2110101 : การทำโปรแกรมคอมพิวเตอร์

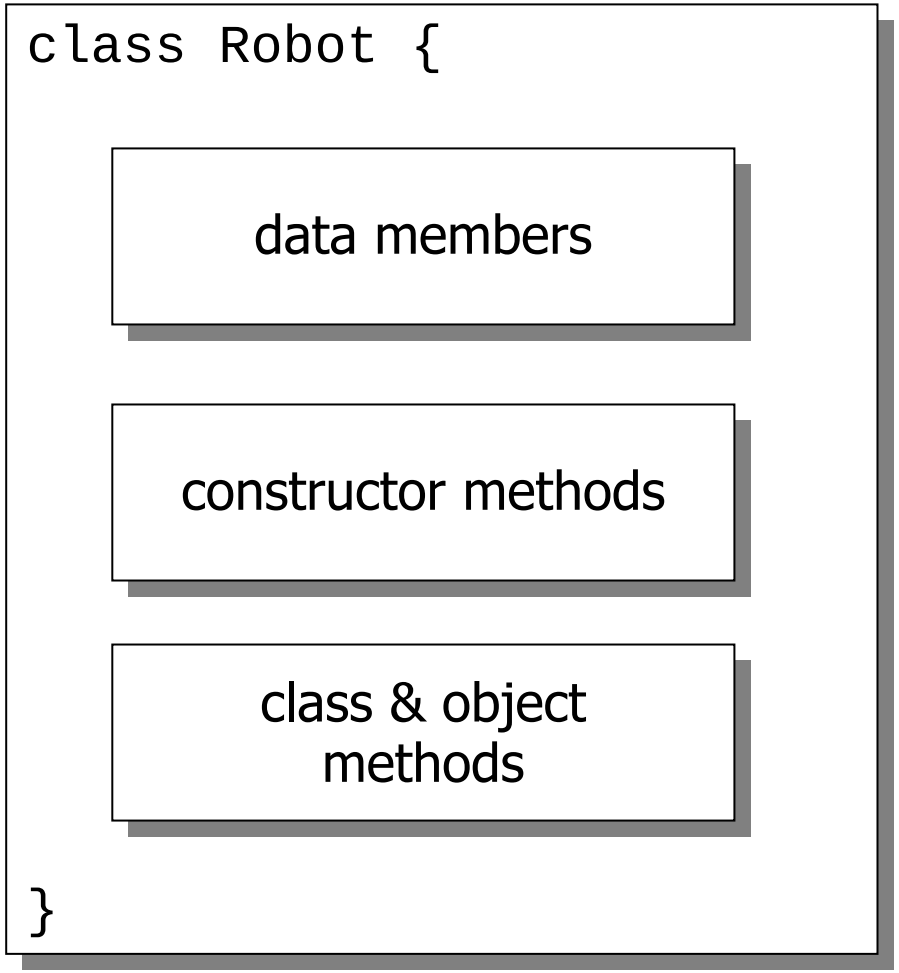
Classes & Objects

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

- องค์ประกอบของคลาส
- การประกาศ การสร้าง และการทำลายออบเจ็กต์
- Instance variables
- Class methods และ object methods
- Constructor methods
- ตัวอย่าง : คลาส Complex
  - Newton-Raphson
  - Mandelbrot Set

องค์ประกอบของคลาส



จำนวนเชิงซ้อน

- จำนวนเชิงซ้อน $z = x + iy$
  - x เป็นจำนวนจริง
  - iy เป็นจำนวนจินตภาพ
    - y เป็นจำนวนจริง
    - $i = \sqrt{-1}$ (imaginary unit)

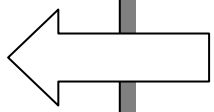
$$e^{i\pi} = -1$$

- conjugate $\bar{z} = x - iy$
- absolute square : $|z|^2 = z\bar{z} = (x+iy)(x-iy) = x^2 + y^2$
- ให้ $z_1 = x_1 + iy_1$ และ $z_2 = x_2 + iy_2$
- $z_1 + z_2 = (x_1+x_2) + i(y_1+y_2)$
- ได้เจออีกเยอะใน ไฟฟ้า เครื่องกล โยธา ...

อยากได้จำนวนเชิงซ้อน

- จาวามีแต่จำนวนจริง จำนวนเต็ม
- แต่ไม่มีจำนวนเชิงซ้อน ต้องออกแบบเอง
- เขียน class ใหม่ชื่อ Complex
- เขียน class = เขียนนิยามของประเภทข้อมูลแบบใหม่
- เก็บจำนวนเชิงซ้อนหนึ่งจำนวน = เก็บจำนวนจริงสองจำนวน $z = x + iy$

```
public class Complex {
  public double  x;
  public double  y;
}
```



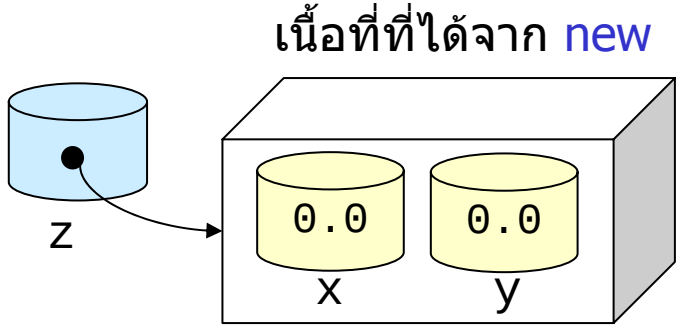
fields
(data members)

