2110101 : การทำโปรแกรมคอมพิวเตอร์

Iteration : while, do-while, for

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

- คำสั่ง do-while
- คำสั่ง while
- คำสั่ง for
- การอ่านแฟ้มข้อความแบบลำดับ
- คำสั่ง break และ continue

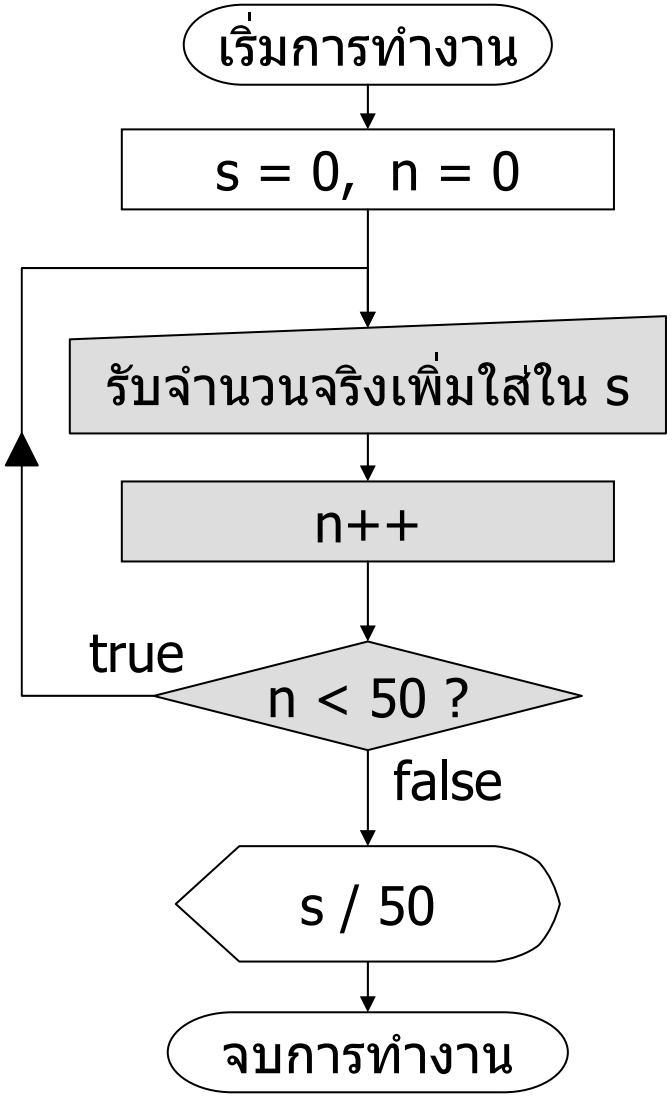
โปรแกรมหาค่าเฉลี่ย

```
import jlab.JLabIO;

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

ถ้าต้องการหาค่าเฉลี่ยของข้อมูล 50 จำนวน จะเขียนอย่างไรสั้นๆ

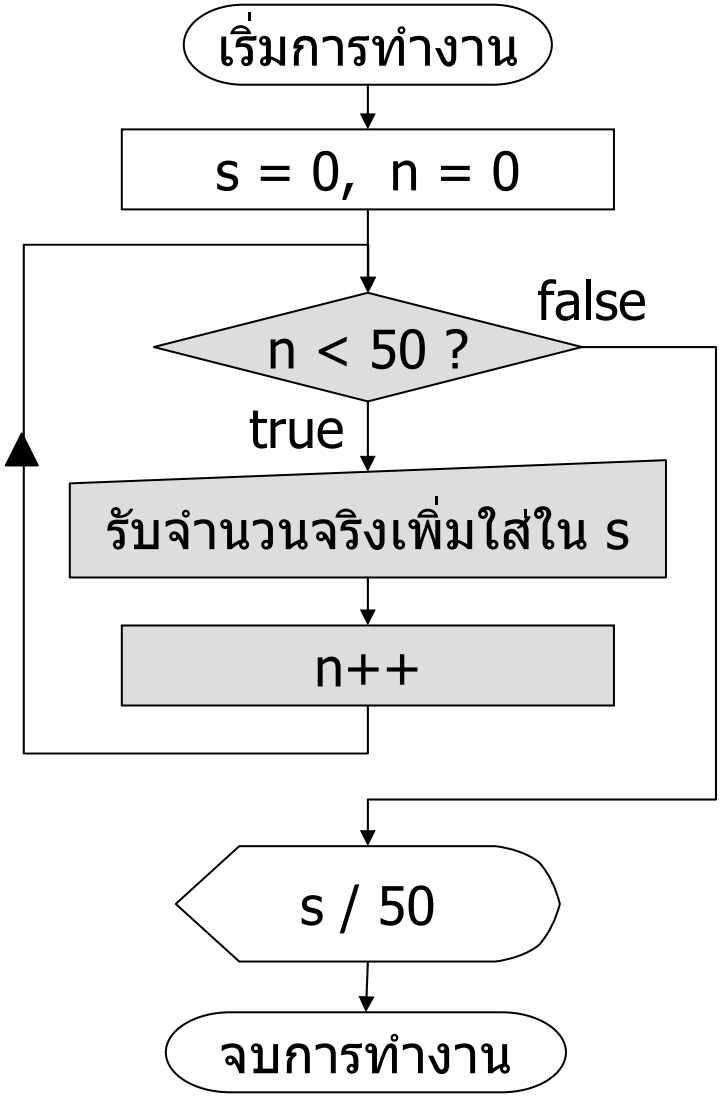
```
JLab>java Mean
data #1 : 12
data #2 : 35.5
data #3 : 16.8
data #4 : 19.0
data #5 : 2.2
mean = 17.1
JLab>
```

ผังงานแบบวงวน do-while



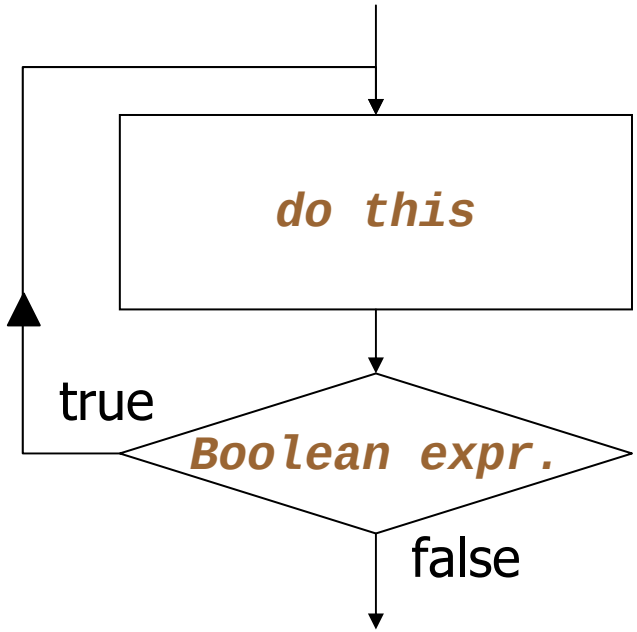

