ตัวสร้างและตัวทำลาย
(Constructor and Destructor)

Benjamas Panyangam
Matinee Kiewkanya
Computer Science, CMU

ตัวสร้าง (Constructor)

Constructor คือ เมท็อดที่จะถูกเรียกใช้โดยอัตโนมัติทุกครั้งที่มีการสร้างวัตถุของคลาสขึ้นมา โดยทั่วไปจะมีหน้าที่เพื่อกำหนดค่าเริ่มต้นให้แก่วัตถุ โดยเมท็อด Constructor ในภาษาจาวา จะมีลักษณะดังนี้

- **มีชื่อเดียวกันกับคลาส**
- **ไม่มีการส่งค่ากลับ โดยไม่ต้องระบุคำว่า void ไว้ด้านหน้า**
- **สามารถรับค่าพารามิเตอร์ได้**
- **แต่ละคลาสจะมี Constructor หรือไม่มีก็ได้**
- **ในหนึ่งคลาสสามารถมี Constructor ได้หลายเมท็อด เรียกว่า Overloaded Constructor**

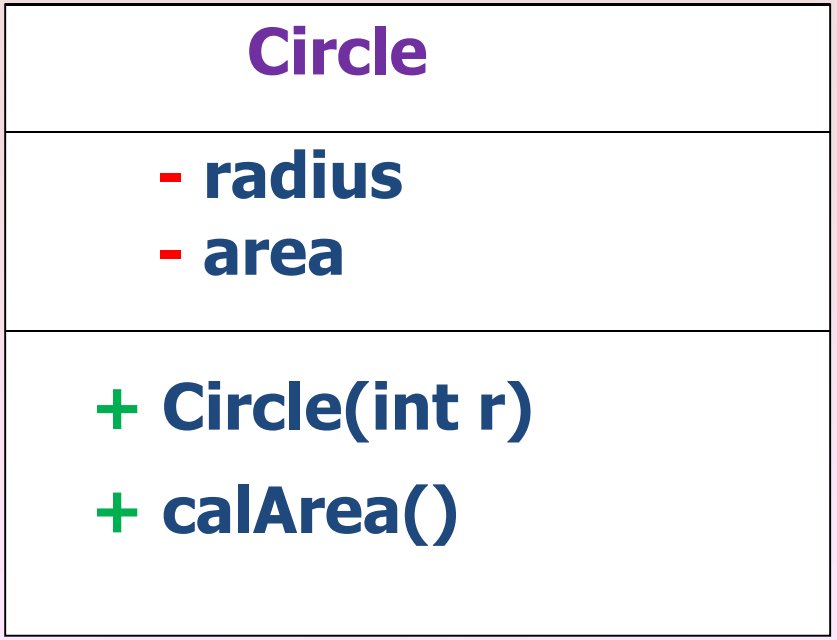
- ในกรณีที่โปรแกรมไม่มีเมท็อด Constructor คอมไพเลอร์จะสร้างให้โดยอัตโนมัติ เรียกว่า Empty Constructor ซึ่งจะเป็น Constructor ที่ไม่มีพารามิเตอร์ และส่วนของการทำงานจะว่างเปล่า คือไม่ทำงานใดเลย

- การสร้างวัตถุด้วยคำสั่ง new จะต้องมีเมท็อด Constructor รองรับทุกรูปแบบ โดยจะเรียกใช้ เมท็อด Constructor ตามรูปแบบที่ประกาศไว้
  เช่น
  - Circle myCircle = new Circle(); จะเรียกใช้ Empty Constructor (แบบ 0 parameter)
  - Circle myCircle = new Circle(2); จะเรียกใช้ Constructor แบบ 1 parameter

- ดังนั้นถ้าโปรแกรมไม่มี Constructor เลย จะสามารถ new วัตถุได้รูปแบบเดียว คือแบบที่ไม่มีพารามิเตอร์

ตัวอย่างโปรแกรมที่ 1

โปรแกรมเพื่อแสดงการทำงานของเมท็อด Constructor ของคลาส Circle กรณีสร้างวัตถุ 1 วัตถุ



โปรแกรม

```
class Circle {
    private  int radius;
    private float area;
    public Circle(int r){
        radius  = r;
        System.out.print("\n *** I'm in the constructor
                method!!! ***\n");
    }
    public void calArea(){
        area = 3.14f * radius * radius;
          System.out.println("Area: " + area);
    }
}
```

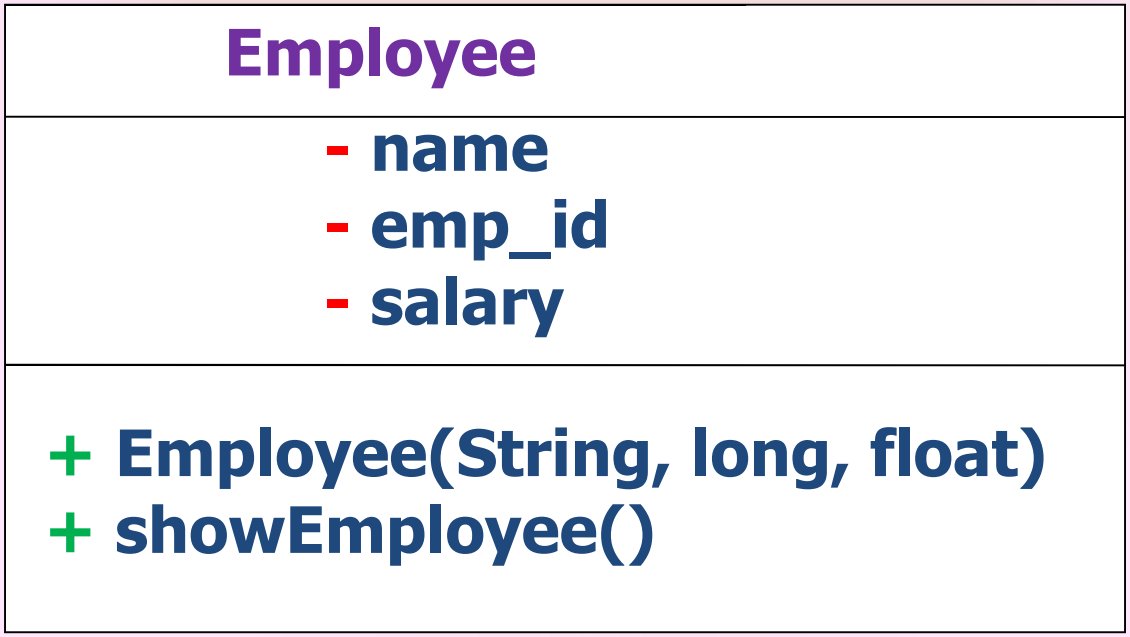
```
public class JavaApp1 {
        public static void main(String[] args) {
                Circle myCircle = new Circle(2);
                myCircle.calArea();
        }
}
```