การประกาศและการสร้าง

```
public class Complex {
   public double  x;
   public double  y;
}
```

นิยามของคลาส

```
   Complex z;
   z = new Complex();
```



`Complex z;`
เป็นการประกาศและสร้างตัวแปร z ซึ่งมีไว้อ้างอิงเนื้อที่เก็บข้อมูลที่เป็นแบบ Complex

`z = new Complex();`

เป็นการสร้างเนื้อที่เก็บข้อมูลที่เป็นแบบ Complex

ศัพท์เทคนิค

class ชื่อ Complex

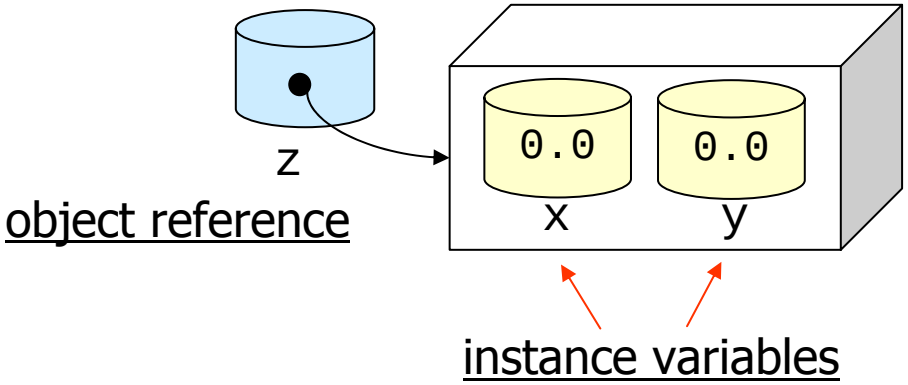
```
public class Complex {

   public double x;

   public double y;
}
```