```
double s = 0.0;
int n = 0;

do {
   s += JLabIO.readDouble();

   n++;


} while( n < 50 );


System.out.println( s/50 );
```

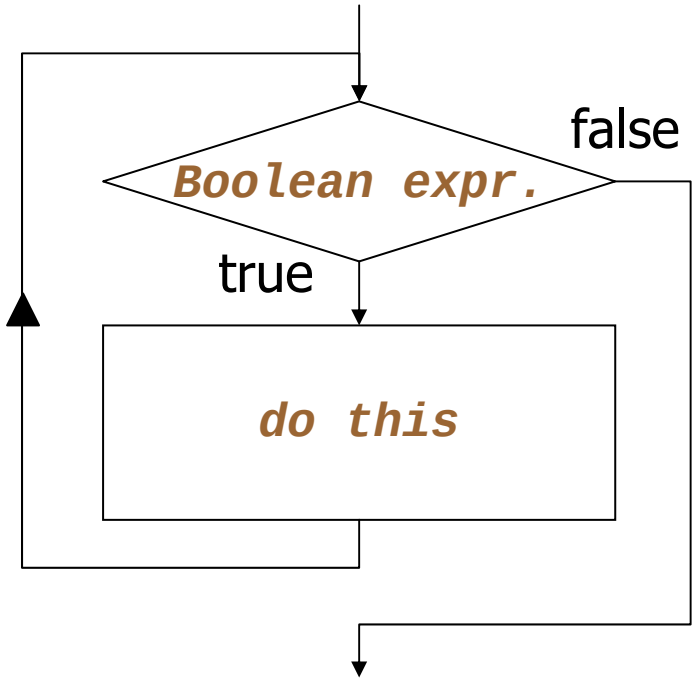
ผังงานแบบวงวน while




```
double s = 0.0;
int n = 0;


while ( n < 50 ) {

   s += JLabIO.readDouble();

   n++;

}

System.out.println( s/50 );
```

วงวนแบบ do-while และ while






```
do {
   do this
} while ( Boolean expr );
```

```
while ( Boolean expr ) {
   do this
}
```

โปรแกรมหารากที่สอง

- Requirement
  - อ่านจำนวนจากผู้ใช้ แล้วแสดงรากที่สองของจำนวนนั้น
- Analysis
  - รับจำนวนจริงผ่านทางแป้นพิมพ์
  - ใช้สูตรในการหารากที่สอง $x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k} \right)$
  - แสดงรากที่สองที่หาได้ทางจอภาพ
- Design
  - เขียนผังงาน
- Implementation
  - เขียนโปรแกรม

การประมาณค่ารากที่สอง

- ต้องการหารากที่สองของ a
- ถ้า x_k เป็นค่าประมาณของ √a
- ค่าของ $x_{k+1} = (x_k + a/x_k)/2$ จะเป็นค่าประมาณที่ใกล้ √a มากกว่า x_k (ไม่พิสูจน์)

```
a = 4.0
x0 = 1.0
x1 = 2.5
x2 = 2.05
x3 = 2.000609756097561
x4 = 2.000000092922294 7
x5 = 2.000000000000002
x6 = 2.0
```

```
a = 100.0
x0 = 1.0
x1 = 50.5
x2 = 26.24009900990099
x3 = 15.025530119986813
x4 = 10.840434673026925
x5 = 10.032578510960604
x6 = 10.000052895642693
x7 = 10.000000000139897
x8 = 10.0
```

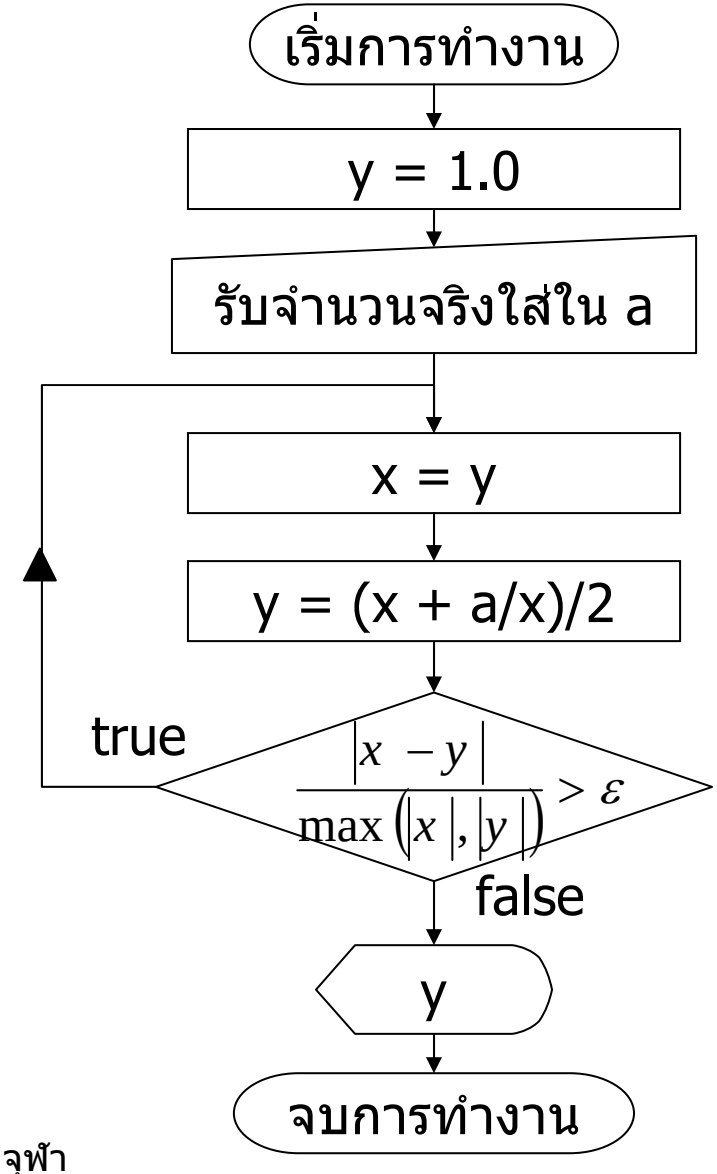
แล้วจะประมาณค่าไปสักกี่รอบ ?

- หาค่า $x_{k+1} = (x_k + a/x_k)/2 \quad k = 1,2,3,...$ เริ่มด้วย $x_0 = 1$
- หาไปจน x_{k+1} กับ x_k มีค่าใกล้เคียงกันมาก
- x และ y (แบบ double) มีค่าใกล้เคียงกันมากเมื่อ

$$\frac{|x-y|}{\max(|x|,|y|)} \leq \varepsilon$$

(ให้ ε มีค่าเท่ากับ 10^{-14})

ผังงานการหารากที่สอง



โปรแกรมหารากที่สอง

