คลาสและอ็อบเจกต์

ที่ผ่านมา : คลาสคือโปรแกรม

```
public class ClassGPA {
   public static void main(...) {
    ...
   }

   เมท็อดอื่น ๆ
   ทุกเมท็อดมีคำว่า static
   (เมท็อดประจำคลาส)

}
```

class methods
static methods

ยังมีอีกแบบ : คลาสคือประเภทข้อมูล

```
public class Ball {
  ...
}
```

```
public class Product {
  ...
}
```

```
public class Point {
  ...
}
```

```
public class ComplexNumber {
  ...
}
```

```
public class Rectangle {
  ...
}
```

```
public class Date {
  ...
}
```

```
public class Employee {
  ...
}
```

```
public class Currency {
  ...
}
```

```
public class ClassRoom {
  ...
}
```

```
public class BankAccount {
  ...
}
```


ตัวอย่างคลาสที่เป็นประเภทข้อมูล : Ball

```
public class Ball {
   ตัวแปรประจำอ็อบเจกต์
   object variables

   ตัวสร้าง
   constructors

   เมท็อดประจำอ็อบเจกต์
   object methods

   ตัวแปรประจำคลาส
   class variables
   เมท็อดประจำคลาส
}  class method
```

แบบง่ายสุด : มีเฉพาะข้อมูลย่อย

```
public class Ball {
    ตัวแปรประจำอ็อบเจกต์
    object variables
}
```

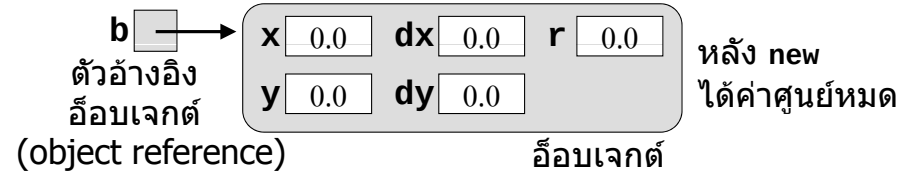
data members
attributes
fields

บรรยายเฉพาะข้อมูลย่อยภายใน

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;
}
```


สร้างอ็อบเจกต์ด้วย new

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;
}
```



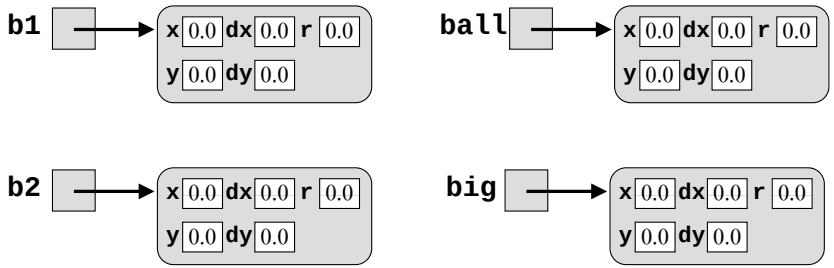
ค่าเริ่มต้นของตัวแปรประจำอ็อบเจกต์

- หลัง **new**, ค่าของตัวแปรประจำอ็อบเจกต์
  - จำนวน มีค่า **0**
  - **boolean** มีค่า **false**
- ให้ค่าอื่นตอนประกาศ object variable ได้

```
public class Ball {
  public double x, y;
  public double dx = 2, dy = 4;
  public double r = 10;
}
```


แต่ละอ็อบเจกต์มีตัวแปรประจำอ็อบเจกต์ของตัวเอง

```
Ball b1   = new Ball();
Ball b2   = new Ball();
Ball ball = new Ball();
Ball big  = new Ball();
```



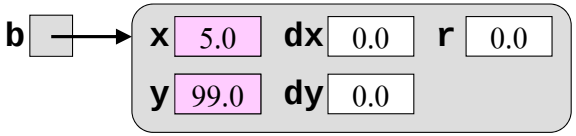
b1, b2, ball, big เป็นตัวอ้างอิงอ็อบเจกต์แบบ **Ball**
บางครั้งเราเรียก "อ็อบเจกต์ **b1**" แทนอ็อบเจกต์ที่ **b1** อ้างอิงอยู่