fields (หรือ data members) ของ class

ตัวแปรอ้างอิง object ของ class Complex



object หนึ่ง (หรือ instance) ของคลาส Complex

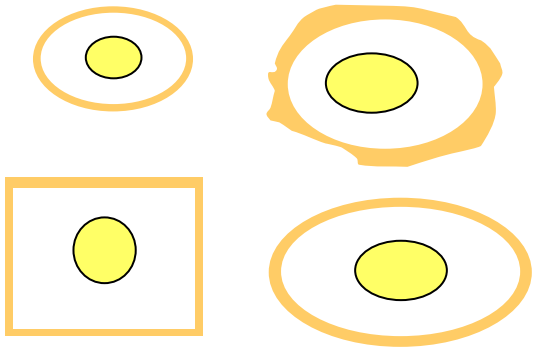
ตำรากับข้าวกับกับข้าว

สูตรไข่ดาว

- ไข่ไก่หนึ่งฟอง
- น้ำมันครึ่งถ้วย

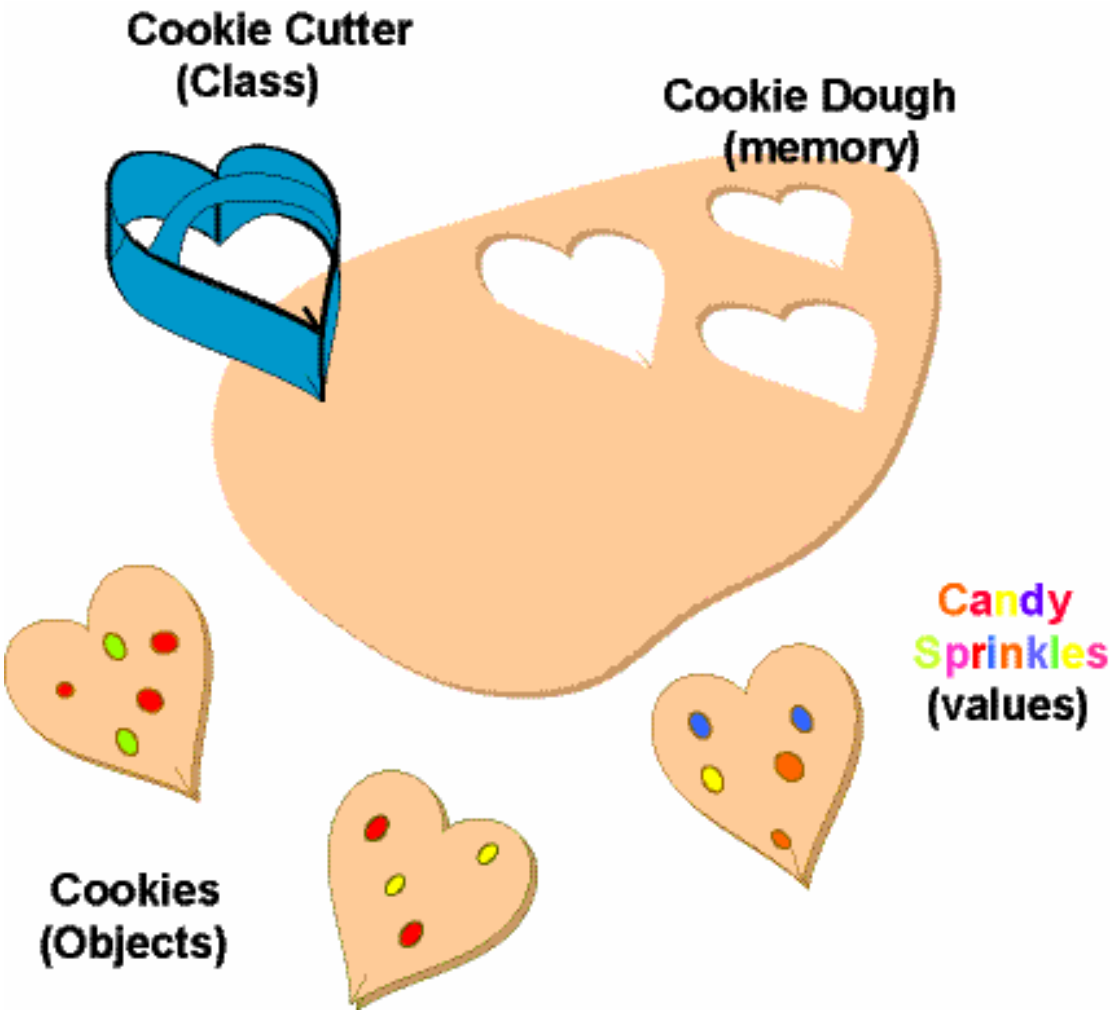
class บรรยายองค์ประกอบ

ไข่ดาวที่ทอดแล้วตามสูตร



object คือของจริงที่ถูกสร้างขึ้น object ของ class เดียวกัน แต่คนละก้อนกัน ก็อาจมีรายละเอียดภายในไม่เหมือนกัน

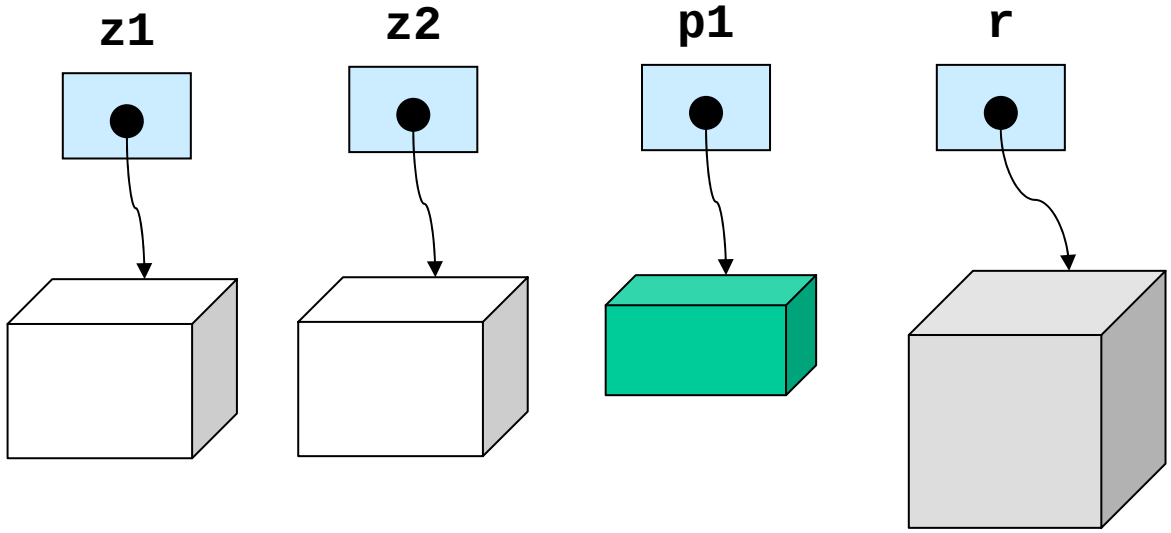
คุกกี้


Cookie Cutter
(Class)
Cookie Dough
(memory)
Candy
Sprinkles
(values)
Cookies
(Objects)

new เป็นเสมือนโรงงานสร้าง objects

- new จะสร้าง object ใหม่ของ class ที่กำหนดให้
- ต้องมีตัวแปรคอยรับตำแหน่งอ้างอิง object ใหม่

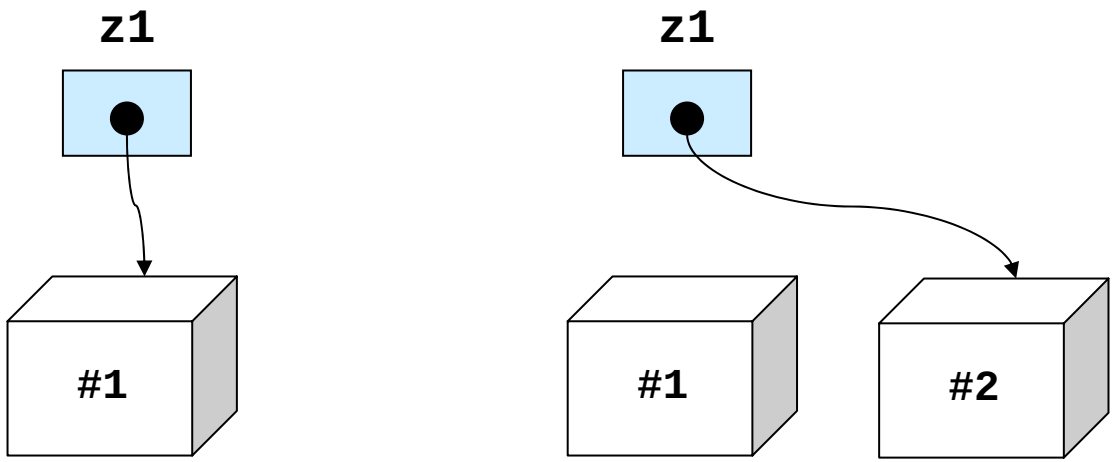
```
Complex   z1 = new Complex();
Complex   z2 = new Complex();
Point     p1 = new Point();
Rectangle r  = new Rectangle();
```