```
import jlab.JLabIO;
public class Sqrt {
  public static void main(String[] args) {
    double x, y = 1.0;
    double a = JLabIO.readDouble("a = ");
    do {
      x = y;
      y = (x + a / x) / 2.0;
    } while ((Math.abs(x - y) /
              Math.max(Math.abs(x), Math.abs(y)))
              >= 1E-14);
    System.out.println("sqrt(" + a + ") = " + y);
  }
}
```

```
JLab>java Sqrt
x = 9E200
sqrt(9.0E200) = 3.0E100
JLab>
```

หมุนในวงวน 339 รอบ

การอ่านแฟ้มข้อความขั้นพื้นฐาน

- แฟ้ม (file) คือที่เก็บข้อมูลในหน่วยความจำสำรอง (โดยทั่วไปคือ harddisk)
- แต่ละแฟ้มมีชื่อกำกับ
- Text file คือแฟ้มข้อความที่เก็บสตริง เป็นบรรทัดๆ แต่ละบรรทัดคั่นด้วยรหัสขึ้นบรรทัดใหม่

C:\DATA\SCORE.TXT

```
4531001321     3
4531003621     3
4531004221     4
4531005921     3
4531006521     3
4531101421     4
4531102021     4
4531102221     0
```

การใช้ JLabIO อ่านแฟ้มข้อความ

```
import java.io.*;
import jlab.JLabIO;

 // open file
 BufferedReader in = JLabIO.openFile("c:\dat.txt");
 ...
 // read the next line from file
 String txt = in.readLine();
 if (txt == null) // end of file
 ...
 // close the file
 in.close();
```

โปรแกรมแสดง text file ออกจอภาพ

```
import jlab.JLabIO;
import java.io.*;

public class DumpFile1 {
  public static void main(String[] args)
         throws IOException {

    String fn = JLabIO.readString("file name : ");
    BufferedReader in = JLabIO.openFile(fn);
    String txt;
    do {
      txt = in.readLine();
      if (txt != null) System.out.println(txt);
    } while ( txt != null );
    in.close();
  }
}
```

โปรแกรมแสดง text file ออกจอภาพ

```
import jlab.JLabIO;
import java.io.*;

public class DumpFile2 {
  public static void main(String[] args)
         throws IOException {