ผลการทำงาน

```
*** I'm in the constructor method!!! ***
Area: 12.56
```

ตัวอย่างโปรแกรมที่ 2

โปรแกรมเพื่อแสดงการทำงานของเมท็อด Constructor ของคลาส Employee กรณีสร้างวัตถุหลายวัตถุ



โปรแกรม

```
class Employee {
        private String name;
        private long emp_id;
        private float salary;
        public Employee (String  n, long  i, float s){
                name = n;
                emp_id= i;
                salary=s;
                System.out.println("I am "+ n );
        }
         public void showEmployee() {
                System.out.println("Employee: " + name);
                System.out.println("Id: "+ emp_id );
                System.out.println("Salary: "+ salary + "\n");
        }
}
```

```
public class JavaApp2 {
    public static void main(String[] args) {
        //สร้างวัตถุที่ 1
        Employee worker1=new Employee("Matinee
                                      Kiewkanya",1001,10000.0f);
        worker1.showEmployee();

        //สร้างวัตถุที่ 2
        Employee worker2=new Employee("Benjamas
                                      Panyangam", 1002,200000.0f);
        worker2.showEmployee();
    }
}
```

ผลการทำงาน

```
I am Matinee Kiewkanya
Employee: Matinee Kiewkanya
Id: 1001
Salary: 10000.0

I am Benjamas Panyangam
Employee: Benjamas Panyangam
Id: 1002
Salary: 200000.0
```

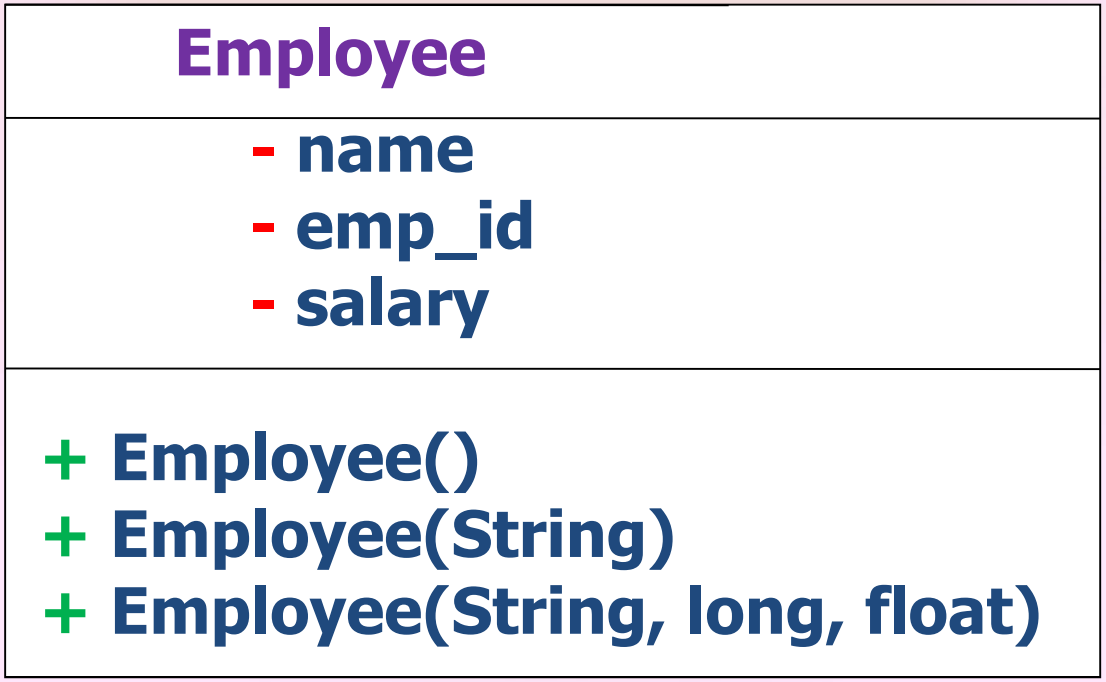
การพ้องชื่อของตัวสร้าง (Constructor Overloading)

- **Method Signature** หมายถึง รายการพารามิเตอร์ของเมท็อด อันประกอบด้วยจำนวนของพารามิเตอร์และชนิดของพารามิเตอร์
- ในกรณีที่คลาสหนึ่งคลาสมีเมท็อด **Constructor** หลายเมท็อด จะทำให้เกิดการพ้องชื่อของตัวสร้าง **(Constructor Overloading)** โดยที่แต่ละเมท็อดจะต้องมี **Method Signature** ที่แตกต่างกัน