การเปลี่ยนค่าของตัวแปรประจำอ็อบเจกต์

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;
}
```

```
Ball b = new Ball();
b.x = 5;
b.y = 99;
```

b.x อ่านว่า ตัวแปร **x** ของอ็อบเจกต์ **b**

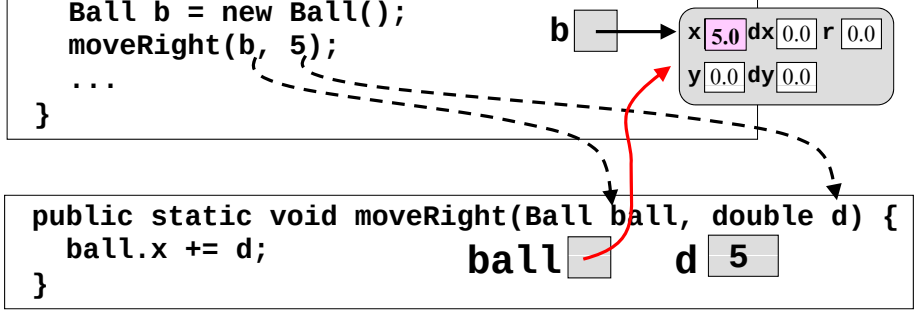


เมท็อดที่รับอ็อบเจกต์

- การส่งอ็อบเจกต์ให้เมท็อด คือการส่งตำแหน่งของอ็อบเจกต์ (ค่าของตัวอ้างอิง)
- ผู้ส่งกับผู้รับจึงอ้างอิงอ็อบเจกต์เดียวกัน

```
public static void main(String[] args) {
  Ball b = new Ball();
  moveRight(b, 5);
  ...
}
```

```
public static void moveRight(Ball ball, double d) {
  ball.x += d;
}
```

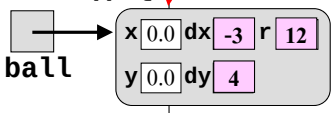


เมท็อดที่คืนอ็อบเจกต์

- การคืนอ็อบเจกต์ให้ผู้เรียกเมท็อด คือการส่งตำแหน่งของอ็อบเจกต์คืนกลับไป

```
public static void main(String[] args) {
  Ball b = createRandomBall();
  ...
}
```

```
public static Ball createRandomBall() {
  Ball ball = new Ball();
  ball.r = random(10, 15);
  ball.dx = random(-5, 5);
  ball.dy = random(-5, 5);
  return ball;
}
```



ตัวแปรในเมท็อดหาย เมื่อเมท็อดทำงานเสร็จ แต่อ็อบเจกต์ที่สร้างในเมท็อดไม่หาย ถ้ายังมีตัวแปรอ้างอิงมันอยู่

ตัวอย่าง : ลูกบอลเด้งในวินโดว์

```
public static void main(String[] args) {
  DWindow w = new DWindow(250, 200);
  Ball b1 = createRandomBall();
  b1.x = 125; b1.y = 100;
  while (true) {
    w.fade(0.3);
    draw(w, b1);
    move(w, b1);
    w.sleep(30);
  }
}
public static void draw(DWindow w, Ball b) {
  w.fillEllipse(b.x, b.y, 2 * b.r, 2 * b.r);
}
public static void move(DWindow w, Ball b) {
  b.x = b.x + b.dx;
  b.y = b.y + b.dy;
  if (b.x + b.r > w.getWidth()  || b.x < b.r) b.dx = -b.dx;
  if (b.y + b.r > w.getHeight() || b.y < b.r) b.dy = -b.dy;
}
```

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;
}
```

ให้คลาส Ball รับผิดชอบการ draw และ move

```
DWindow w = new DWindow(250, 200);
Ball b1 = createRandomBall();
b1.x = 125; b1.y = 100;
while (true) {
  w.fade(0.3);
  Ball.draw(w, b1);
  Ball.move(w, b1);
  w.sleep(30);
}
```