    String fn = JLabIO.readString("file name : ");
    BufferedReader in = JLabIO.openFile(fn);
    String txt;
    while ( (txt = in.readLine()) != null ) {
      System.out.println(txt);
    }
    in.close();
  }
}
```

ทำไมต้องมี throws IOException ที่ main ?

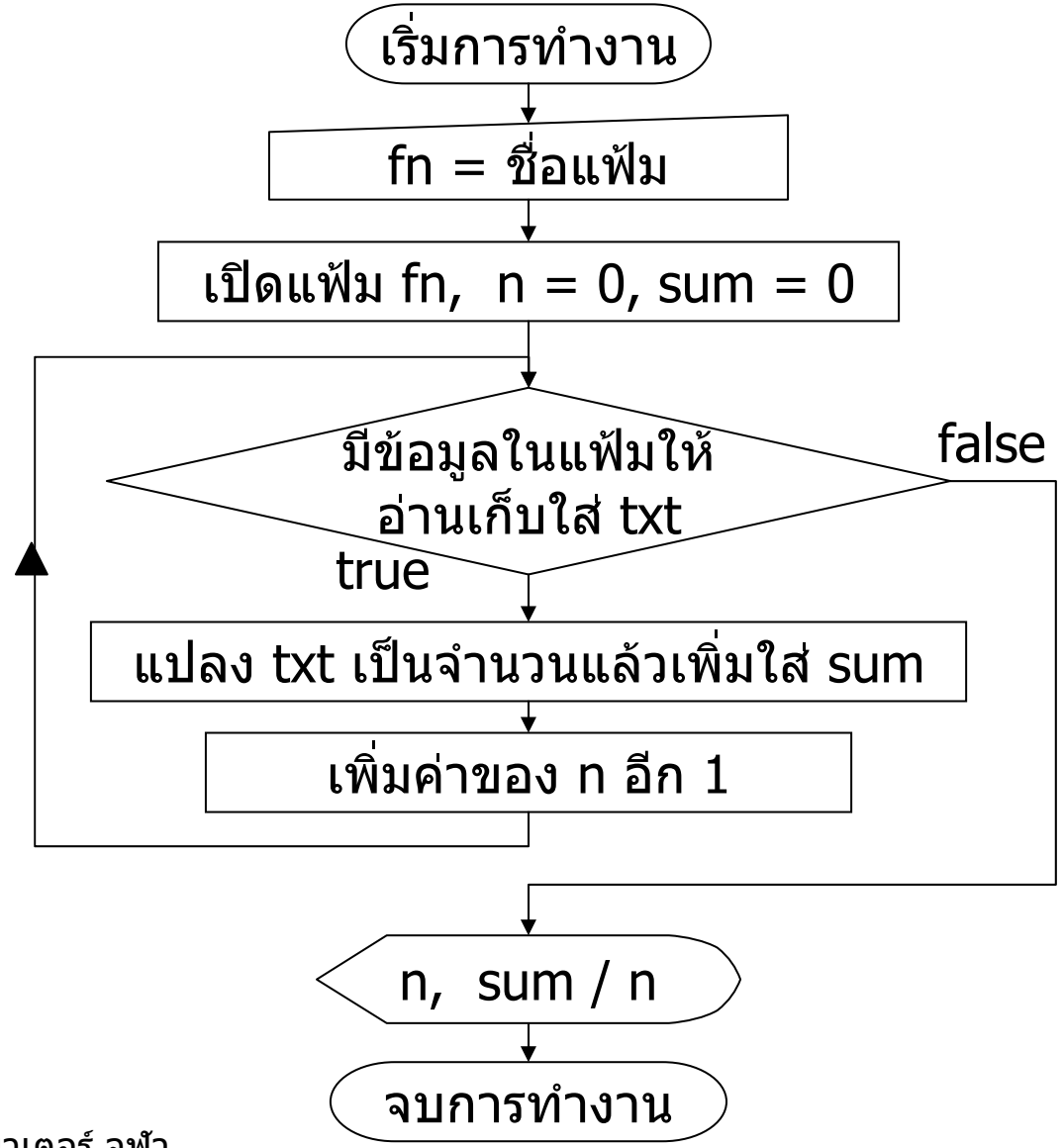
- มีโอกาสเกิดข้อผิดพลาดระหว่างการทำงาน ซึ่งเกี่ยวข้องกับแฟ้มข้อมูลได้หลายประการ (IOException)
  - เปิดแฟ้มที่ไม่มีอยู่จริง (เช่นสะกดชื่อผิด)
  - ตอนอ่านข้อมูลจากแฟ้ม อาจเกิดข้อผิดพลาดขึ้นได้ (เช่น อ่านจาก diskette แต่ผู้ใช้ดันดึงแผ่นออก)
- เพื่อความง่าย ขอไม่สนใจข้อผิดพลาดเหล่านี้
- เลยโยนข้อผิดพลาดแบบ IOException ให้ระบบ (ตัวโปรแกรมที่เราเขียนไม่ขอรับผิดชอบ)

`IOException` เป็นชื่อ class มาตรฐานในระบบจาวา

โปรแกรมหาค่าเฉลี่ย

- Requirement
  - หาค่าเฉลี่ยจากข้อมูลทั้งหมดในแฟ้ม
- Analysis
  - ผู้ใช้ระบุชื่อแฟ้มทางแป้นพิมพ์
  - แฟ้มข้อมูลเก็บจำนวนจริงบรรทัดละหนึ่งจำนวน
  - ใช้ Double.parseDouble(s) แปลงสตริง s เป็น double
  - แสดงจำนวนข้อมูล และค่าเฉลี่ยที่หาได้ทางจอภาพ
- Design
  - เขียนผังงาน
- Implementation
  - เขียนโปรแกรม

ผังงานการหาค่าเฉลี่ย



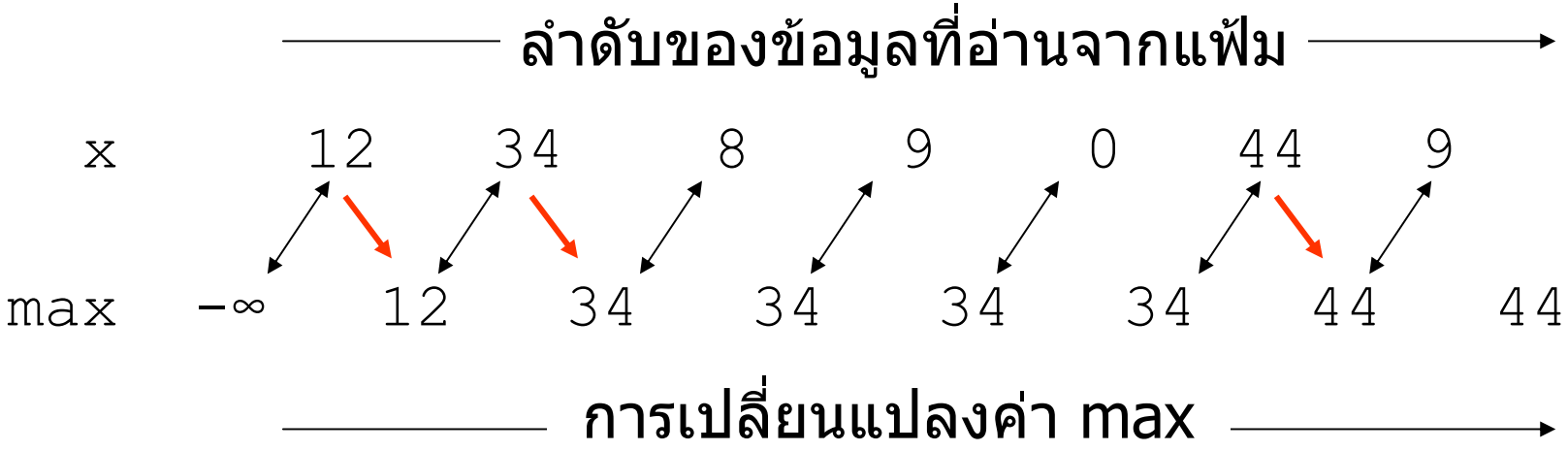
โปรแกรมหาค่าเฉลี่ย