- ตัวอย่าง Constructor Overloading ของคลาส Employee เช่น
  - Employee () ;
  - Employee(String n);
  - Employee(String n, long i, float s);
- เมื่อมีการสร้างวัตถุขึ้นมา การที่จะเรียกใช้เมท็อด Constructor ใด จะพิจารณาจากรายการของพารามิเตอร์ที่ส่งมาให้แก่เมท็อด

ตัวอย่างโปรแกรมที่ 3

โปรแกรมเพื่อแสดง Constructor Overloading ของคลาส Employee



โปรแกรม

```
class Employee {
        private String name;
        private long emp_id;
        private float salary;
        public Employee(){              //constructor ที่ 1
           System.out.println("I am the first constructor");
        }
        public Employee(String n) { //constructor ที่ 2
           name = n;
           System.out.println("I am the second constructor");
        }
         public Employee (String  n, long  i, float s){//constructor ที่ 3
           name = n;
           emp_id= i;
           salary=s;
           System.out.println("I am the third constructor");
         }
}
```

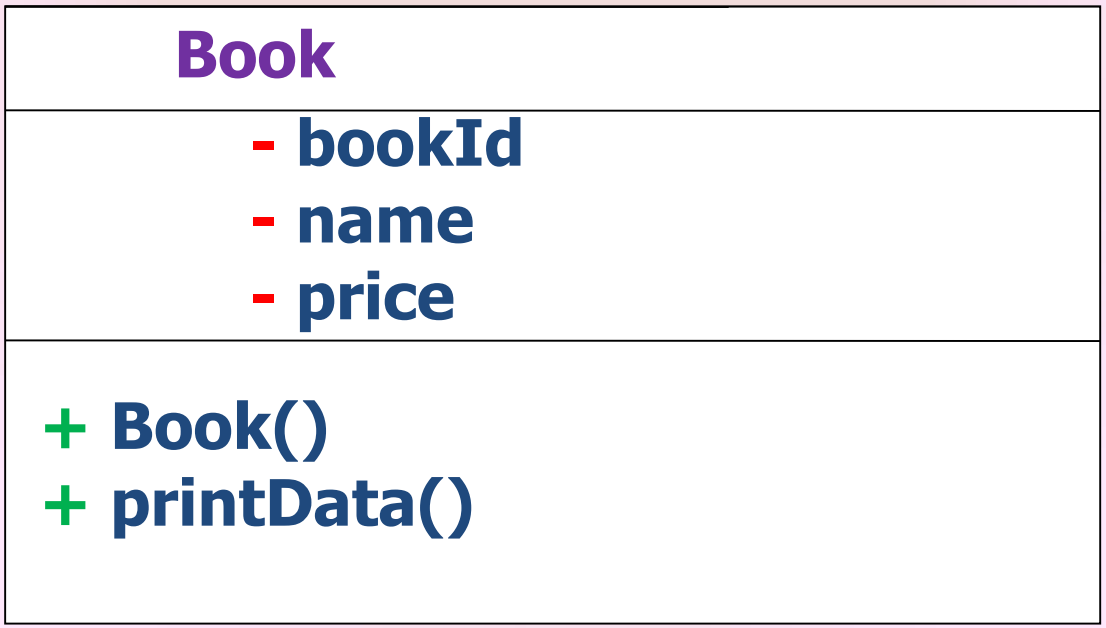
```
public class JavaApp3 {
   public static void main(String[] args) {

      Employee   e1 = new Employee();
      Employee   e2 = new Employee("Matinee Kiewkanya");
      Employee   e3 = new Employee("Benjamas
                                   Panyangam",1002,10000.0f);
}
```

ผลการทำงาน

```
I am the first constructor
I am the second constructor
I am the third constructor
```

ตัวอย่างโปรแกรมที่ 4

โปรแกรมแสดงการใช้งาน Constructor เพื่อสร้างวัตถุที่เป็นสมาชิกใน Array



```
class Book {
    private int bookId;
    private String name;
    private int price;

    public Book(int i, String n, int p){
        bookId = i;
        name=n;
        price=p;
    }

    public void printData() {
       System.out.println("Id: "+bookId+" Name: "+name
       +" Price: "+price);
    }
}
```

โปรแกรม

```
public class JavaApp4 {
   public static void main(String[] args) {

       Book book1 = new Book(1,"Java", 290);
       Book book2 = new Book(2,"SQL", 250);
       Book book3 = new Book(3,"PHP", 260);

       Book[] mybook = new Book[3];
       mybook[0]=book1;
       mybook[1]=book2;
       mybook[2]=book3;

       for(int i=0;i<=2;i++)
          mybook[i].printData();
    }
}
```

ผลการทำงาน

```
Id: 1 Name: Java Price: 290
Id: 2 Name: SQL Price: 250
Id: 3 Name: PHP Price: 260
```

ตัวอย่างโปรแกรมที่ 5

โปรแกรมแสดงการใช้งาน Constructor เพื่อสร้างวัตถุที่เป็นสมาชิกใน Array

code ในส่วนของคลาส Book จะใช้เหมือนกับตัวอย่างที่ 4 แต่จะเปลี่ยนแปลงในส่วนของ main()

```
public class JavaApp5 {
   public static void main(String[] args) {

      Book[] mybook =  {new Book(1,"Java", 290),
                        new Book(2,"SQL", 250),
                        new Book(3,"PHP", 260)};

      for(int i=0;i<=2;i++)
         mybook[i].printData();
   }
}
```

ผลการทำงาน

```
Id: 1 Name: Java Price: 290
Id: 2 Name: SQL Price: 250
Id: 3 Name: PHP Price: 260
```