เรียก **draw** ที่เขียนในคลาส **Ball**

```
public class Ball {
  public double x, y, dx, dy, r;

  public static void draw(DWindow w, Ball b) {
    w.fillEllipse(b.x, b.y, 2 * b.r, 2 * b.r);
  }
  public static void move(DWindow w, Ball b) {
    b.x = b.x + b.dx;
    b.y = b.y + b.dy;
    if (b.x + b.r > w.getWidth()  || b.x < b.r) b.dx = -b.dx;
    if (b.y + b.r > w.getHeight() || b.y < b.r) b.dy = -b.dy;
  }
}
```


เขียน draw ในคลาส Ball ได้สองแบบ

class method

```
public class Ball {
  public double x, y, dx, dy, r;

  public static void draw(DWindow w, Ball b) {
    w.fillEllipse(b.x, b.y, 2 * b.r, 2 * b.r);
  }
```

```
Ball.draw(w, b1);
```

เรียก **draw** ที่เขียนในคลาส **Ball**

ต้องรับ **Ball b** มา เพื่อจะใช้ค่าต่าง ๆ ของอ็อบเจกต์ **b**

object method

```
public class Ball {
  public double x, y, dx, dy, r;

  public void draw(DWindow w) {
    w.fillEllipse(x, y, 2 * r, 2 * r);
  }
```

```
b1.draw(w);
```

b1 เป็นอ็อบเจกต์ที่ถูกเรียกใช้บริการ **draw**

ตัวแปรเหล่านี้คือตัวแปรของอ็อบเจกต์ที่ถูกเรียก

เรียก **b1.draw(w), x** ก็คือ **b1.x**

class method กับ object method

- class method
  - มีคำว่า static ที่หัวเมท็อด
  - รูปแบบการเรียก : *ชื่อคลาส*. *ชื่อเมท็อด*(...)
- object method
  - ไม่มีคำว่า static ที่หัวเมท็อด
  - รูปแบบการเรียก : *ตัวอ้างอิงอ็อบเจกต์*. *ชื่อเมท็อด*(...)

บริการที่เกี่ยวกับอ็อบเจกต์มักเขียนเป็น object method

```
public class Ball {
  public double x, y, dx, dy, r;

  public static void draw(DWindow w, Ball b) {
    w.fillEllipse(b.x, b.y, 2 * b.r, 2 * b.r);
  }
```

class method

```
public class Ball {
  public double x, y, dx, dy, r;

  public void draw(DWindow w) {
    w.fillEllipse(x, y, 2 * r, 2 * r);
  }
```

object method

องค์ประกอบของคลาส : ตัวสร้าง (constructor)

```
Ball b = new Ball();
b.x = 50; b.y = 50;
b.dx = 2; b.dy = 3;
b.r = 20;
```

```
Ball b = new Ball(50, 50, 2, 3, 20);
```

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;

  public Ball(double x1, double y1,
              double dx1, double dy1, double r1) {
    x = x1; y = y1; dx = dx1; dy = dy1; r = r1;
  }
  ...
```

ชื่อต้องเหมือนชื่อคลาส
ไม่มี static
ไม่มี ประเภทผลลัพธ์

หนึ่งคลาสมีตัวสร้างหลายแบบก็ได้

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;

  public Ball() {
  }
  public Ball(double x1, double y1) {
    x = x1; y = y1; r = 10;
  }
  public Ball(double x1, double y1, double r1) {
    x = x1; y = y1; r = r1;
  }
  public Ball(double x1, double y1,
              double dx1, double dy1, double r1) {
    x = x1; y = y1; dx = dx1; dy = dy1; r = r1;
  }
  ...
```


Default Constructor (ตัวสร้างที่ระบบเติมให้ ถ้าไม่ได้เขียน)

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;
}
```

ไม่เขียนตัวสร้าง, compiler จะเติมให้อัตโนมัติ

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;

  public Ball() { }
}
```

```
public class Ball {
  public double x, y;
  public double dx, dy;
  public double r;

  public Ball(double x1, double y1) {
    x = x1; y = y1; r = 10;
  }
```