```
import jlab.JLabIO;
import java.io.*;
public class Average1 {
  public static void main(String[] args)
         throws IOException {
    String fn = JLabIO.readString("file name : ");
    BufferedReader in = JLabIO.openFile(fn);
    int n = 0;
    double sum = 0.0;
    while ((txt = in.readLine()) != null) {
      sum += Double.parseDouble(txt);
      n++;
    }
    in.close();
    System.out.println("n = " + n +
                       " avg = " + sum / n);
  }
}
```

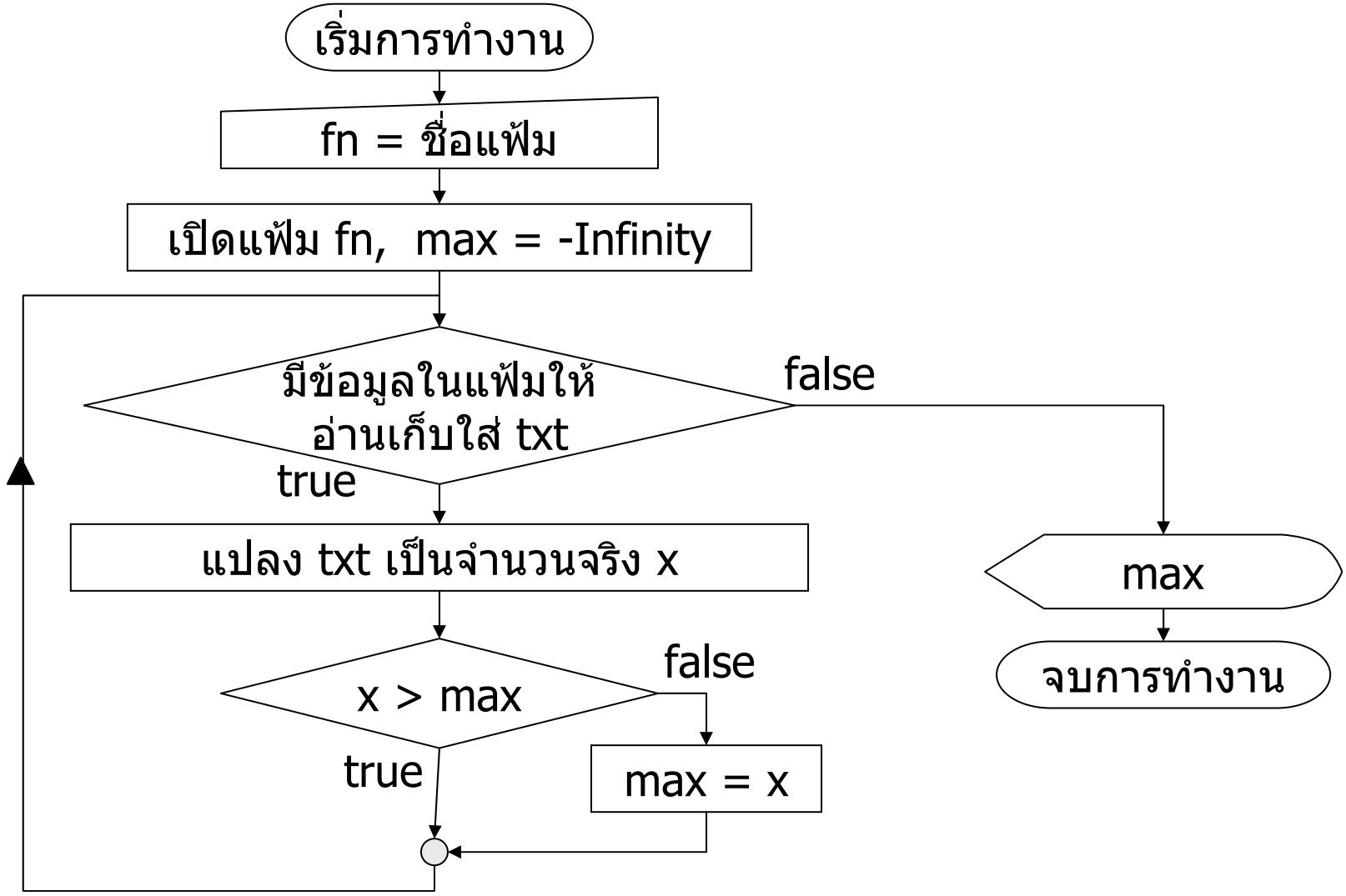
โปรแกรมหาค่ามากสุด

- Requirement
  - หาค่ามากสุดจากข้อมูลทั้งหมดในแฟ้ม
- Analysis
  - ผู้ใช้ระบุชื่อแฟ้มทางแป้นพิมพ์
  - แฟ้มข้อมูลเก็บจำนวนจริงบรรทัดละหนึ่งจำนวน
  - แสดงค่ามากสุดที่หาได้ทางจอภาพ
- Design
  - เขียนผังงาน
- Implementation
  - เขียนโปรแกรม

การหาค่ามากสุด



ผังงานการหาค่ามากสุด



โปรแกรมหาค่ามากสุด

```
import jlab.JLabIO;
import java.io.*;
public class MaxValue {
  public static void main(String[] args)
         throws IOException {

    String fn = JLabIO.readString("file name : ");
    BufferedReader in = JLabIO.openFile(fn);
    double x, max = Double.NEGATIVE_INFINITY;
    while ((txt = in.readLine()) != null) {
      x = Double.parseDouble(txt);
      if (x > max) max = x;
    }
    in.close();
    System.out.println("max = " + max);
  }
}
```

การใช้ while กับโปรแกรม EAN-13

```
import jlab.JLabIO;
public class EAN13CheckDigit {
  public static void main(String[] args) {
    int s = 0;
    String d = JLabIO.readString("Enter 12 digits : ");
    s += Character.digit(d.charAt(1), 10);
    s += Character.digit(d.charAt(3), 10);
    s += Character.digit(d.charAt(5), 10);
    s += Character.digit(d.charAt(7), 10);
    s += Character.digit(d.charAt(9), 10);
    s += Character.digit(d.charAt(11), 10);
    s *= 3;
    s += Character.digit(d.charAt(0), 10);
    s += Character.digit(d.charAt(2), 10);
    s += Character.digit(d.charAt(4), 10);
    s += Character.digit(d.charAt(6), 10);
    s += Character.digit(d.charAt(8), 10);
    s += Character.digit(d.charAt(10), 10);
    System.out.println("Check Digit is " + (10-(s%10))%10);
  }
}
```

การใช้ while กับโปรแกรม EAN-13

```
import jlab.JLabIO;
public class EAN13CheckDigit {
  public static void main(String[] args) {
    int s = 0, k;
    String d = JLabIO.readString("Enter 12 digits : ");

    k = 1;
    while (k<12) {
      s += Character.digit(d.charAt(k), 10);
      k += 2;
    }
    s *= 3;

    k = 0;
    while (k<12) {
      s += Character.digit(d.charAt(k), 10);
      k += 2;
    }
    System.out.println("Check Digit is " + (10-(s%10))%10);
  }
}
```

การใช้ while กับโปรแกรม EAN-13

```
import jlab.JLabIO;
public class EAN13CheckDigit {
  public static void main(String[] args) {
    int s = 0, k;
    String d = JLabIO.readString("Enter 12 digits : ");

    k = 0;
    while (k<12) {
      if ( (k % 2) == 0 ) {
        s += Character.digit(d.charAt(k), 10);
      } else {
        s += 3 * Character.digit(d.charAt(k), 10);
      }
      k++;
    }