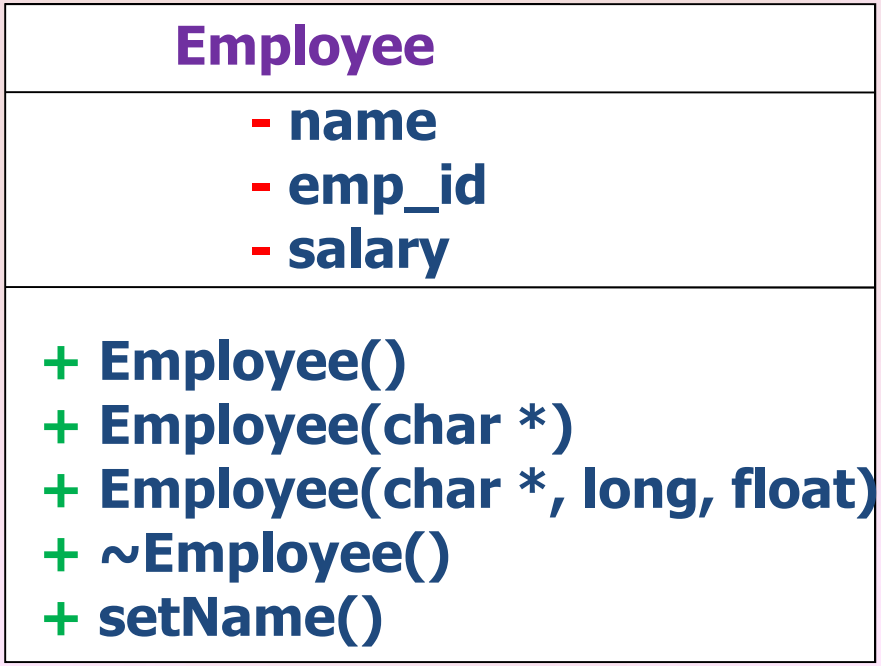
ตัวทำลาย (Destructor)

Destructor คือ เมท็อดที่จะถูกเรียกใช้อัตโนมัติทุกครั้งเพื่อทำลายวัตถุ ซึ่งก็คือการคืนพื้นที่บนหน่วยความจำให้กับระบบเมื่อสิ้นสุดการทำงานภายในขอบเขตที่วัตถุนั้นถูกสร้างขึ้นมา โดยภายในขอบเขตเดียวกันวัตถุที่ถูกสร้างก่อนจะถูกทำลายทีหลัง

- ภาษาจาวาไม่มี เมท็อด Destructor เพราะมี Garbage collection ให้อยู่แล้ว
- ในภาษา C++ เมท็อด Destructor จะมีลักษณะดังนี้
  - ชื่อเดียวกับคลาส แต่มีเครื่องหมาย ~ (Tilde) นำหน้า
  - ไม่มีการส่งค่ากลับ โดยไม่ต้องระบุคำว่า void ไว้ด้านหน้า
  - ไม่มีการรับค่าพารามิเตอร์
  - แต่ละคลาสจะมี Destructor หรือไม่มีก็ได้
  - ในหนึ่งคลาสสามารถมี Destructor ได้เพียงหนึ่งเมท็อดเท่านั้น

ตัวอย่างโปรแกรมภาษา C++

โปรแกรมเพื่อแสดงการทำงานของเมท็อด Destructor ของคลาส Employee



โปรแกรม

```
#include <iostream>
#include <string.h>
using namespace std;
class Employee{
        private:
                char name[50];
                long emp_id;
                float salary;
        public:
                Employee();
                Employee(char* n);
                Employee(char* n, long i, float s);
                ~ Employee() ;
                void setName();
};

Employee:: Employee() {          //constructor
        cout<<"I am the first constructor\n";
}
```

```
Employee:: Employee(char* n) {                    //constructor
        strcpy(name,n);
        cout<< "I am the second constructor\n";
}

Employee:: Employee(char* n, long i,float s) { //constructor
        strcpy(name,n);
        emp_id=i;
        salary=s;
        cout<< "I am the third constructor\n";
}

Employee:: ~Employee() {                          //destructor
        cout<< "I am destructor of " << name <<endl;
}

void Employee:: setName() {
        cout<< "Input your name : " ;   cin>> name;
}
```

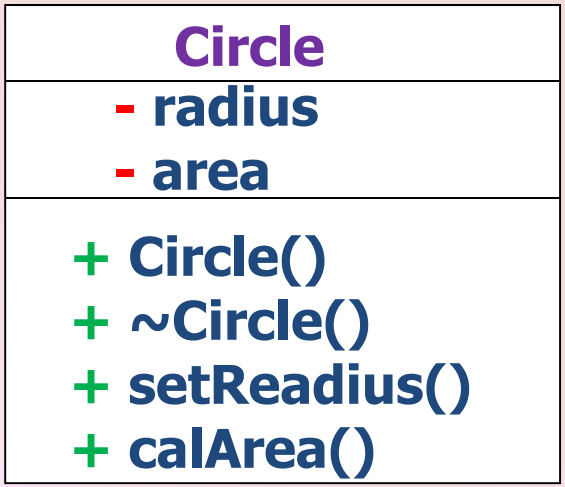
```
int main () {
        Employee   e1;
        e1.setName();
        Employee   e2("Benjamas");
        Employee   e3("Wuttipong",1003,10000.0);
}
```

ผลการทำงาน

```
I am the first constructor
Input your name : Matinee
I am the second constructor
I am the third constructor
I am destructor of Wuttipong
I am destructor of Benjamas
I am destructor of Matinee
```