ขยะ

- Object ที่ถูกสร้างขึ้น แต่ไม่มีตัวแปรใดอ้างอิง
- เนื้อที่สูญเปล่า
- ระบบจาวาจะ "เก็บขยะ" นำเนื้อที่เหล่านี้ไปใช้ใหม่

```
Complex   z1 = new Complex();  // #1
...
z1 = new Complex();            // #2
```

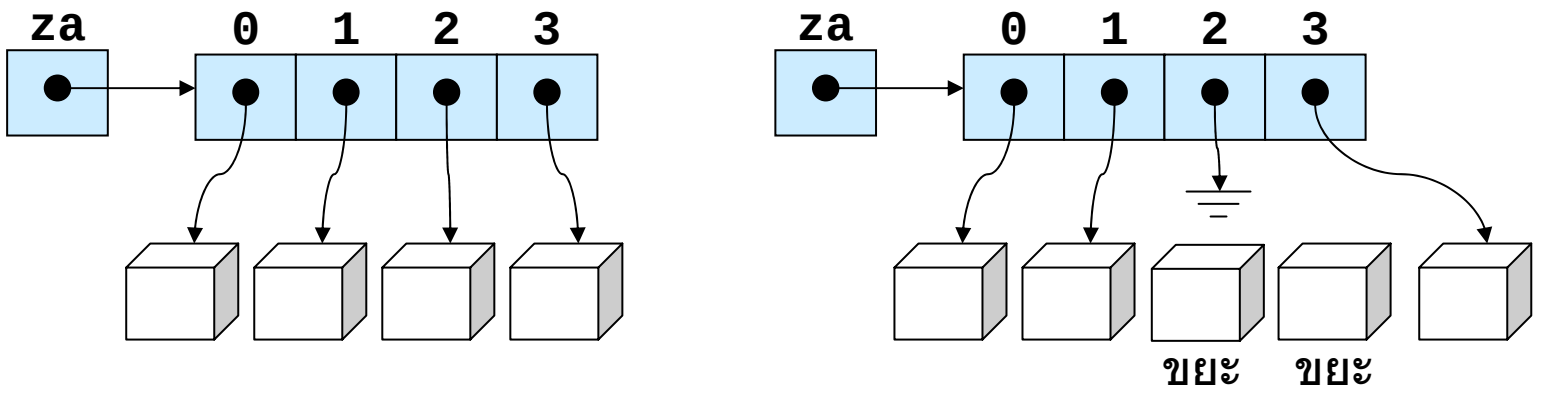


การสร้างและทำลาย object

- สร้างโดยใช้ new
- ทำลายโดยอย่าให้มีตัวแปรใดไปอ้างอิงถึง

```
Complex [] za  = new Complex[4];
for(int i=0; i< za.length; i++) {
  za[i] = new Complex();
}
za[2] = null;
za[3] = new Complex();
```

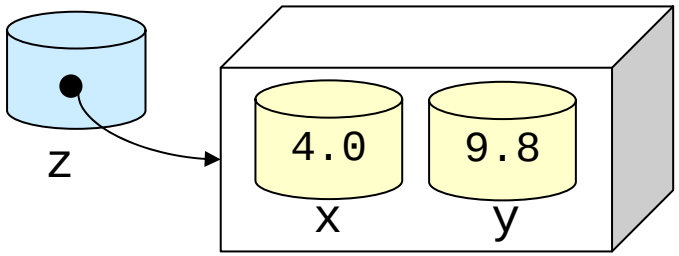
null เป็นค่าคงตัวพิเศษแทน object reference ที่ไม่ได้อ้างอิงอะไร



การใช้ Instance Variables

- instance variables คือตัวแปรต่างๆ ภายในของ object ที่ได้ถูกสร้างขึ้น
- อยากจะใช้ ให้เขียนในรูป *object* . *variableName*

```
public class Complex {
  public double  x;
  public double  y;
}
```




```
  Complex z = new Complex();

  z.x = 4;
  z.y = 9.8;
```

การให้ค่าเริ่มต้นกับ instance variable

```
public class Complex {
  public double  x;
  public double  y;
}
```

ปกติเมื่อ object หนึ่งถูกสร้างขึ้น จะมีการให้ค่า 0 และ false กับ instance variables ต่างๆ ของ object ที่เป็นประเภทจำนวน และ boolean ตามลำดับ

```
public class Complex {
  public double  x = 1.0;
  public double  y = 0.0;
}
```

ใส่ค่าเริ่มต้นเองก็ได้ เช่นเดียวกับการตั้งค่าเริ่มต้นของตัวแปรภายในเมท็อด

Methods

- ออกแบบคลาสเพื่อบรรยายลักษณะของคลาส
  - fields บรรยายลักษณะของ object
  - methods บรรยายบริการที่มีให้กระทำกับ object
- คลาส Complex
  - fields :
    - x และ y แทนจำนวนจริงและจินตภาพ
  - methods :
    - หา absolute
    - หา congujate
    - หาผลบวกของจำนวนเชิงซ้อนสองจำนวน
    - sort ข้อมูลใน array ของ int ที่กำหนดให้ ⇦ ไม่เกี่ยว