    System.out.println("Check Digit is " + (10-(s%10))%10);
  }
}
```

while และ do-while

- วงวนที่เขียนด้วย while เปลี่ยนเป็น do-while ได้
- วงวนที่เขียนด้วย do while เปลี่ยนเป็น while ได้
- โปรแกรมเมอร์ควรเลือกแบบที่สื่อความหมาย อ่านง่าย
- while : ตราบเท่าที่เงื่อนไขเป็นจริง จึงจะทำ
  - อาจไม่ได้ทำใน block
- do-while ทำไปเรื่อยๆ ตราบเท่าที่เงื่อนไขยังเป็นจริง
  - อย่างน้อยต้องทำใน block หนึ่งครั้ง

วงวนที่ทำเป็นจำนวนครั้งตามที่กำหนด

- มีตัวแปรคอยนับตามจำนวนรอบ

```
i = 1;
while (i <= n) {

   // do something here

   i++;
}
```

```
i = 0;
while (i < n) {

   // do something here

   i++;
}
```

```
i = n;
while (i > 0) {

   // do something here

   i--;
}
```

```
i = n-1;
while (i >= 0) {

   // do something here

   i--;
}
```

ข้อควรระวัง

- วนขาดหนึ่งรอบ วนเกินหนึ่งรอบ หรือวนติด loop

```
i = 1;
while (i < n) {

   // do something here

   i++;
}
```

```
i = 0;
while (i <= n) {

   // do something here

   i++;
}
```

```
i = n;
while (i > 1) {

   // do something here

   i--;
}
```

```
i = 0;
while (i != n) {
   // The test above can
   // cause infinite
   // loop.
   i++;
}
```

วงวนแบบ for

- วงวนที่ทำเป็นจำนวนครั้งตามที่กำหนด (counting loop) เป็นวงวนที่มักพบบ่อยมาก
- ตัวภาษาจึงมีคำสั่ง for เพื่อทำ counting loop

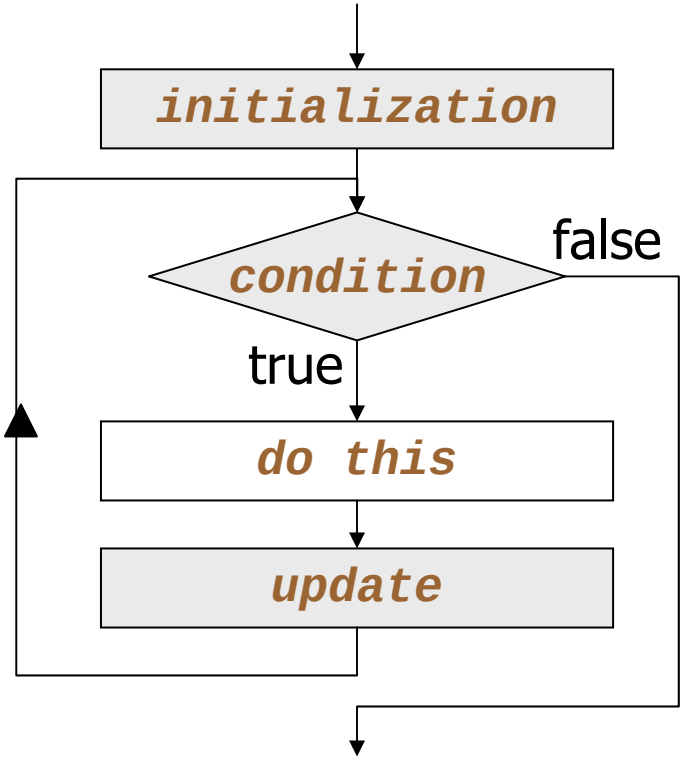
```
i = 1;
while (i <= n) {
   // do something here
   i++;
}
```

```
for (i = 1; i <= n; i++) {
   // do something here
}
```

```
i = n;
while (i > 0) {
   // do something here
   i--;
}
```

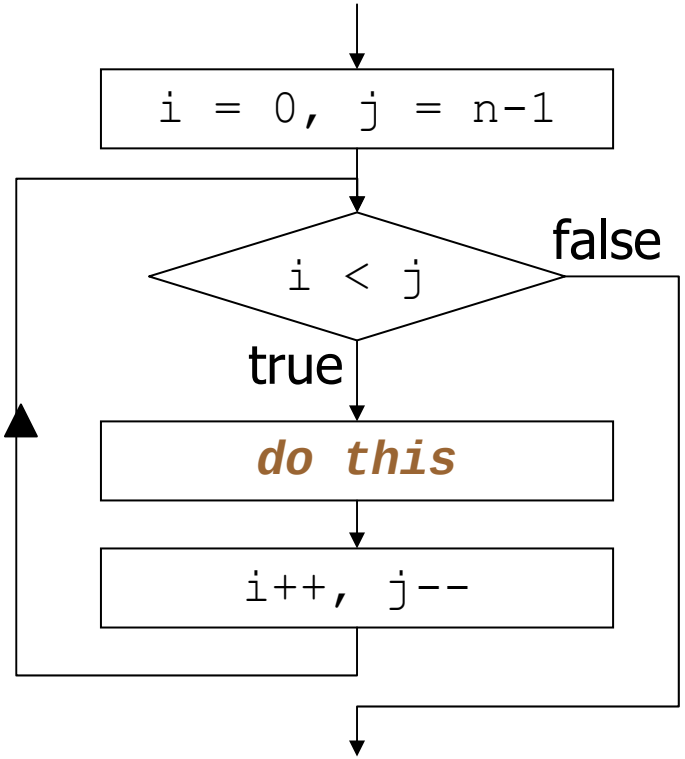
```
for (i = n; i > 0; i--) {
   // do something here
}
```

วงวนแบบ for




```
for ( initialization ; condition ; update ) {
  do this
}
```

comma




```
for ( i = 0, j = n-1; i < j; i++, j--) {
  do this
}
```

ตัวอย่างวงวนแบบ for

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

```
for (i = 1; i <= n; i += 2) {
   System.out.println(i);
}
```

```
n = JLabIO.readInt("n = ");
for (i = 2; (i < n) && (n % i != 0); i++) {
}
if (i < n) {
   System.out.println( n + " = " + i + "x" + (n/i) );
} else {
   System.out.println( n + " is prime.");
}
```

หนึ่ง block หรือ หนึ่ง statement