ตัวอย่างโปรแกรม ภาษา C++

โปรแกรมเพื่อแสดงการทำงานของเมท็อด Constructor และ Destructor ของคลาส Circle กรณีสร้างวัตถุไว้ภายใต้ขอบเขตของคำสั่งย่อย



โปรแกรม

```
#include <iostream>
using namespace std;
class Circle {
        private:
                int radius;
                float area;
        public:
                Circle();
                ~Circle();
                void setRadius();
                void calArea();
};

Circle::Circle() {                  //constructor
        cout << "\nHello from constructor\n";
}

Circle::~Circle() {                 //destructor
        cout << "\nGood bye from destructor\n";
}
```

```
void Circle::setRadius() {
        cout << "Enter radius: "; cin>>radius;
}

void Circle::calArea() {
        area = 3.14 * radius * radius;
        cout << "Area: " << area;
}

int main() {
        for(int i=1;i<=2;i++) {
                cout<<"\n*****Round " <<i<<"*****";
                        Circle myCircle;
                        myCircle.setRadius();
                        myCircle.calArea();
        }
}
```

ผลการทำงาน

```
*****Round 1*****
Hello from constructor
Enter radius: 1
Area: 3.14
Good bye from destructor

*****Round 2*****
Hello from constructor
Enter radius: 2
Area: 12.56
Good bye from destructor
```

การห่อหุ้มและการซ่อนสารสนเทศ (Encapsulation)

Benjamas Panyangam
Matinee Kiewkanya
Computer Science, CMU

การห่อหุ้มและการซ่อนสารสนเทศ

- **การห่อหุ้ม (Encapsulation)** แปลจากคำศัพท์จะหมายถึงการรวมกันโดยมีเปลือกห่อหุ้ม เปรียบเสมือนกับยาเม็ดแคปซูลที่มีเปลือกห่อหุ้มอยู่ ซึ่งจะมองไม่เห็นตัวยาที่อยู่ภายใน แต่ผู้ใช้จะทราบว่าจะใช้ยานี้ได้อย่างไรเท่านั้นเอง

วัตถุในชีวิตประจำวันที่ใช้แนวคิดของการห่อหุ้ม

การห่อหุ้มและการซ่อนสารสนเทศ

- ในการพัฒนาโปรแกรมเชิงวัตถุ การห่อหุ้มจะหมายถึงการรวมกันของโครงสร้างข้อมูลกับฟังก์ชันที่ใช้จัดการข้อมูลเหล่านั้น
- หรือกล่าวอีกอย่างหนึ่งคือการห่อหุ้มนี้จะหมายถึงการรวมกันของแอทริบิวต์และเมท็อดของวัตถุนั่นเองภายใน แต่ผู้ใช้จะทราบว่าจะใช้ได้อย่างไรเท่านั้นเอง
- จะทำให้วัตถุสามารถซ่อนข้อมูลหรือซ่อนสารสนเทศ (Information Hiding) จากภายนอก การเข้าถึงข้อมูลจะสามารถทำได้โดยผ่านส่วนต่อประสาน โดยผู้ใช้ไม่จำเป็นต้องทราบรายละเอียดภายใน

การห่อหุ้มและการซ่อนสารสนเทศ

- ปกป้องสมาชิกของคลาสด้วยการกำหนดประเภทของสมาชิก
- การกำหนดระดับการเข้าถึงด้วย modifier
  - **private:** จะสามารถเข้าถึงได้จากเมท็อดภายในคลาสของตนเอง
  - **public:** จะสามารถเข้าถึงได้ทั้งจากเมท็อดภายในคลาสและเมท็อดภายนอกคลาส
  - **protected:** จะสามารถเข้าถึงได้จากเมท็อดภายในคลาส และเมท็อดของคลาสที่สืบทอด

ตัวดัดแปร (Modifier)

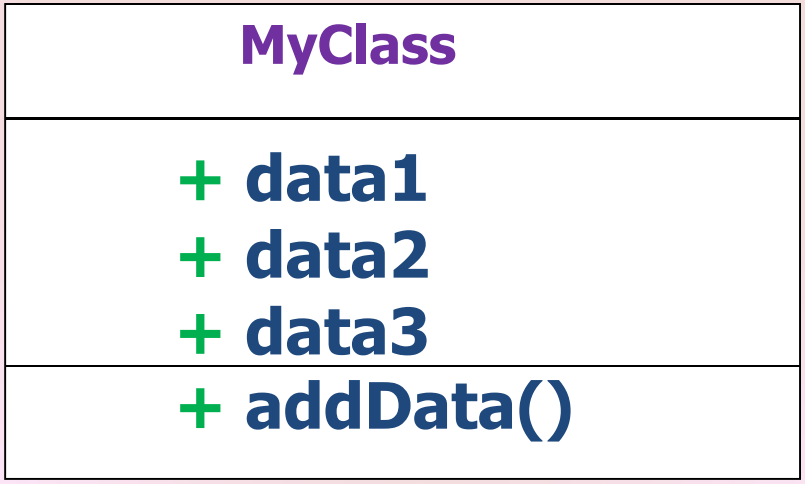
- เป็นคำสำคัญที่ใช้ระบุการเข้าถึงสมาชิกของคลาส

	ใช้ได้ทั้งหมด	package เดียวกัน	ต่าง package กัน	ต่าง package กัน แต่เป็นคลาสแม่ คลาสลูกกัน	คลาสเดียวกัน
public	✓	✓	✓	✓	✓
protected	✗	✓	✗	✓	✓
package	✗	✓	✗	✗	✓
private	✗	✗	✗	✗	✓

private, package, protected, public
(เรียงลำดับจากความเข้มงวดมากถึงน้อย (อิสระที่สุด))