เขียนตัวสร้างแล้ว compiler ไม่เติมอีก

ตัวอย่าง : BankAccount

```
public class BankAccount {
  public String id = "";
  public double balance = 0;

  public BankAccount(String newID, double initialBalance) {
    id = newID;
    balance = initialBalance;
  }
  public void withdraw(double amt) {
    if (amt > 0 && balance >= amt) balance = balance - amt;
  }
  public void deposit(double amt) {
    if (amt > 0) balance = balance + amt;
  }
}
```

```
BankAccount ba = new BankAccount("A11", 100);
ba.deposit(200);
ba.withdraw(50);
System.out.println(ba.balance);
ba.id = "99-99-9999-9";   // !!!
ba.balance = -1e6;        // !!!
```


public ไม่ปลอดภัย, private ป้องกันได้

```
public class BankAccount {
  private String id = "";
  private double balance = 0;

  public BankAccount(String newID, double initialBalance) {
    id = newID;
    balance = initialBalance;
  }
  public void withdraw(double amt) {
    if (amt > 0 && balance >= amt) balance = balance - amt;
  }
  public void deposit(double amt) {
    if (amt > 0) balance = balance + amt;
  }
}
```

ใช้สมาชิกที่เป็น private ได้เฉพาะในคลาสที่เขียนเท่านั้น

```
BankAccount ba = new BankAccount("A11", 100);
ba.id = "99-99-9999-9";
ba.balance = -1e6;
```

แปลไม่ผ่าน

อาจต้องให้บริการขอข้อมูลที่เป็น private

```
public class BankAccount {
  private String id;
  private double balance;
  ...
  public String getID() {
    return id;
  }
  public double getBalance() {
    return balance;
  }
```

```
BankAccount ba = new BankAccount("02-125-872", 0);
ba.deposit(100);
System.out.println(ba.getID() + "," +
                   ba.getBalance());
```


ตัวอย่าง : Dice

- เขียนคลาสเพื่อสร้างลูกเต๋า ให้บริการดังนี้
  - **`int roll()`** : โยนลูกเต๋า แล้วคืนหมายเลขหน้าที่ได้
  - **`int getValue()`** : คืนหมายเลขหน้าของลูกเต๋า

```
public class Dice {
  private int value;
  public Dice() {
    roll();
  }
  public int roll() {
    value = 1 + (int)(6 * Math.random());
    return value;
  }
  public int getValue() {
    return value;
  }
}
```


ตัวอย่าง : PiggyBank

- เขียนคลาสเพื่อสร้างกระปุกออมสินเก็บได้เฉพาะเหรียญ 1 บาท, 2 บาท, 5 บาท และ 10 บาท ให้บริการดังนี้
  - **`PiggyBank()`** : ตัวสร้างกระบุกออมสินเปล่า ๆ
  - **`void add1(int c)`** : หยอดเหรียญ 1 บาท **c** เหรียญ
  - **`void add2(int c)`** : หยอดเหรียญ 2 บาท **c** เหรียญ
  - **`void add5(int c)`** : หยอดเหรียญ 5 บาท **c** เหรียญ
  - **`void add10(int c)`** : หยอดเหรียญ 10 บาท **c** เหรียญ
  - **`void clear()`**: เทเหรียญทั้งหมดออกจากกระปุก
  - **`int getTotal()`** : คืนจำนวนเงินทั้งหมดที่เก็บในกระปุก

```
PiggyBank pb = new PiggyBank();
pb.add1(3); pb.add2(1); pb.add5(2); pb.add10(1);
pb.add1(2);  System.out.println(pb.getTotal());
pb.clear();  System.out.println(pb.getTotal());
```


ตัวอย่าง : PiggyBank : ตัวแปรประจำอ็อบเจกต์

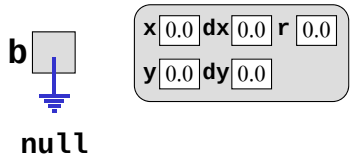
```
public class PiggyBank {
  private int one, two, five, ten;

  public PiggyBank() {
    clear();
  }
  public void clear() {
    one = two = five = ten = 0;
  }
  public void  add1(int coins) {one  = one + coins;}
  public void  add2(int coins) {two  = two + coins;}
  public void  add5(int coins) {five = five + coins;}
  public void add10(int coins) {ten  = ten + coins;}
  public int getTotal() {
    return one + 2*two + 5*five + 10*ten;
  }
}
```

