');
        }
        System.out.println();
      }
  }
}
```

โครงสร้างแบบซ้อน (Nested Structure)

เราสามารถที่จะเขียนคำสั่งโครงสร้างควบคุมใด ๆ ซ้อนอยู่ภายในได้ ซึ่งโครงสร้างควบคุมภายในและภายนอกไม่จำเป็นต้องเป็นคำสั่งชนิดเดียวกัน

ตัวอย่าง เช่น การเขียนโครงสร้างทำซ้ำแบบซ้อน (for อยู่ใน for)

```
public class NestedFor {
  public static void main(String args[]) {
      for (int i=1; i<=3; i++) {
        for (int j=1; j<=5; j++) {
           System.out.print('*');
        }
        System.out.println();
      }
  }
}
```

ผลลัพธ์

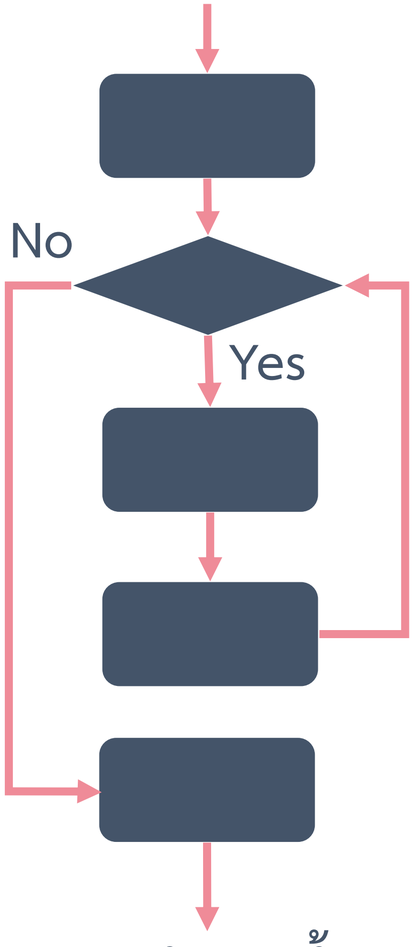
*****

*****

*****

โดยที่ **for** ที่อยู่ภายใน จะพิมพ์เครื่องหมาย ‘*’ เท่ากับจำนวนคอลัมน์ (column) และ **for** ที่อยู่ภายนอก จะทำคำสั่ง **for** ภายในเท่ากับจำนวนแถว (row)

การทำงานแบบทำซ้ำ(Iteration)



ไม่รู้จำนวนรอบ

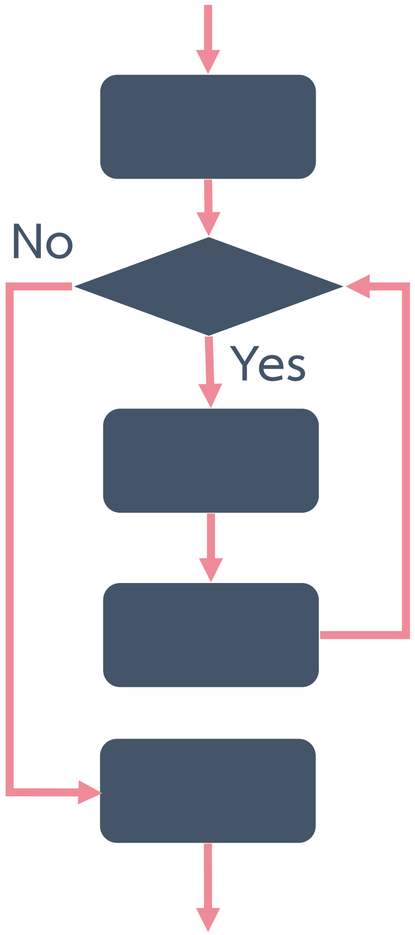
การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขก่อน</u>

การทำงานซ้ำ

แบบ<u>ตรวจสอบเงื่อนไขหลัง</u>

การทำงานแบบทำซ้ำ(Iteration)




```
while (เงื่อนไข) {
        ประโยคที่ทำซ้ำขณะที่เงื่อนไขเป็นจริง;
}
```

```
int money = sc.nextInt();
while (money <= 10000) {
        money += sc.nextInt();
}
```

for VS while

- รู้จำนวนรอบ

```
int num = 1;
while (num <= 10) {
        System.out.println(num);
        num++;
}
```

start
stop
step

```
for (int n = 1; n <= 10; n++) {
        System.out.println(n);
}
```

for VS while

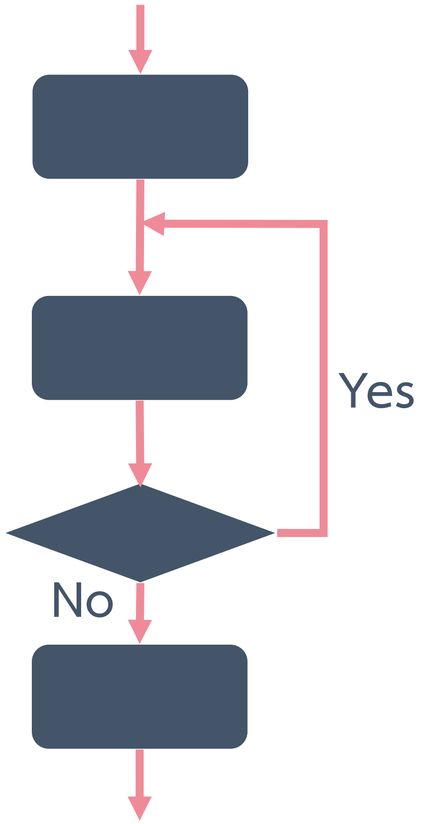
- ไม่รู้จำนวนรอบ

```
int money = sc.nextInt();
while (money <= 10000) {
        money += sc.nextInt();
}
```