ตัวอย่างโปรแกรมที่ 1

โปรแกรมเพื่อหาผลบวกของเลข 2 จำนวน แสดงให้เห็นการใช้ระดับการเข้าถึงสมาชิกของคลาสแบบ public



โปรแกรม

```
class MyClass{
    public  int  data1 ;
    public  int  data2;
    public  int  data3;

    public  void addData()  {
            data3 = data1 + data2 ;
    }
}
public class JavaApp1 {
   public static void main(String[] args){
         MyClass myObj;
         myObj = new MyClass();

         myObj.data1 = 10;
         myObj.data2 = 20;
         myObj.addData( );
         System.out.println("result = " + myObj.data3);
   }
}
```

ผลการทำงาน

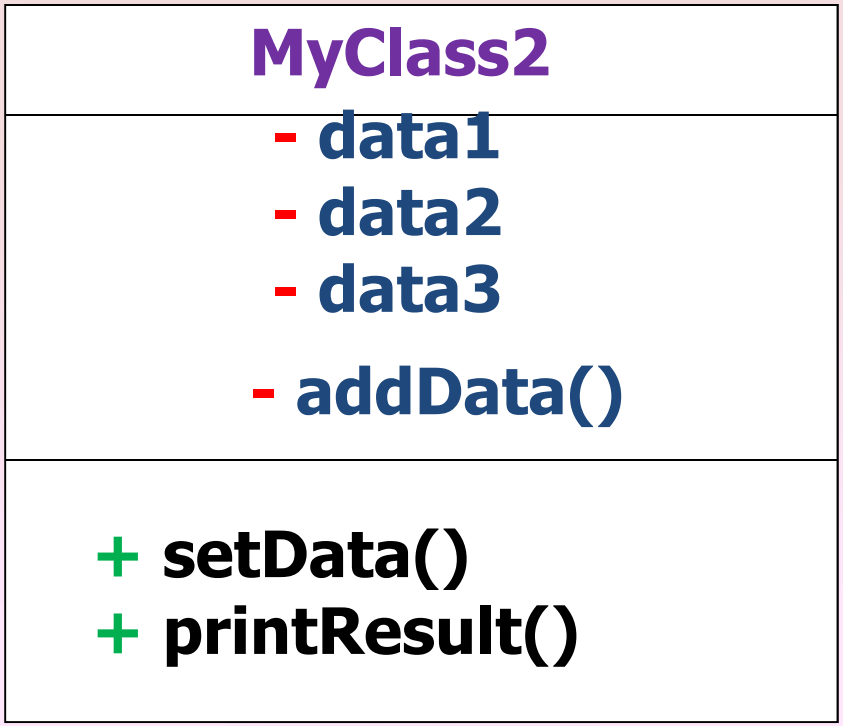
```
result = 30
```

สามารถเข้าถึงได้จากเมท็อดภายในคลาส

สามารถเข้าถึงได้จากเมท็อดภายนอกคลาส

ตัวอย่างโปรแกรมที่ 2

โปรแกรมเพื่อหาผลบวกของเลข 2 จำนวน แสดงให้เห็นการใช้ระดับการเข้าถึงสมาชิกของคลาสแบบ private และ public



โปรแกรม

```
import java.util.Scanner;
class MyClass2{
    private int data1 ;
    private int data2;
    private int data3;
    public void setData(){
        Scanner input = new Scanner(System.in);
        System.out.print("Enter data1 : ");
        data1 = input.nextInt();
        System.out.print("Enter data2 : ");
        data2 = input.nextInt();
    }
    private void addData(){
        data3 = data1 + data2 ;
    }
    public void printResult(){
        addData( );
        System.out.println("result = " + data3);
    }
}
```

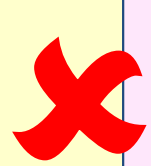
สามารถเข้าถึงได้จากเมท็อดภายในคลาส

สามารถเข้าถึงได้จากเมท็อดภายในคลาส

สามารถเข้าถึงได้จากเมท็อดภายในคลาส

ตัวอย่างการเขียนโปรแกรมที่ไม่ถูกต้อง

```
public class JavaApp2 {
    public static void main(String[] args){
        MyClass2 myObj;
        myObj = new MyClass2();

        myObj.data1 = 10;
        myObj.data2 = 20;
        myObj.addData( );
        System.out.println("result = " + myObj.data3);
    }
}
```



ผิดทุกคำสั่ง

data1, data2, data3 และ add_data( ) เป็น private
ไม่สามารถเข้าถึงได้จากเมท็อดภายนอกคลาส

ตัวอย่างการเขียนโปรแกรมที่ถูกต้อง

```
public class JavaApp2 {
    public static void main(String[] args){
        MyClass2 myObj;
        myObj = new MyClass2();

        myObj.setData ( );
        myObj.printResult ( );
}
```