```
for ( initialization ; condition ; update ) {
  statements
}
```

```
for ( initialization ; condition ; update )
  statement ;
```

```
do {
  statements
} while ( Boolean expr );
```

```
while ( Boolean expr ) {
  statements
}
```

```
do
  statement ;
while ( Boolean expr );
```

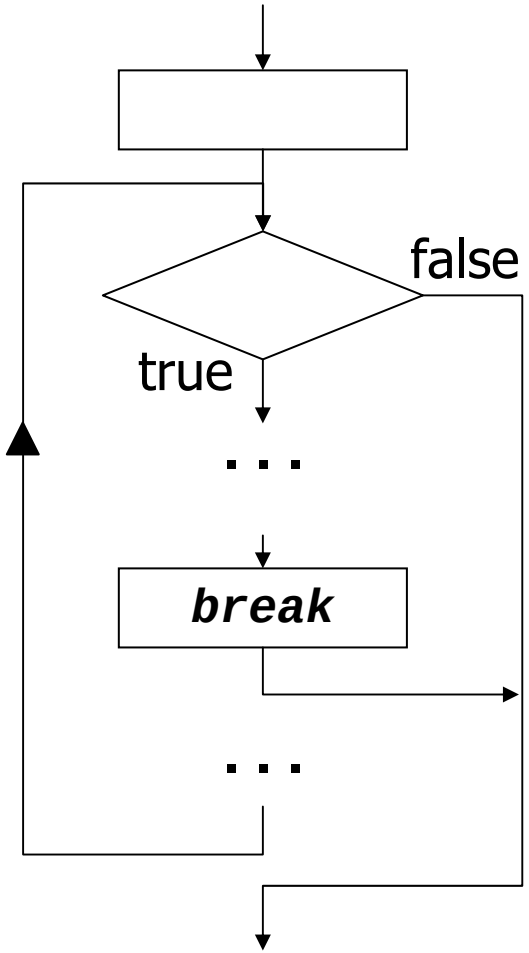
```
while ( Boolean expr )
  statement ;
```

อย่าใช้ for ผิดวัตถุประสงค์

```
double x = 0, y = 1;
double a = JLabIO.readDouble("a = ");
while ((Math.abs(x - y) /
            Math.max(Math.abs(x), Math.abs(y)))
            >= 1E-14) {
   x = y;
   y = (x + a / x) / 2.0;
}
System.out.println("sqrt(" + a + ") = " + y);
```

```
double x = 0, y = 1;
double a = JLabIO.readDouble("a = ");
for ( ;
      (Math.abs(x - y) /
       Math.max(Math.abs(x), Math.abs(y))) >= 1E-14);
      x = y, y = (x + a / x) / 2.0 ) ;
System.out.println("sqrt(" + a + ") = " + y);
```

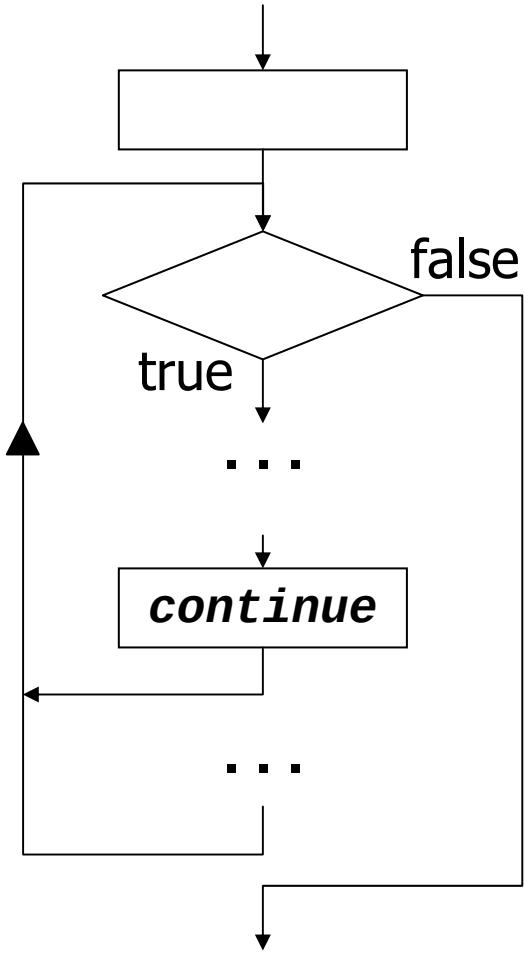
คำสั่ง break เพื่อออกจากวงวน




```
while ((t = in.readLine()) != null) {
   if ("END".equals(t)) break;
   x = Double.parseDouble(t);
   if (x > max) max = x;
}
```

สนใจหาค่ามากสุดของจำนวนที่เก็บในแฟ้มข้อมูลก่อนบรรทัดที่มีคำว่า END

คำสั่ง continue เพื่อกลับไปต้นวงวน

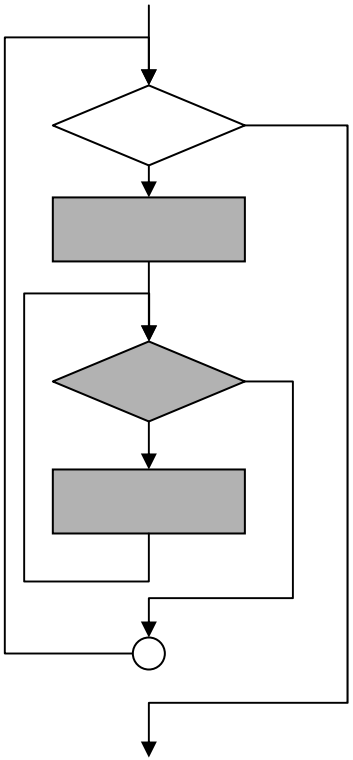