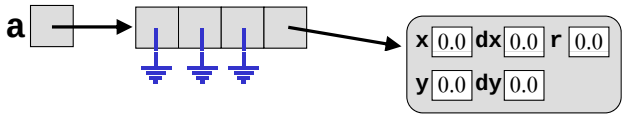
ลองปรับปรุงให้กระปุกมีความจุ จะไม่เพิ่มให้ถ้าเต็มแล้ว

null : ค่าที่แทนการไม่อ้างอิงอ็อบเจกต์ใด

```
Ball b = new Ball();
b = null;
```



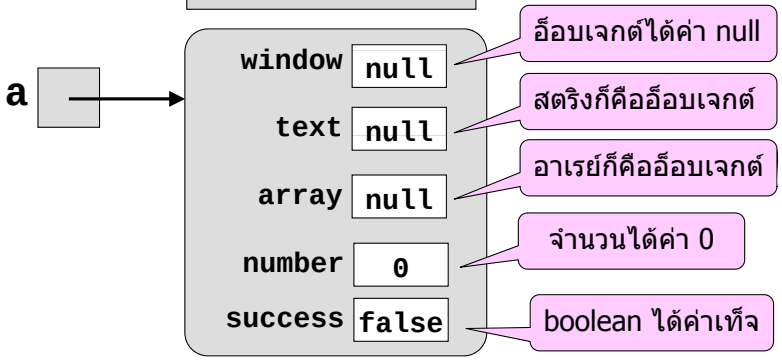

```
Ball[] a = new Ball[4]; // สร้างแค่อาเรย์
a[3] = new Ball();
```



ค่าเริ่มต้นของตัวแปรประจำอ็อบเจกต์

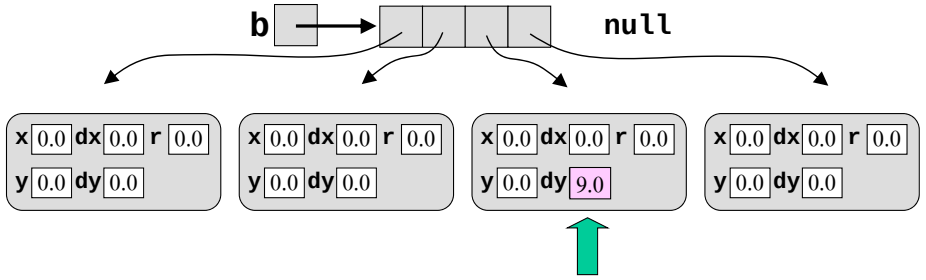
```
public class A {
  private DWindow window;
  private String text;
  private int[] array;
  private int number;
  private boolean success;
  ...
}
```

```
A a = new A();
```



อาเรย์ของอ็อบเจกต์

```
Ball[] b = new Ball[4]; // สร้างแค่อาเรย์
for (int i = 0; i < b.length; i++) {
  b[i] = new Ball();
}
b[2].dy = 9;
```



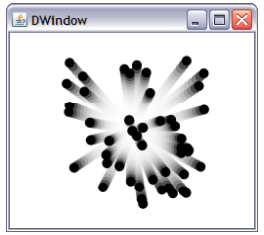
ตัวอย่าง : ลูกบอล 50 ลูก (ใช้อาเรย์ 5 แถว)

```
DWindow w = new DWindow(250, 200);
int n = 50;
double[] x = new double[n], y = new double[n];
double[] r = new double[n];
double[] dx = new double[n], dy = new double[n];
for (int i = 0; i < n; i++) {
  x[i] = 125; y[i] = 100; r[i] = 5;
  dx[i] = -4 + 8 * Math.random();
  dy[i] = -4 + 8 * Math.random();
}
while (true) {
  w.fade(0.3);
  for (int i = 0; i < n; i++) {
    w.fillEllipse(x[i], y[i], 2 * r[i], 2 * r[i]);
    x[i] = x[i] + dx[i];  y[i] = y[i] + dy[i];
    if (x[i] + r[i] > 250) dx[i] = -dx[i];
    if (x[i] - r[i] <   0) dx[i] = -dx[i];
    if (y[i] + r[i] > 200) dy[i] = -dy[i];
    if (y[i] - r[i] <   0) dy[i] = -dy[i];
  }
  w.sleep(30);
}
```