ผลการทำงาน

```
Enter data1 : 10
Enter data2: 20
result = 30
```

setData( ) และ printResult ( ) เป็น public
จึงสามารถเข้าถึงได้จากภายนอกคลาส

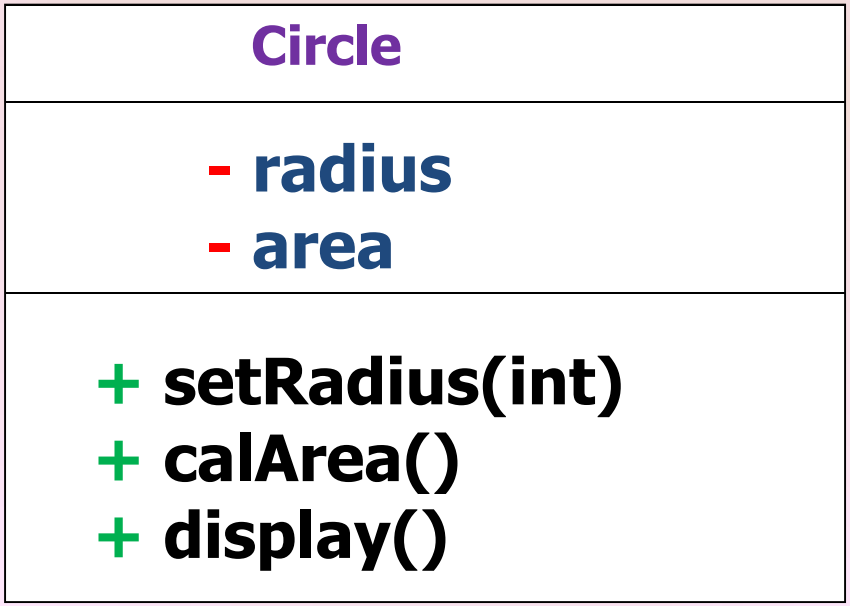
การใช้เมท็อดอ่านค่าและเมท็อดกำหนดค่า

โดยปกติแล้ว การเข้าถึงข้อมูลสมาชิกจะมีวัตถุประสงค์ 2 อย่าง คือ

- เข้าถึงเพื่ออ่านค่าที่มีอยู่แล้วในข้อมูล จะทำผ่านเมท็อดอ่านค่าที่เรียกว่า **Get Method** โดยการตั้งชื่อเมท็อดจะนิยมตั้งชื่อขึ้นต้นด้วยคำว่า **get** แล้วตามด้วยชื่อข้อมูลที่ต้องการจะอ่านค่า การทำงานจะเป็น**การส่งค่าข้อมูลกลับ ผ่านคำสั่ง return**
- เข้าถึงเพื่อกำหนดค่าลงในข้อมูล ซึ่งจะทำผ่านเมท็อดกำหนดค่าที่เรียกว่า **Set Method** โดยการตั้งชื่อเมท็อดจะนิยมตั้งชื่อขึ้นต้นด้วยคำว่า **set** แล้วตามด้วยชื่อ**ข้อมูลที่ต้องการจะกำหนดค่า** การทำงานจะเป็นการรับค่าตัวแปรเข้ามาเป็นพารามิเตอร์ของเมท็อด แล้วจึงนำค่าดังกล่าวมากำหนดให้เป็นค่าของข้อมูลที่ต้องการ

ตัวอย่างโปรแกรมที่ 3

โปรแกรมเพื่อกำหนดค่ารัศมี แล้วคำนวณหาพื้นที่ของวงกลม แสดงให้เห็นการใช้ Set Method



โปรแกรม

```
class Circle {
        private  int radius;
        private  float  area;

         public void setRadius (int r) {
                radius  = r;
        }
         public void calArea(){
                area = 3.14f * radius * radius;
        }
         public void display( ) {
                System.out.println("Area of circle with
                radius " + radius   + " is " + area) ;
        }
}
```

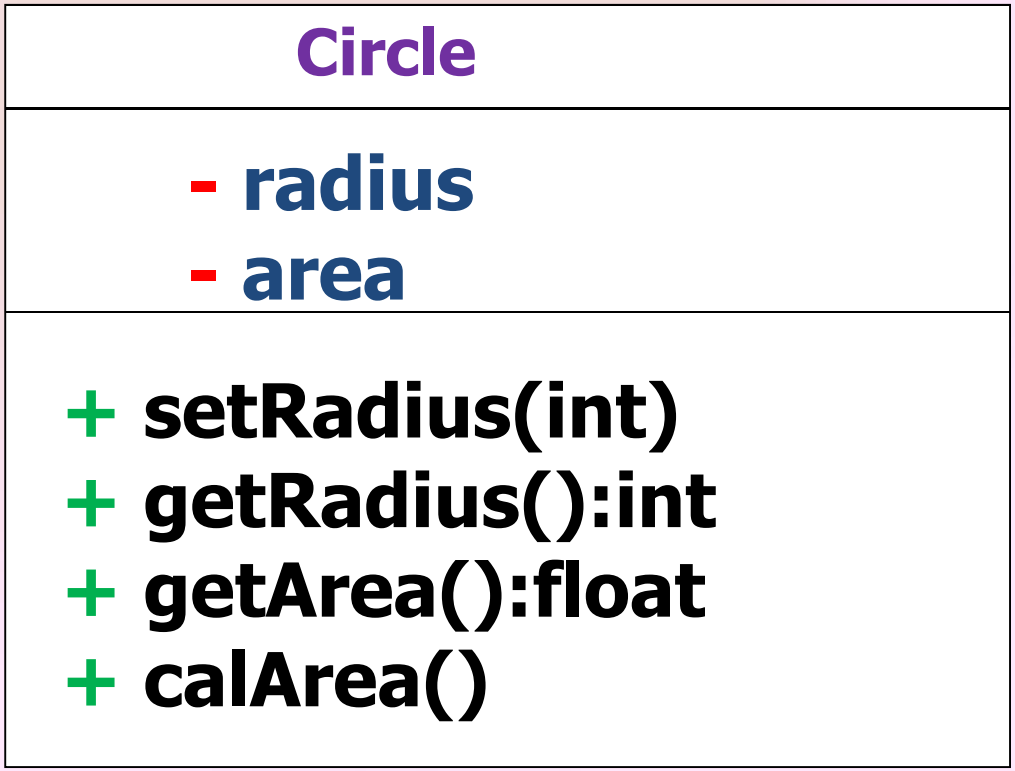
```
public class JavaApp3{
   public static void main(String[] args){

         Circle circleObj = new Circle();

         circleObj.setRadius(5);
         circleObj.calArea();
         circleObj.display( );
}
```