```
while ((t = in.readLine()) != null) {
   x = Double.parseDouble(t);
   if (x <= 0) continue;
   if (x > max) max = x;
}
```

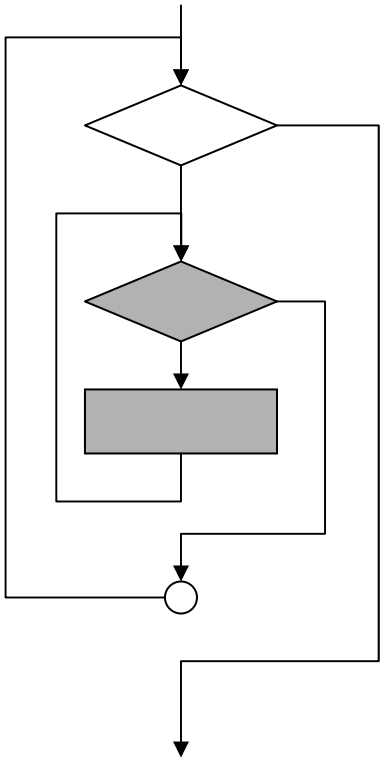
สนใจหาค่ามากสุดของเฉพาะจำนวนเต็มบวก ที่เก็บในแฟ้มข้อมูล

Nested Loops

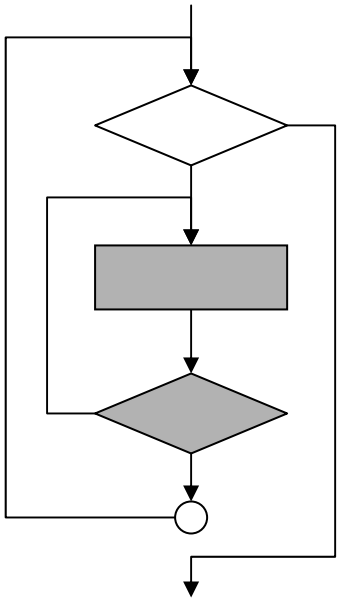
- ภายในวงวนหนึ่งจะมีวงวนอื่นก็ได้



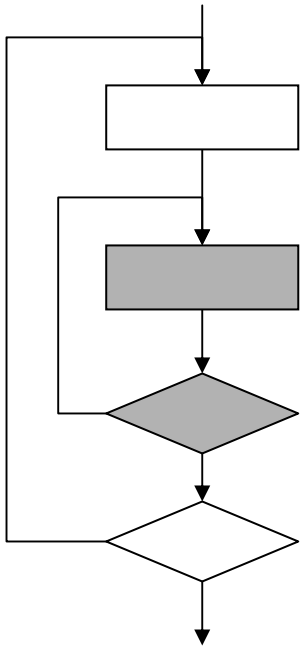
for ใน while



while ใน while



do-while ใน while

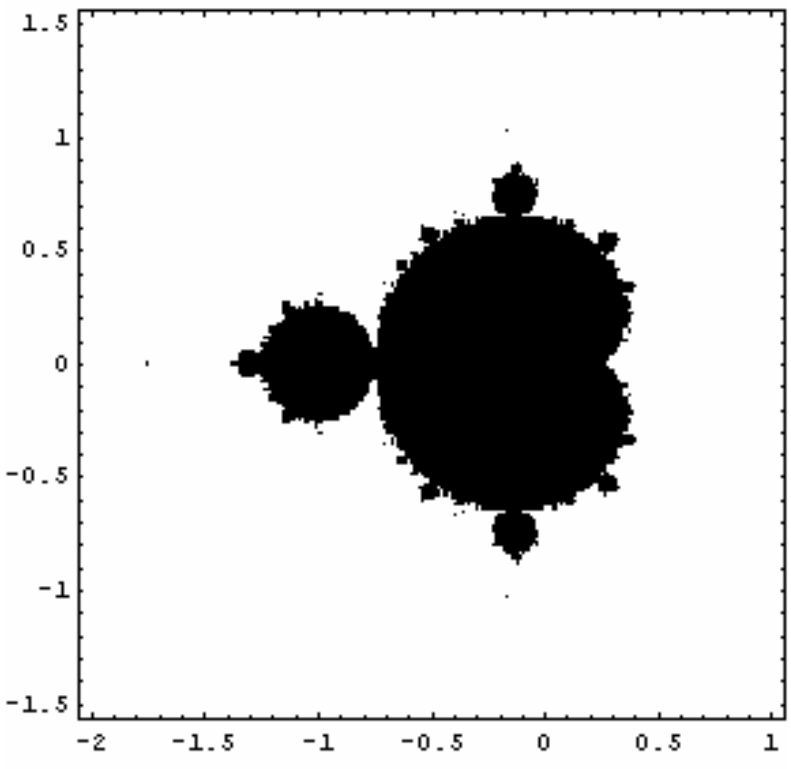


do-while ใน do-while

break และ continue ภายในวงวนใด ก็จะทำงานกับวงวนนั้น

Mandelbrot Set

- ให้ลำดับของจำนวนเชิงซ้อน d_n คำนวณได้จาก $d_n = (d_{n-1})^2 + c$ โดยที่ $d_0 = 0+i0 = 0$ และ c เป็นจำนวนเชิงซ้อน
- c เป็นสมาชิกของ Mandelbrot set ถ้าลำดับของ $|d_n|$ ลู่เข้าหาค่าคงตัว



แกน x แทนเส้นจำนวนจริง แกน y แทนเส้นจำนวนจินตภาพ
จุดดำในรูปคือสมาชิกของ Mandelbrot set

การบวกและคูณจำนวนเชิงซ้อน

- $| (x+iy) | = (x^2 + y^2)^{1/2}$
- $(x+iy)+(p+iq) = (x+p) + i(y+q)$
- $(x+iy)(p+iq) = (xp-yq) + i(xq+yp)$

ตัวอย่างการเปลี่ยนแปลงของค่า $|d_n|$

c = (0.2+i0.2) จะได้
0.344, 0.351, 0.327, 0.305, 0.304, 0.312, 0.316, 0.316, 0.314,
0.313, 0.313, 0.313, 0.314, 0.314, 0.314, ...(converge)

c = (1.0+i1.0) จะได้
3.162, 9.899, 97.01, 9409, 8.853E7, 7.837E15, 6.142E31,
3.773E63, 1.424E127, ... (diverge to infinity)

โปรแกรมแสดง Mandelbrot set

```
public void paint(Graphics g) {
  double xnew, ynew, xold, yold, x, y;
  double stepX = 3.0 / getSize().width;
  double stepY = 3.0 / getSize().height;
  int n;
  for (y = -1.5; y <= 1.5; y += stepY) {
    for (x = -2.0; x <= 1.0; x += stepX) {
      for (xold = x, yold = y, n = 0; (n < 256); n++) {
        xnew = xold * xold - yold * yold + x;
        ynew = 2 * xold * yold + y;
        if (Math.sqrt(xnew * xnew + ynew * ynew) > 2) break;
        xold = xnew;
        yold = ynew;
      }
      if (n >= 256) {
        g.drawRect((int)((x + 2) / stepX),
                   (int)((y + 1.5) / stepY), 1, 1);
      }
    }
  }
}
```

Visualizing Mandelbrot Set