Class Methods vs. Object Methods

- **ออกแบบเมท็อดให้ผู้อื่นเรียกใช้ได้สองวิธี**
  - class methods *className* .*methodName*(...)

```
double a = Complex.abs(z1);
Complex z2 = Complex.conjugate(z1);
Complex z3 = COmplex.add(z2, z1);
```

  - object methods *objectName* .*methodName*(...)

```
double a = z1.abs();
Complex z2 = z1.conjugate();
Complex z3 = z2.add(z1);
```

ย้อนอดีต

- เคยใช้ class methods มาแล้ว
  - Math.sqrt(x)
  - JLabIO.readInt( )
  - Array.equals(a, b)
  - public static void main(String [] args)
  - ...
- เคยใช้ object methods มาแล้ว
  - str.toUpperCase()
  - in.readLine()
  - System.out.println(msg)
  - ...

Class Methods

```
public class Complex {
  public double  x;
  public double  y;

  public static double abs( Complex z ) {
    return Math.sqrt( z.x * z.x + z.y * z.y );
  }
}
```

บรรยายคลาส Complex

```
public class Test {
  public static void main( String [] args ) {
    double a = JLabIO.readDouble("real part : ");
    double b = JLabIO.readDouble("imag part : ");
    Complex z = new Complex();
    z.x = a;
    z.y = b;
    System.out.println( Complex.abs(z) );
  }
}
```

สร้าง object และใช้ method ของ Complex

Object Methods

```
public class Complex {
  public double  x;
  public double  y;

  public static double abs( ) {
    return Math.sqrt(this.x*this.x + this.y*this.y);
  }
}
```

ตัดคำว่า static ทิ้ง ก็หมายถึง object method

```
public class Test {
  public static void main( String [] args ) {
    double a = JLabIO.readDouble("real part : ");
    double b = JLabIO.readDouble("imag part : ");
    Complex z = new Complex();
    z.x = a;
    z.y = b;
    System.out.println( z.abs() );
  }
}
```

this ใช้อ้างถึง object ที่ถูกเรียกให้มาทำงานที่เมท็อด

Class Methods vs Object Methods

- class method มีคำว่า "static" กำกับที่หัว method
- object method ไม่มีคำว่า "static" กำกับ
- เรียก class methods ใช้ชื่อคลาสนำ
- เรียก object methods ใช้ชื่อออบเจกต์นำ
- คำถาม : บรรทัดใดข้างล่างนี้เรียกใช้ class method
  - System.getProperties( )
  - Math.random( )
  - JLabIO.format(x, 5, 2)
  - a.compareTo(b)
  - rectangle.setSize(w, h)
  - Math.PI

การตั้งชื่ออย่างมีระบบจะช่วยให้เข้าใจโปรแกรมได้ดีขึ้น

ตัวอย่าง Object Methods ของ Complex

```
public class Complex {
  public double  x, y;                    // data members

  public double abs( ) {
    return Math.sqrt(this.x*this.x + this.y*this.y);
  }
  public Complex conjugate() {
    Complex zbar = new Complex();
    zbar.x = this.x;
    zbar.y = -(this.y);
    return zbar;
  }
  public Complex add( Complex z ) {
    Complex sumz = new Complex();
    sumz.x = this.x + z.x;
    sumz.y = this.y + z.y;
    return sumz;
  }
}
```

$$\bar{z} = x - iy$$

$$z_1 + z_2 = (x_1 + x_2) + i(y_1 + y_2)$$

Object Method ที่ควรมี : toString

- toString มีไว้แปลงข้อมูลภายใน object (instance variables) ให้เป็น String ที่คนอ่านเข้าใจ
- โดยทั่วไปใช้เพื่อทำให้แสดงผลได้สะดวก เพราะว่า System.out.println(z) จะเรียก z.toString() โดยอัตโนมัติ

```
public class Complex {
  public double x, y;
  ...
  public String toString() {
    return "(" + this.x + ", i" + this.y + ")";
  }
  ...
}
```

Object Method ที่ควรมี : equals

- ทดสอบ obj1 == obj2 หมายความว่า obj1 และ obj2 เป็น object เดียวกันหรือไม่
- ทดสอบ obj1.equals(obj2) หมายความว่า object obj1 "เท่ากับ" object obj2 หรือไม่
- ความหมายของการ "เท่ากัน" ของ object นั้นขึ้นกับผู้ออกแบบ class

```
public class Complex {
  public double x, y;
  ...
  public boolean equals( Complex z ) {
    return this.x == z.x && this.y == z.y;
  }
  ...
}
```

equals ตามกฎในจาวาจะจุกจิกกว่านี้

ข้อสังเกต

- เรียกใช้ instance variable object.variable
- เรียกใช้ object method object.method(...)
- ต่างกันตรงที่มีวงเล็บหรือไม่

```
public class Complex {
  public double x, y;
  ...
  public String toString() {
    return "(" + this.x + ", i" + this.y + ")";
  }
  ...
}
```

```
Complex z1 = new Complex();
...
double p = z1.x;
String s = z1.toString();
double q = z1.x();              // WRONG
String t = z1.toString;         // WRONG
System.out.println( "what am I ?" );
```