สร้างอาเรย์

ให้ค่าเริ่มต้น

วาดและปรับค่า



คลาส Ball อีกครั้ง

```
public class Ball {
  public double x, y, dx, dy, r;

  public Ball(double x1, double y1,
              double dx1, double dy1, double r1) {
    x = x1; y = y1; dx = dx1; dy = dy1; r = r1;
  }
  public void draw(DWindow w) {
    w.fillEllipse(x, y, 2 * r, 2 * r);
  }
  public void move(DWindow w) {
    x = x + dx;
    y = y + dy;
    if (x + r > w.getWidth()  || x < r) dx = -dx;
    if (y + r > w.getHeight() || y < r) dy = -dy;
  }
}
```


ตัวอย่าง : ลูกบอล 50 ลูก (ใช้อาเรย์ของ Ball 1 แถว)

```
public static void main(String[] args) {
  DWindow w = new DWindow(250, 200);
  Ball[] b = new Ball[50];
  for (int i = 0; i < b.length; i++) {
    b[i] = new Ball(125, 100,
                    random(-5, 5), random(-5, 5), 10);
  }
  while (true) {
    w.fade(0.3);
    for (int i = 0; i < b.length; i++) {
      b[i].draw(w);
      b[i].move(w);
    }
    w.sleep(30);
  }
}
public static double random(int a, int b) {
  return a + (b - a) * Math.random();
}
```

เรียก **move** ของลูกบอล **b[i]**

ตัวอย่าง : Card (ไพ่ 1 ใบ)

```
public class Card {
  private String suit; // โพธิ์ดำ โพธิ์แดง หลามตัด หัวใจ
  private int rank;    // 2, 3, ..., 14

  public Card(int r, String s) {
    suit = s;
    rank = r;
  }
  public int getRank() {
    return rank;
  }
  public String getSuit() {
    return suit;
  }
}
```

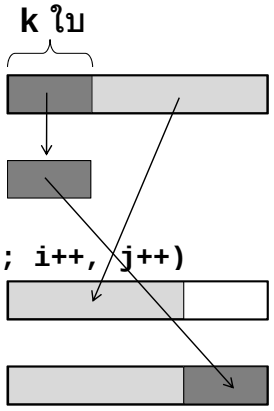
ตัวอย่าง : Deck (ไพ่ 1 สำรับ)

```
public class Deck {
  private Card[] deck;
  private int numberOfCards;

  public Deck() {
    deck = new Card[52];
    int k = 0;
    for (int i = 2; i <= 14; i++) {
      deck[k++] = new Card(i, "โพธิ์ดำ");
      deck[k++] = new Card(i, "โพธิ์แดง");
      deck[k++] = new Card(i, "หลามตัด");
      deck[k++] = new Card(i, "หัวใจ");
    }
    numberOfCards = 52;
  }
  public Card nextCard() {
    numberOfCards--;
    return deck[numberOfCards];
  }
```

ตัวอย่าง : Deck : ตัดไพ่

```
public class Deck {
  private Card[] deck;
  private int numberOfCards;
  // ...
  public void cut(int k) {
    Card[] temp = new Card[k];
    for (int i = 0; i < k; i++)
      temp[i] = deck[i];
    int j = 0;
    for (int i = k; i < numberOfCards; i++, j++)
      deck[j] = deck[i];
    for (int i = 0; i < k; i++, j++)
      deck[j] = temp[i];
  }
```






องค์ประกอบของคลาส : ตัวแปรประจำคลาส

- ตัวแปรประจำอ็อบเจกต์
  - แต่ละอ็อบเจกต์มีที่เก็บเป็นของตัวเอง เช่น ball.dx
    (ต้องมีอ็อบเจกต์ถึงจะมีตัวแปรแบบนี้)
- เมท็อดประจำอ็อบเจกต์
  - บริการที่อ็อบเจกต์มีให้เรียกใช้ เช่น account.deposit(100)
    (ต้องมีอ็อบเจกต์ถึงจะเรียกเมท็อดแบบนี้ได้)
- เมท็อดประจำคลาส
  - เรียกใช้ได้เลยโดยไม่ต้องสร้างอ็อบเจกต์ Math.sin(x)
    (นำหน้าชื่อเมท็อดด้วยชื่อคลาส)
- ตัวแปรประจำคลาส
  - ที่เก็บข้อมูลประจำตัวคลาส
    (ไม่ต้องสร้างอ็อบเจกต์ก็มีที่เก็บนี้ให้ใช้)