```
int money = sc.nextInt();
for (; money <= 10000;) {
        money += sc.nextInt();
}
```

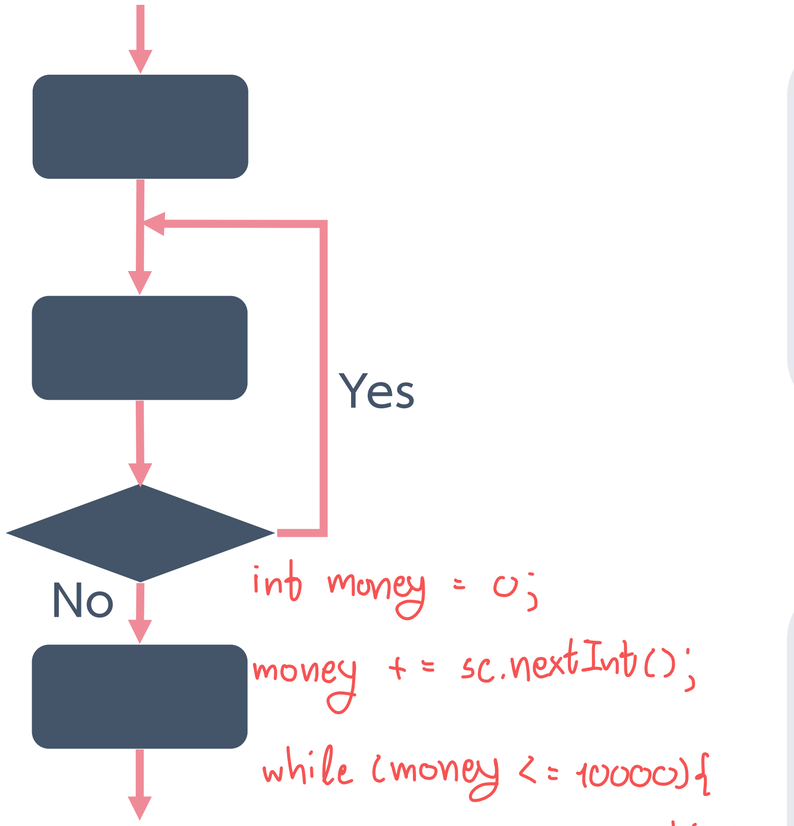
การทำงานแบบทำซ้ำ(Iteration)

การทำงานซ้ำ
แบบตรวจสอบเงื่อนไขก่อน



การทำงานซ้ำ
แบบตรวจสอบเงื่อนไขหลัง

การทำงานแบบทำซ้ำ(Iteration)




```
int money = 0;
money += sc.nextInt();
while (money <= 10000){
    money += sc.nextInt();
}
```

```
do {
        // command;
} while ( condition );
```

```
int money = 0;
do{
  money += sc.nextInt();
}while (money <= 10000);
```

คำสั่ง break และ continue

- คำสั่ง `break` จะทำให้**หยุดสิ้นสุด**การทำงานของโครงสร้างแบบทำซ้ำ
- คำสั่ง `continue` จะ**ข้ามการทำงานคำสั่งที่เหลือ**ภายในบล็อก { } โดยไปเริ่มการทำซ้ำในรอบต่อไปใหม่

ตัวอย่างเช่น

```
for (int i = 0; i < 5; i++) {
  System.out.print("<");
  if (i == 2)
      break;
   System.out.print(i + ">");
}
```

<0><1><.

```
for (int i = 0; i < 5; i++) {
  System.out.print("<");
  if (i == 2)
      continue;
   System.out.print(i + ">");
}
```

<0><1><3><4>.

คำสั่งอื่น ๆ ในการควบคุม loop

- **`label : statements;`**
  - label เป็นการระบุตำแหน่งของ loop กรณีที่มี loop ซ้อนกัน
- **`break [label];`**
  - คำสั่งให้สิ้นสุดการทำงานใน loop
- **`continue [label];`**
  - คำสั่งให้ข้ามการทำงานของคำสั่งที่เหลือทั้งหมดใน loop

ตัวอย่างโปรแกรม

```
public class SampleBreak2 {
     public static void main(String args[]) {
          int i, j, product;
 outer:    for (i=1; i<=3; i++) {
                for (j=1; j<=3; j++){
                  product = i*j;
                  if (j==3) break outer;
                  System.out.println(i+" * "+j+" = "+product);
                }
            }
           System.out.println("Outside nested loops.");
        }
}
```

ผลลัพธ์ที่ได้จากการรันโปรแกรม

```
1 * 1 = 1
1 * 2 = 2
Outside nested loops.
```

วิเคราะห์โปรแกรมต่อไปนี้

(กรณี ใช้ for-loop มากกว่า 1 ตำแหน่งในโปรแกรม)

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("กรุณาใส่ความสูงของชั้นที่ 1 : ");
    int t1 = s.nextInt();
    System.out.print("กรุณาใส่ความสูงของชั้นที่ 2 : ");
    int t2 = s.nextInt();

    System.out.println("  #");
    System.out.println(" ##");
    System.out.println("###");

    for(int x = 1; x <= t1;x++){
      System.out.println("=A=");
    }

    System.out.println("###");
    System.out.println("###");
    System.out.println("###");

    for(int x = 1; x <= t2;x++){
      System.out.println("=B=");
    }

    System.out.println("###");
    System.out.println(" ##");
    System.out.println("  #");
}
```

fix ① ② fix ③ ④ fix ⑤ ⑥

```
กรุณาใส่ความสูงของชั้นที่ 1 : 3
กรุณาใส่ความสูงของชั้นที่ 2 : 6
  #
 ##
###
=A=
=A=
=A=
###
###
###
=B=
=B=
=B=
=B=
=B=
=B=
###
 ##
  #
```

```
กรุณาใส่ความสูงของชั้นที่ 1 : 4
กรุณาใส่ความสูงของชั้นที่ 2 : 2
  #
 ##
###
=A=
=A=
=A=
=A=
###
###
###
=B=
=B=
###
 ##
  #
```