ย้อนอดีต

- String และ array เป็น object
- String มีเมท็อด length()
- array มี instance variable ชื่อว่า length

```
String s = "Hi Ho Hi Ho;
int a = new int[100];

System.out.println( s.length() );
System.out.println( a.length );
```

Constructor Methods

- เป็นเมท็อดพิเศษของคลาส ซึ่งระบบเรียกใช้โดยอัตโนมัติหลัง new คลาสนั้น
- มีไว้เพื่อตั้งค่าเริ่มต้นให้กับ instance variables ของ object ใหม่ซึ่งเพิ่งถูกสร้าง
- เป็นเมท็อดพิเศษ
  - ชื่อต้องเหมือนชื่อ class
  - ต้องไม่ระบุ return type ของเมท็อด

```
public class Complex {
  public double x, y;

  public Complex( ) {
    this.x = 0.0;
    this.y = 0.0;
  }
```

Constructor Methods

- Overload ได้ด้วยการกำหนดรายการของพารามิเตอร์ที่แตกต่างกัน

```
public class Complex {
  public double x, y;

  public Complex( ) {
    this.x = 0.0;
    this.y = 0.0;
  }
  public Complex( double x1, double y1 ) {
    this.x = x1;
    this.y = y1;
  }
  public Complex( Complex z ) {
    this.x = z.x;
    this.y = z.y;
  }
```

```
Complex z1 = new Complex();
Complex z2 = new Complex(2, 3.5);
Complex z3 = new Complex( z2 );
```

Constructor Methods

- Constructor เรียกกันเองได้ โดยใช้ this

```
public class Complex {
  public double x, y;

  public Complex( ) {
    Complex(0, 0);          // WRONG
  }
  public Complex( double x1, double y1 ) {
    this
    this
  }
```

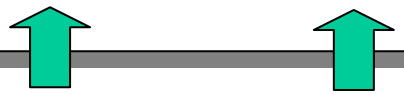
```
public class Complex {
  public double x, y;

  public Complex( ) {
    this(0, 0);             // CORRECT
  }
  public Complex( double x1, double y1 ) {
    this.x = x1;
    this.y = y1;
  }
```

เรื่องน่ารู้

- Utility class คือคลาสที่มีแต่ class methods (เช่น `Math.*`, `Arrays.*`, `JLabIO.*`)
- คลาสไม่จำเป็นต้องมีเมท็อด main เมื่อเป็นคลาสที่ผู้ใช้ไม่ได้เรียกใช้งานโดยตรง
- ถ้าคลาสไม่มี constructor ระบบจะเติมให้แบบไม่มีการรับพารามิเตอร์
- ตั้งชื่อคลาสเป็น noun ตั้งชื่อเมท็อดเป็น verb
- ตัด this. ทิ้งก็ได้

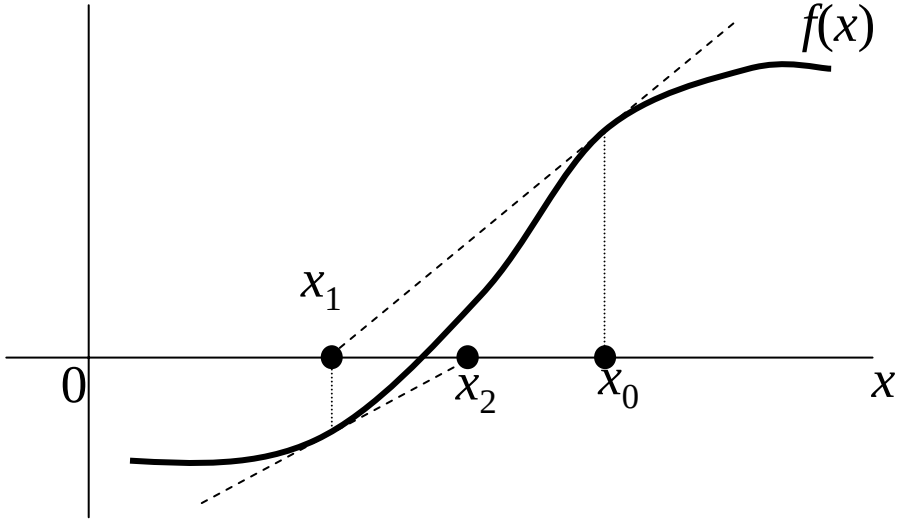
```
public class Complex {
  public double x, y;
  ...
  public boolean equals( Complex z ) {
    return x == z.x && y == z.y;
  }
```



ตัวอย่าง : หารากด้วย Newton-Raphson

- ต้องการหารากของสมการ f(x) = 0
- หาได้เฉพาะรากที่เป็นจำนวนจริง
- วิธีของ Newton-Raphson
  - ต้องรู้อนุพันธ์ของ f(x)
  - ถ้า x_k คือค่าประมาณของราก จะสามารถคำนวณค่าประมาณของราก (ที่ดีกว่า) x_{k+1} ดังนี้