ใส่ static กำกับตัวแปรประจำคลาส

```
public class DayOfWeek {
  public static String getDayOfWeek(int d) {
    String[] dow = {"เสาร์","อาทิตย์","จันทร์",
                    "อังคาร","พุธ","พฤหัสบดี","ศุกร์"};
    return dow[d];
  }
```

อาเรย์นี้ถูกสร้างทุกครั้งที่ getDayOfWeek ถูกเรียก

อาเรย์นี้ถูกสร้างครั้งเดียวที่คลาส DayOfWeek ถูกเรียกใช้

```
public class DayOfWeek {
  private static String[] dow =
   {"เสาร์","อาทิตย์","จันทร์","อังคาร","พุธ","พฤหัสบดี","ศุกร์"};

  public static String getDayOfWeek(int d) {
    return dow[d];
  }
```


ใส่ static กำกับตัวแปรประจำคลาส

```
public class Coin {
  private static int numberOfCoins = 0;
  private int value;

  public Coin(int v) {
    value = v;
    numberOfCoins++;
  }
  public int getValue() {
    return value;
  }
  public static int getNumberOfCoins() {
    return numberOfCoins;
  }
  ...
}
```

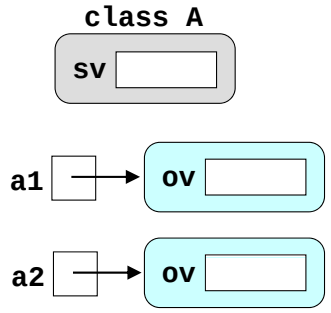
ถ้าลบ static ออก จะเกิดอะไรขึ้น ?

นับจำนวนเหรียญที่ได้ผลิตมา

ตัวอย่างทดสอบความเข้าใจ : class & obj. var.

```
public class A {
 private static int sv;
 private        int ov;
 ...
}
```

```
A.sv = 1;
A a1 = new A();
A a2 = new A();
A.sv++;
a1.ov++;
a2.ov++
a1.sv++;
a2.sv++;
A.ov = 9 // wrong
```



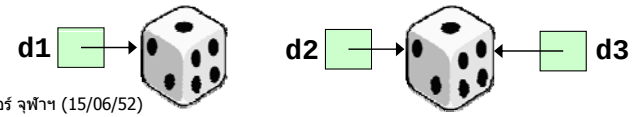
ตัวอย่างที่เราเคยใช้มา

- Class methods
  - Math.sin(x), Math.random(), ...
- Class variables
  - System.in, System.out, DWindow.RED, ...
- Object methods
  - kb.nextInt(), System.out.println(), w.sleep(30), ...
- Object variables
  - มีน้อย เพราะมักให้เป็น private

การใช้ == กับอ็อบเจกต์

- ถ้า a และ b เป็นตัวแปรอ้างอิงอ็อบเจกต์
- a == b เป็นการทดสอบว่า
  - a และ b อ้างอิงตำแหน่งเดียวกันหรือไม่
  - a และ b อ้างอิงอ็อบเจกต์ตัวเดียวกันหรือไม่
- การเปรียบเทียบความเท่ากันของสตริง จึงไม่ใช้ == (`"a" == "A".toLowerCase()` ได้ `false`)
  - ต้องใช้เป็นเมท็อดแทน (`s1.equals(s2)`)

```
Dice d1 = new Dice();
Dice d2 = new Dice();
Dice d3 = d2;
System.out.println( (d1 == d2) ); //false
System.out.println( (d2 == d3) ); // true
```



equals ของ Dice

```
public class Dice {
  private int value;
  public Dice() { roll(); }
  public int getValue() { return value; }
  public int roll() {
    value = 1 + (int)(6 * Math.random());
    return value;
  }
  public boolean equals(Dice d) {
    return value == d.getValue();
  }
}
```

```
Dice d1 = new Dice();
Dice d2 = new Dice();
int c = 1;
while (!d1.equals(d2)) {
  d1.roll();
  d2.roll();
  c++;
}
System.out.println( c );
```