ตัวอย่างโปรแกรม

```
public class Main {
    public static void main(String[] args) {
            for (int x = 1; x <=4; x++) {
                  if (x%2 == 0) {
                        System.out.println(x + "  is even.");
                  } else {
                        System.out.println(x + " is odd.");
                  }
            }
    }
}
```

หัวข้อ

- คำสั่งทำงานซ้ำ
- **ขอบเขต**
- การแปลงชนิดข้อมูล

ขอบเขต

การระบุขอบเขตในภาษาจาวาจะพิจารณาจาก 3 ตัวดำเนิน ได้แก่ semicolon ; วงเล็บ (...) และวงเล็บปีกกา {....}

ขอบเขตของ statement

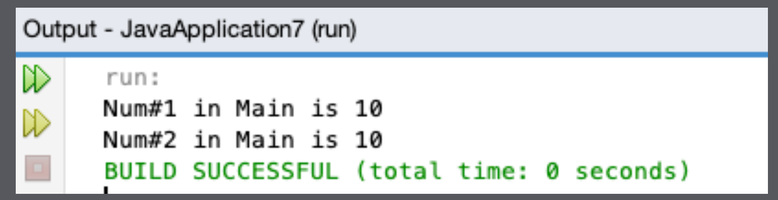
ขอบเขตของ method

ขอบเขตของ object

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

      int a = 10;

      System.out.println("line 1: "+a);

   }
}
```

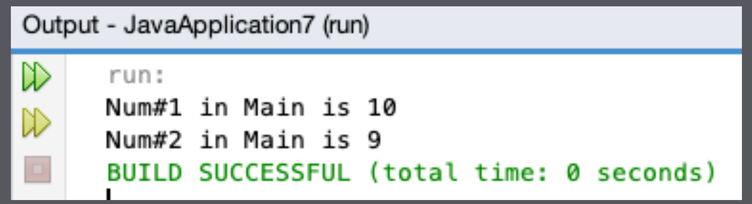
ขอบเขต

```
public class Main {
      public static int num = 10;
      public static void main(String[] args) {
            System.out.println("Num#1 in Main is "+num);
            System.out.println("Num#2 in Main is "+num);
      }
}
```




```
public class Main {
      public static int num = 10;
      public static void main(String[] args) {
            System.out.println("Num#1 in Main is "+num);
            int num = 9;
            System.out.println("Num#2 in Main is "+num);
      }
}
```

ใน { } 1 คู่สามารถประกาศตัวแปรที่ชื่อซ้ำกันได้เพียง 1 ตัว



ขอบเขต

```
public class Main {
     public static int num = 10;
     public static void main(String[] args) {

               System.out.println("Num#1 in Main is "+num);

               int num = 9;
               System.out.println("Num#2 in Main is "+num);

               num += (9+3*7);
               System.out.println("Num#3 in Main is "+num);
     }
}
```

A

B

1

2

3

Num #1 in Main is 10 → A

Num #2 in Main is 9 → B

Num #3 in Main is 39 → B

ขอบเขต

```
public class Main {
   public static void main(String[] args) {
            int a = 1;
            if ( a == 1 ){
                int b = 20;
                System.out.println("Var A is " + a);
                System.out.println("Var B is " + b);
            }
            System.out.println("Var A is " + a);
            System.out.println("Var B is " + b);
      }
}
```

1

2

3

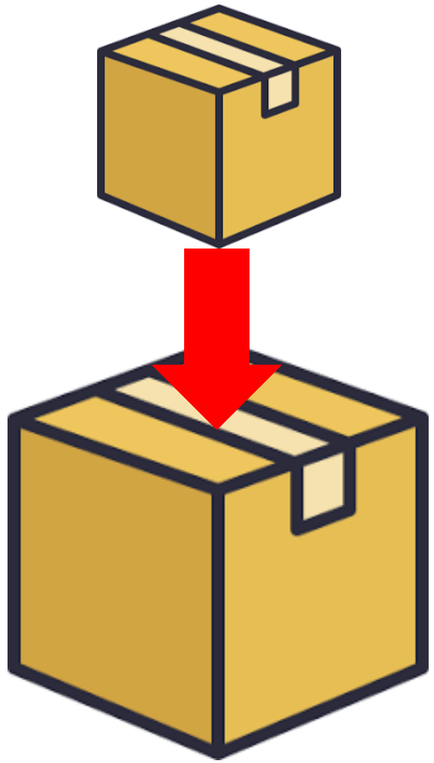
4

Var A is 1
Var B is 20
Var A is 1
error

หัวข้อ

- คำสั่งทำงานซ้ำ
- ขอบเขต
- การแปลงชนิดข้อมูล

การแปลงชนิดข้อมูล



วิธีที่ 1 การแปลงผ่านเครื่องหมาย =

แปลงข้อมูลเป็นขนาดใหญ่ขึ้นเท่านั้นถ้าแปลงเป็นขนาดเล็กกว่าจะเกิดข้อผิดพลาดขณะคอมไพล์

วิธีที่ 2 การแปลงเมื่อเกิดการดำเนินการ

เมื่อ x * y โดย x เป็น double และ y เป็น int ซึ่ง y จะถูกเปลี่ยนเป็น double ก่อนดำเนินการโดยอัตโนมัติ

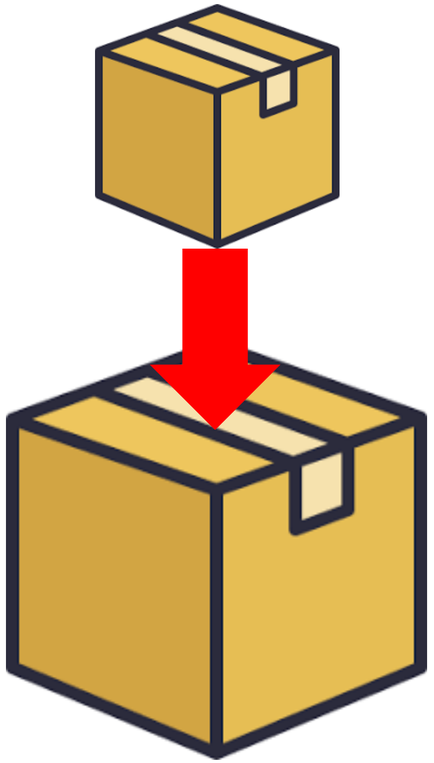
วิธีที่ 3 การแปลงผ่านการทำ Casting

ลักษณะการใช้งาน เมื่อ x เป็น int และ y เป็น double เช่น **`int x = (int) y;`**

วิธีที่ 4 การใช้เมธอดจากคลาส Wrapper

ใช้เมธอดที่มาพร้อมคลาส เช่น parseInt หรือ parseBoolean เพื่อแปลงข้อมูลชนิดใด ๆ ก็ได้มาเป็นข้อมูลชนิด int หรือ boolean ตามลำดับ เป็นต้น

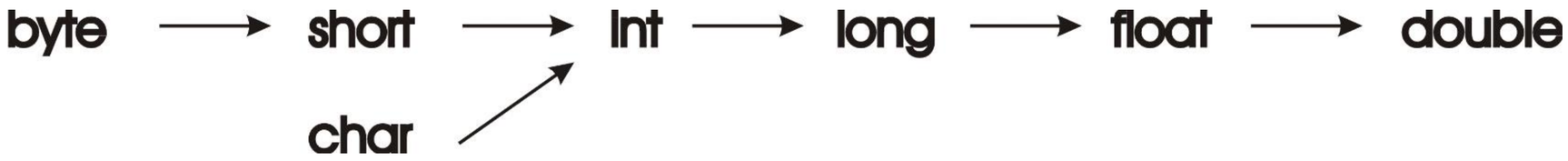
การแปลงชนิดข้อมูล



- ชนิดข้อมูลตัวเลขจำนวนเต็มสามารถแปลงให้เป็นชนิดข้อมูลตัวเลขทศนิยมได้
- ชนิดข้อมูล `float` สามารถแปลงให้เป็นชนิดข้อมูล `double` ได้
- ชนิดข้อมูลตัวเลขจำนวนเต็มมีขนาดเรียงกันจากน้อยไปมากดังนี้

`byte` → `short` → `int` → `long`

- ชนิดข้อมูล `char` สามารถแปลงให้เป็นชนิดข้อมูล `int` ได้
- ชนิดข้อมูล `boolean` จะไม่มีความสัมพันธ์กับชนิดข้อมูลแบบพื้นฐานอื่นๆ



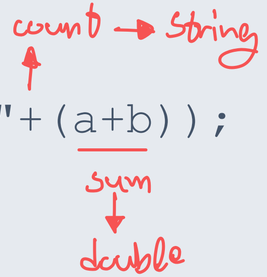
การแปลงชนิดข้อมูล

- วิธีที่ 1 การแปลงผ่านเครื่องหมาย =