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$



Class NewtonRaphson

```
public class NewtonRaphson {
  public static double f (double x) {
    return x * x + 4 * x + 4;
  }
  public static double df (double x) {
    return 2 * x + 4;
  }
  public static void solve (double xnew, double tol, int imax) {
    double xold;
    int i;
    for (i = 0; i < imax; i++) {
      xold = xnew;
      xnew = xold - f(xold) / df(xold);;
      System.out.println("x = " + xnew);
      if (Math.abs(xnew - xold) < tol) break;
    }
    if (i >= imax) System.out.println("not converge");
  }
  public static void main(String[] args) {
     solve(0, 1e-10, 100);
  }
}
```

$$f(x) = x^2 + 4x + 4$$

$$f'(x) = 2x + 4$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

หาราก (เป็นจำนวนเชิงซ้อนได้)

- เขียนคลาส Complex ให้มี
  - data members : x และ y
  - constructor
  - object methods :
    add, subtract, multiply, divide, abs, toString
- เขียนคลาส NewtonRaphsonComplex

Class Complex

```
public class Complex {
  public double x, y;

  public Complex(double x1, double y1) {
    x = x1; y = y1;
  }
  public String toString() {
    if (y >= 0) return "(" + x + " + i" + y + ")";
    else        return "(" + x + " - i" + (-y) + ")";
  }
  public Complex multiply(Complex z) {
    return new Complex(x * z.x - y * z.y, x * z.y + y * z.x);
  }
  public Complex divide(Complex z) {
    double abs2 = z.x * z.x + z.y * z.y;
    return new Complex((x * z.x + y * z.y) / abs2,
                       (z.x * y - z.y * x) / abs2);
  }
  public double abs( ) {
    return Math.sqrt(x * x + y * y);
  }
  ...
```

เขียนเมท็อด add และ subtract เอง

Class NewtonRaphsonComplex

```
public class NewtonRaphsonComplex {

  public static Complex f(Complex x) {
    Complex two = new Complex(2, 0);
    Complex c1 = x.multiply(x);
    Complex c2 = x.multiply(two);
    return c1.add(c2.add(two));
  }

  public static Complex df(Complex x) {
    Complex two = new Complex(2, 0);
    Complex c2 = x.multiply(two);
    return c2.add(two);
  }
```

$$f(x) = x^2 + 2x + 2$$

$$f'(x) = 2x + 2$$

ยังมีต่อ...

Class NewtonRaphsonComplex

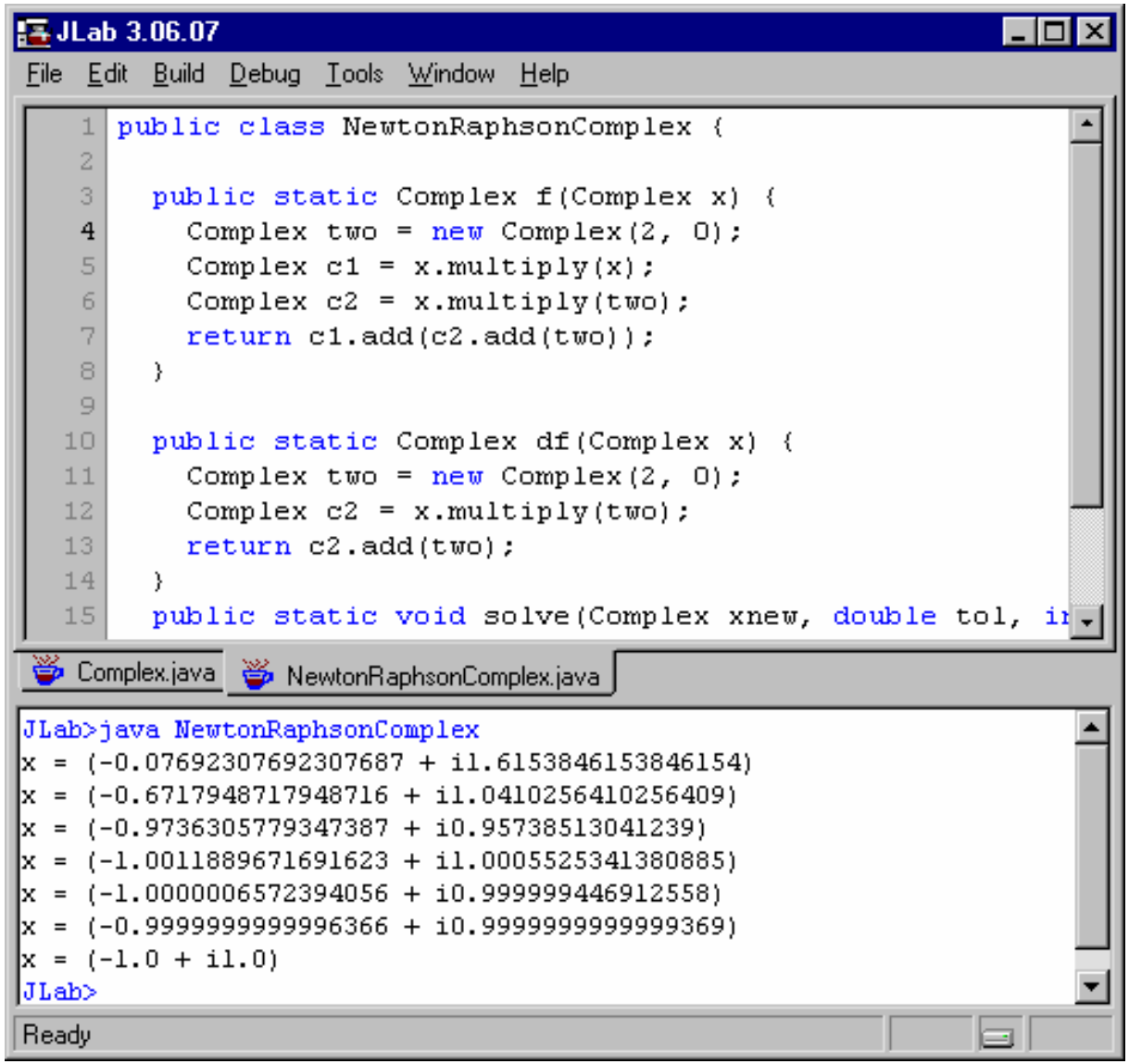
...เขียนต่อ

```
  public static void solve(Complex xnew, double tol, int imax) {
    Complex xold, temp;
    int i;
    for (i = 0; i < imax; i++) {
      xold = xnew;
      temp = f(xold).divide(df(xold));
      xnew = xold.subtract(temp);
      System.out.println("x = " + xnew);
      temp = xnew.subtract(xold);
      if (temp.abs() < tol) break;
    }
    if (i >= imax) System.out.println("not converge");
  }

  public static void main(String[] args) {
     solve(new Complex(1, 3), 1e-10, 100);
  }
}
```