```
public class App03 {
    public static void main(String[] args) {
        int a = 10;
        double b = a;
    }
}
```

- วิธีที่ 2 การแปลงเมื่อเกิดการดำเนินการ

```
public class App04 {
    public static void main(String[] args) {
        int a = 10;
        double b = 15.0;
        System.out.println("line 1: "+(a+b));
    }
}
```



การแปลงชนิดข้อมูล

- วิธีที่ 3 การแปลงผ่านการทำ Casting (ตัดข้อมูลออก)

```
public class App05 {
    public static void main(String[] args) {
        double b = 15.7;
        int a = (int) b;
    }
}
```

- วิธีที่ 4 การใช้เมธอดจากคลาส Wrapper

```
public class App06 {
    public static void main(String[] args) {
        String str = "1506";
         int  num = Integer.parseInt(str);
        System.out.println(num);
    }
}
```

SIMPLE TYPE	WRAPPER CLASSES
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long

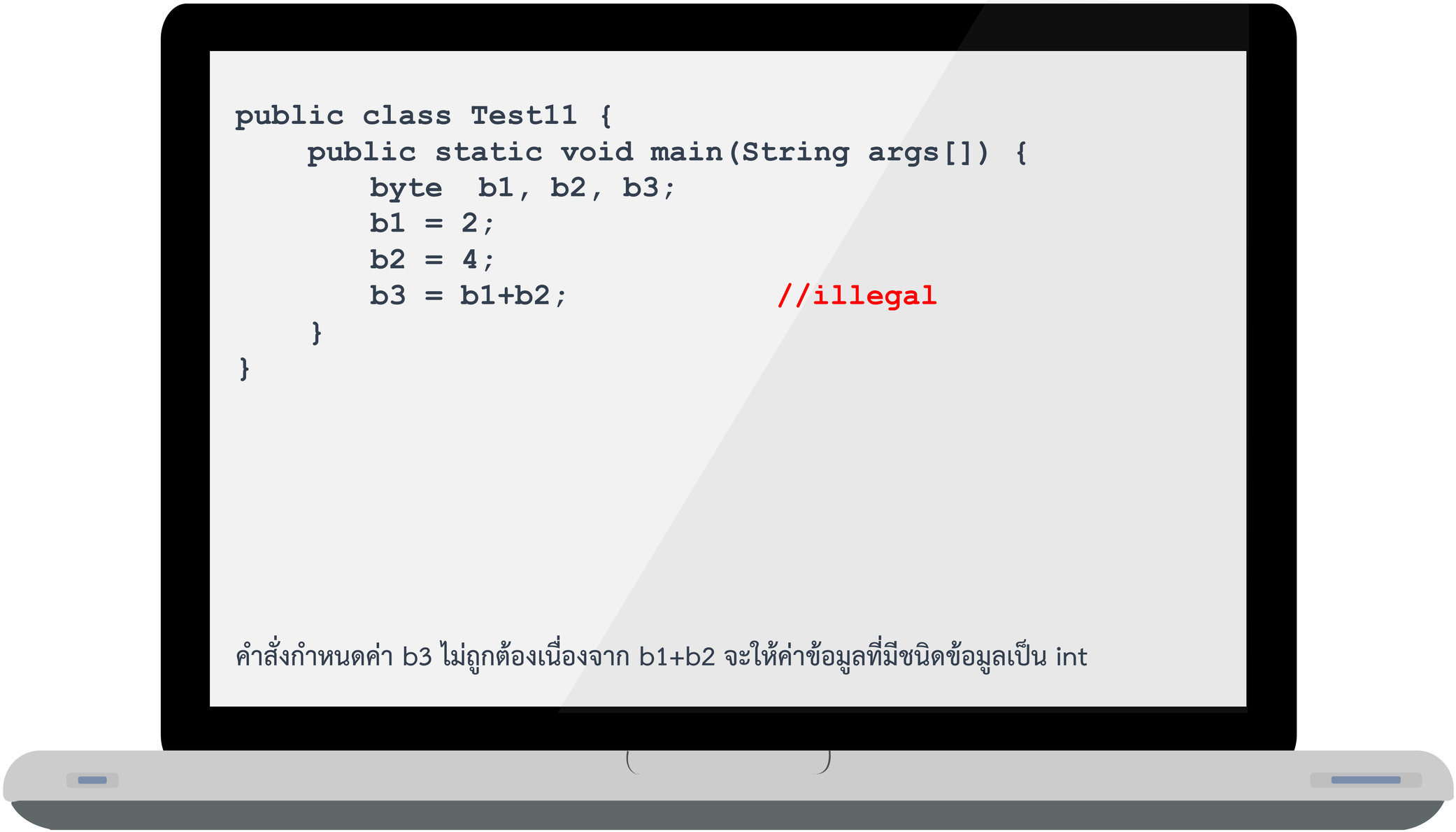
การแปลงชนิดข้อมูล