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Edit + Compile + Run



คำถาม

```
Complex c1 = new Complex(5,0);
Complex c2 = new Complex(4,0);
Complex c3 = new Complex(3,0);

Complex c4 = c1.add(c2).subtract(c3);
Complex c5 = c1.multiply(c2).add(c3);
Complex c6 = c1.add(c2).multiply(c3);
Complex c7 = c1.add( c2.multiply(c3) );
```

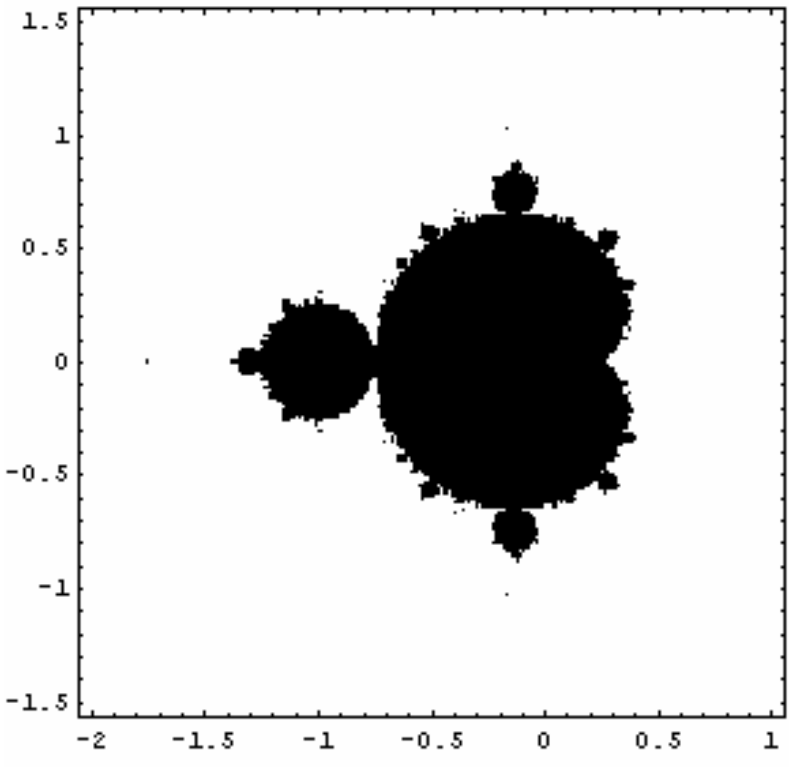
c4, c5, c6, c7 = ?

```
public Complex {
  ...
  public Complex sqr( ) {
    return                    ;
  }
  ...
}
```

จงเขียนเมท็อดยกกำลังสองในคลาส Complex โดยการใช้เมท็อด multiply

Mandelbrot Set (อีกครั้ง)

- 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

โปรแกรมแสดง Mandelbrot set

```
public void paint(Graphics g) {
  int n;
  Complex c, d;
  double stepX = 3.0 / getSize().width;
  double stepY = 3.0 / getSize().height;
  for (double y = -1.5; y <= 1.5; y += stepY) {
    for (double x = -2.0; x <= 1.0; x += stepX) {
      c = new Complex(x, y);
      d = new Complex(0, 0);
      n = 0;
      while (d.abs() < 2 && n++ < 256) {
        d = d.multiply(d).add(c);
      }
      if (n >= 256) {
        g.drawRect((int)((x + 2) / stepX),
                   (int)((y + 1.5) / stepY), 1, 1);
      }
    }
  }
}
```

$|d_n| < 2$

$d_n = (d_{n-1})^2 + c$

แบบฝึกหัด : อยากได้คลาส "Rational"

- Rational เป็นคลาสเก็บเลขตรรกยะ
  - ต.ย. 3/4, 15/7, 8/1
- มีสอง fields : numerator และ denominator
  - ต.ย. 15/7 มี numerator เป็น 15 และ denominator เป็น 7
- มีเมท็อดดังนี้
  - add, subtract, multiply, divide, toString, และ equals
- ควรเก็บแบบลดรูป เช่น 4/8 ควรเก็บเป็น 1/2

```
public class Rational {
  public int numerator;
  public int denominator;
  ...
}
```