```
public class Main {
    public static void main(String[] args) {
        // Convert int to String
        int i = 10;
        String str1 = String.valueOf(i);
        String str2 = i + "";

        // Convert String to int
        int num1 = Integer.parseInt("200");

        // Convert double to String
        double num2 = 12.3;
        String str3 = String.valueOf(num2);

        // Convert String to double
        double num3 = Double.parseDouble("23.6");
    }
}
```

String → int
"200" → 200

String → double
"23.6" → 23.6

ตัวอย่างโปรแกรม



```
public class Test11 {
    public static void main(String args[]) {
        byte  b1, b2, b3;
        b1 = 2;
        b2 = 4;
        b3 = b1+b2;              //illegal
    }
}
```

คำสั่งกำหนดค่า b3 ไม่ถูกต้องเนื่องจาก b1+b2 จะให้ค่าข้อมูลที่มีชนิดข้อมูลเป็น int

ตัวอย่างของการแปลงชนิดข้อมูล

ภาษาจาวาจะปรับชนิดข้อมูลให้อัตโนมัติ ในกรณีต่อไปนี้

- กำหนดค่าชนิด*ข้อมูลที่เล็กกว่า*ให้กับตัวแปรชนิดข้อมูลที่ใหญ่กว่าอาทิเช่น

```
int i = 4;
long l = i;
```

โดยที่ นิพจน์ i จะถูกปรับชนิดข้อมูลจาก int ให้เป็น long โดยอัตโนมัติ

- กำหนดค่าชนิดข้อมูล*จำนวนเต็มให้กับจำนวนเลขทศนิยม* อาทิเช่น

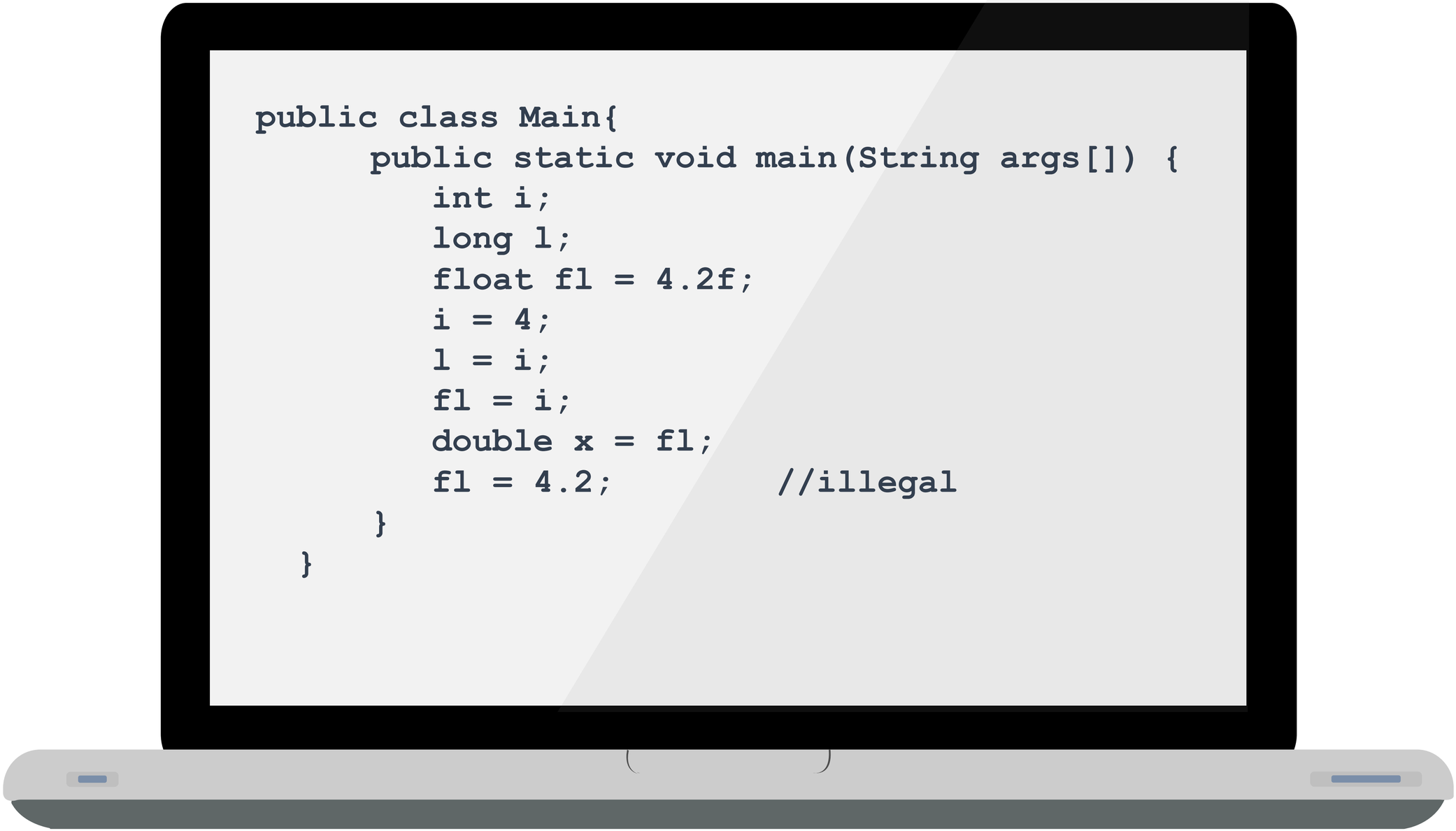
```
double x = 3;
```

โดยที่ นิพจน์ 3 จะถูกปรับชนิดข้อมูลจาก int ให้เป็น double โดยอัตโนมัติ

- ตัวอย่างที่ไม่ถูกต้อง

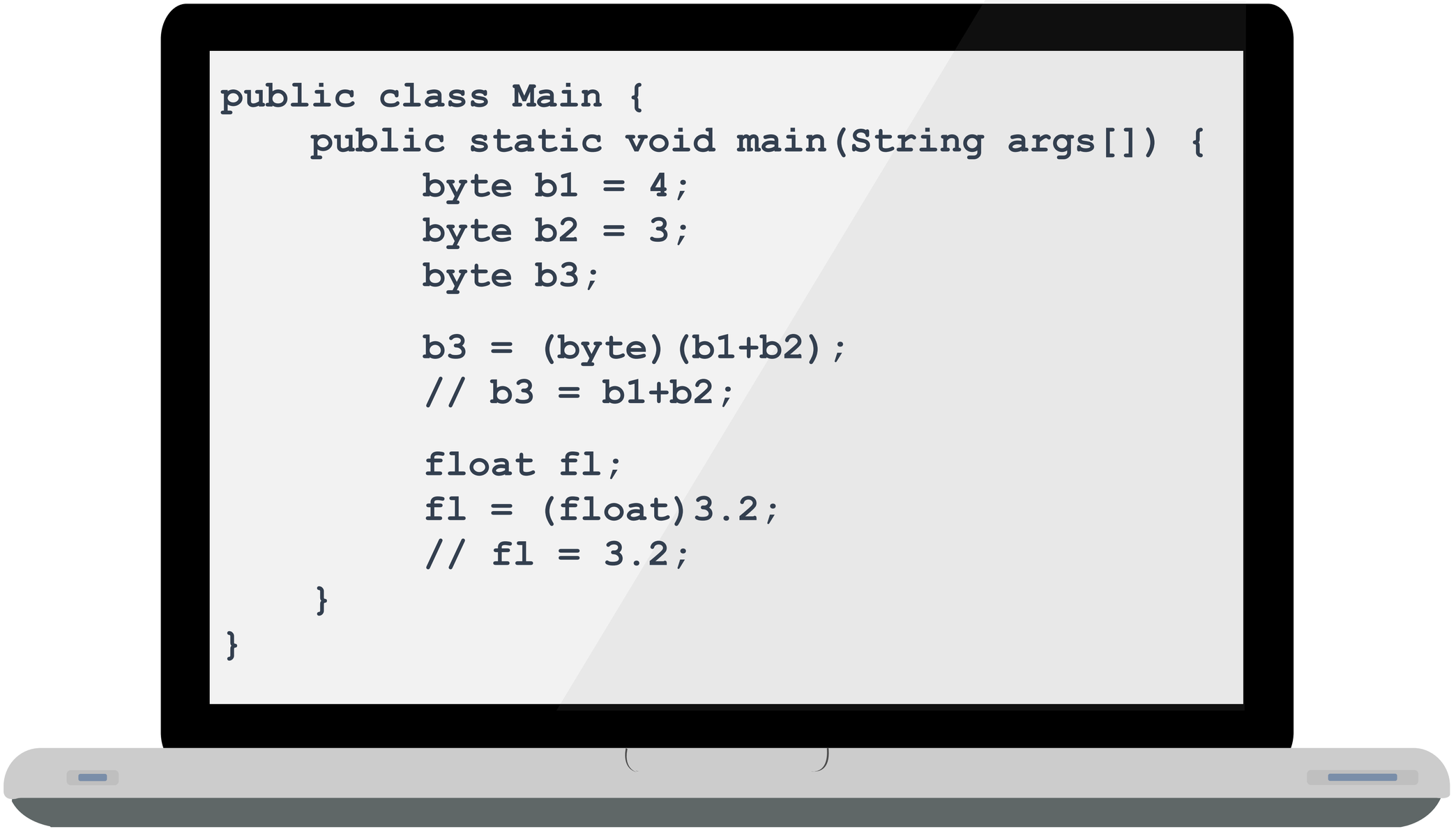
```
int amount = 123L;   //illegal
float f = 4.0;       //illegal
```

ตัวอย่างโปรแกรม



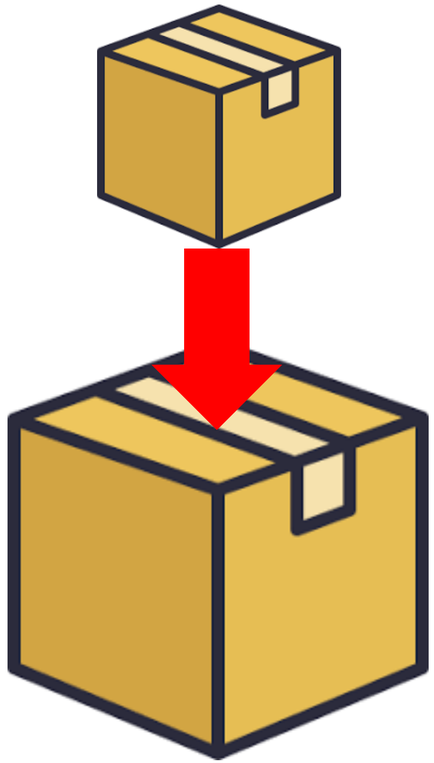
```
public class Main{
      public static void main(String args[]) {
          int i;
          long l;
          float fl = 4.2f;
          i = 4;
          l = i;
          fl = i;
          double x = fl;
          fl = 4.2;          //illegal
      }
  }
```

ตัวอย่างโปรแกรม



```
public class Main {
    public static void main(String args[]) {
        byte b1 = 4;
        byte b2 = 3;
        byte b3;

        b3 = (byte)(b1+b2);
        // b3 = b1+b2;

        float f1;
        f1 = (float)3.2;
        // f1 = 3.2;
    }
}
```

การแปลงชนิดข้อมูล



วิธีที่ 1 การแปลงผ่านเครื่องหมาย =

แปลงข้อมูลเป็นขนาดใหญ่ขึ้นเท่านั้นถ้าแปลงเป็นขนาดเล็กกว่าจะเกิดข้อผิดพลาดขณะคอมไพล์

วิธีที่ 2 การแปลงเมื่อเกิดการดำเนินการ

เมื่อ x * y โดย x เป็น double และ y เป็น int ซึ่ง y จะถูกเปลี่ยนเป็น double ก่อนดำเนินการโดยอัตโนมัติ

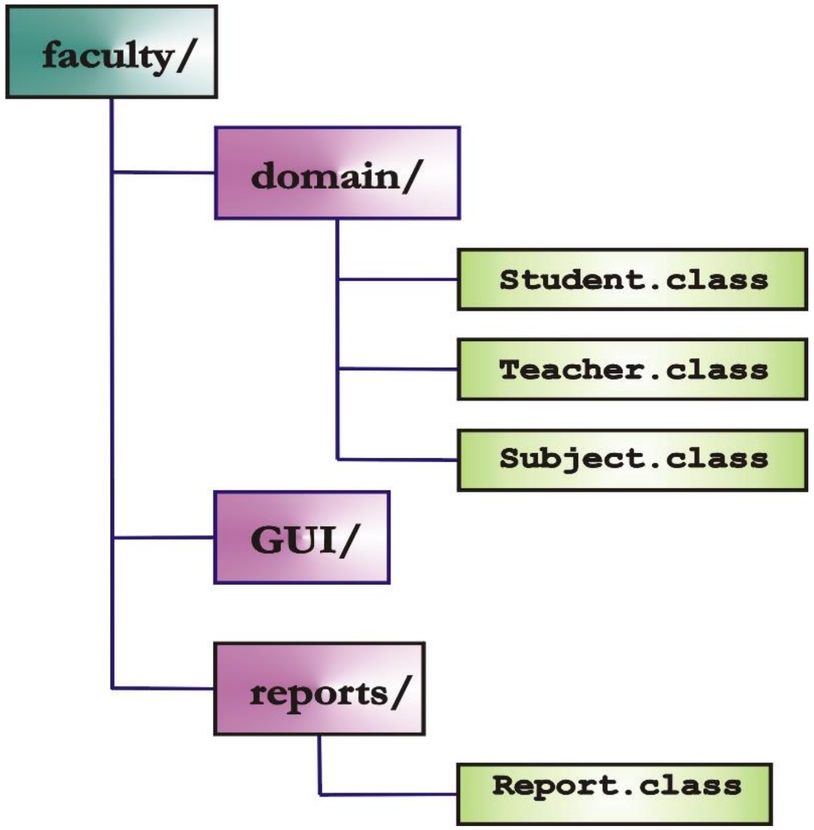
วิธีที่ 3 การแปลงผ่านการทำ Casting

ลักษณะการใช้งาน เมื่อ x เป็น int และ y เป็น double เช่น **`int x = (int) y;`**

วิธีที่ 4 การใช้เมธอดจากคลาส Wrapper

ใช้เมธอดที่มาพร้อมคลาส เช่น parseInt หรือ parseBoolean เพื่อแปลงข้อมูลชนิดใด ๆ ก็ได้มาเป็นข้อมูลชนิด int หรือ boolean ตามลำดับ เป็นต้น

แพคเก็จ (Package)



- ซอฟต์แวร์แพคเก็จช่วยในการจัดการการพัฒนาโปรแกรมขนาดใหญ่
- ในโปรแกรมภาษาจาวา แพคเก็จจะเป็นที่รวมของคลาสของภาษาจาวาหลาย ๆ คลาส
- โปรแกรมอาจแบ่งเป็นแพคเก็จและแพคเก็จย่อย (Subpackage)
- แพคเก็จจะเก็บไว้ในไดเร็กทอรี่ (Directory) ซึ่งจะเป็นชื่อของแพคเก็จ

โครงสร้างโปรแกรมภาษาจาวา

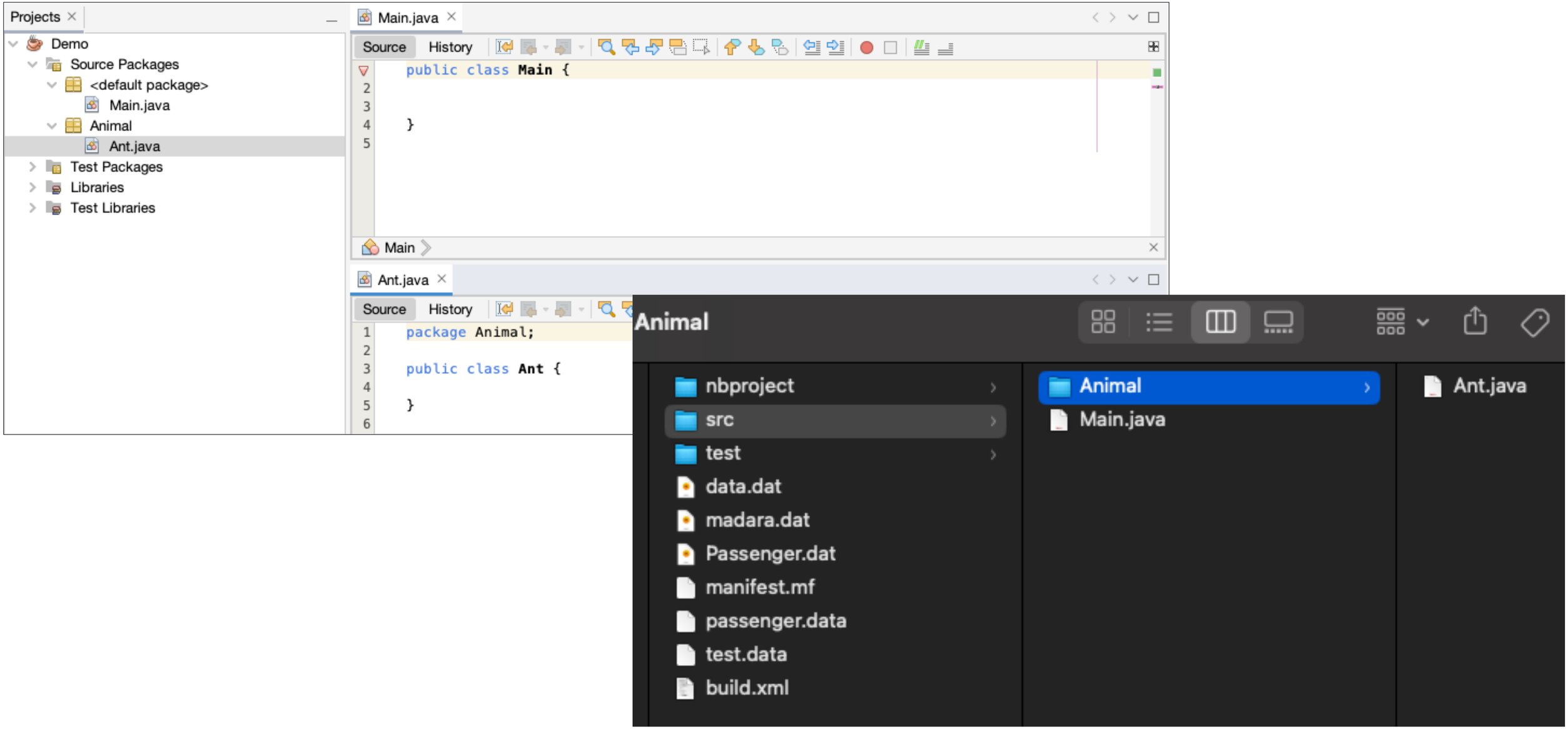
รูปแบบโปรแกรมภาษาจาวามีดังนี้

```
package_declaration
```

```
import_declaration
```

```
class_declaration
```

โครงสร้างโปรแกรมภาษาจาวา



คำสั่ง package

คำสั่ง **package** เป็นการระบุว่าคลาสอยู่ในแพคเก็จใด โดยมีรูปแบบของคำสั่ง **package ดังนี้**

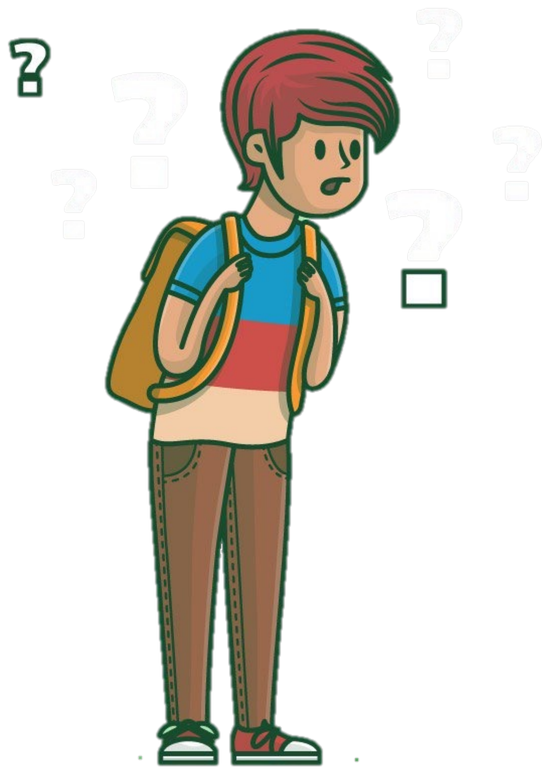
```
package <package_name>[<sub_package_name>];
```

ตัวอย่าง

```
package faculty.domain;
```

ซึ่งโปรแกรมภาษาจาวาหนึ่งโปรแกรมจะมีคำสั่ง **package** ได้เพียงคำสั่งเดียว โดยจะเป็นคำสั่งแรกของโปรแกรม และกรณีที่ไม่มีคำสั่ง **package** คลาสจะถูกกำหนดไว้ในแพคเก็จ **default**

คำสั่ง import



- คำสั่ง import เป็นการเรียกใช้คลาสในแพคเก็จต่าง ๆ
- รูปแบบของคำสั่ง import

```
import <package_name>[.<sub_package_name>].<Class_name>
```

  - หรือ

```
import <package_name>[.<sub_package_name>].*;
```

- ตัวอย่าง

```
import faculty.reports.Report;
```

  - หรือ

```
import java.awt.*;
```

- คำสั่ง import จะอยู่ก่อนหน้าการประกาศคลาส
- โปรแกรมภาษาจาวาหนึ่งโปรแกรมสามารถมีคำสั่ง import ได้หลายคำสั่ง