ผลการทำงาน

```
Area of circle with radius 5 is 78.5
```

ตัวอย่างโปรแกรมที่ 4

โปรแกรมเพื่อหาผลรวมของพื้นที่วงกลม 5 วง ที่มีค่ารัศมีตั้งแต่ 1 ถึง 5 แสดงให้เห็นการใช้ Set Method และ Get Method



โปรแกรม

```
class Circle {
        private int radius;
        private float  area;

         public void setRadius(int r){
                radius  = r;
        }
        public int getRadius(){
                return radius;
        }
         public float  getArea(){
                return area;
        }
        public void calArea(){
                area = 3.14f * radius * radius;
        }
}
```

```
public class JavaApp4 {
    public static void main(String[] args){
        Circle[] circleObj = new Circle[5];
        int i;
        float sum =0f;

        for(i=0;i<5;i++)  {
            circleObj[i]= new Circle();
            circleObj[i].setRadius(i+1);
            circleObj[i].calArea();

            System.out.print("radius = " + circleObj[i].getRadius());
            System.out.print(" area = " + circleObj[i].getArea()+ "\n");
            sum += circleObj[i].getArea();
        }
        System.out.println("Total area = " + sum);
    }
}
```

ผลการทำงาน

```
radius = 1 area = 3.14
radius = 2 area = 12.56
radius = 3 area = 28.26
radius = 4 area = 50.24
radius = 5 area = 78.5
Total area = 172.7
```

การใช้งาน keyword static

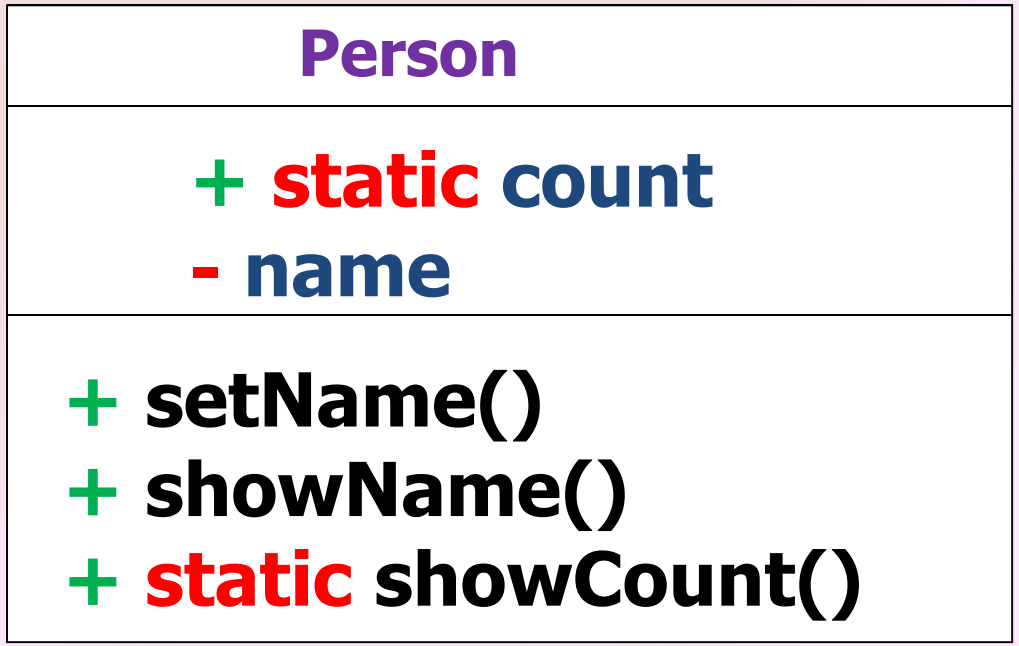
- ในการประกาศ data และ method ของ class หากมี keyword **static** นำหน้า แสดงว่าเป็น class data หรือ class method
- แต่หากการประกาศ data และ method ของ class ไม่มี static นำหน้า แสดงว่าเป็น instance data หรือ instance method
- การเรียกใช้จะแตกต่างกันดังนี้
  - class data หรือ class method จะเรียกใช้ด้วย

    ชื่อคลาส. ชื่อdata หรือ *ชื่อคลาส.ชื่อmethod()*
  - instance data หรือ instance method จะเรียกใช้ด้วย

    ชื่อวัตถุ.ชื่อdata หรือ *ชื่อวัตถุ.ชื่อmethod()*

ตัวอย่างโปรแกรมที่ 5

โปรแกรมเพื่อรับข้อมูลชื่อบุคคล พร้อมทั้งนับจำนวน



โปรแกรม

```
class Person {
    private String name; //instance data
    public static int count=0;   //class data

    public void setName(){ //instance method
        Scanner input = new Scanner(System.in);
        System.out.print("Enter name : ");
        name = input.nextLine();      count++;
        System.out.println("count = "+count);
    }

    public void showName(){//instance method
        System.out.println(" name : "+name);
    }

    public static void showCount(){//class method
        System.out.println("count = "+count);
    }
}
```

```
public class JavaApp5 {
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        char ans;
        int i=0;
        Person p[]=new Person[50];
        do{
              p[i] = new Person();
              p[i].setName();  //call instance method
              System.out.print("Any person?(y/n)");
              ans = input.nextLine().charAt(0);
              i++;
        }while (ans=='y');

        for(i=0;i<Person.count;i++) {
              p[i].showName();  //call instance method
        }
        Person.showCount();  //call class method
}}
```

ผลการทำงาน

```
Enter name : Matinee
count = 1
Any person?(y/n)y
Enter name : Prapaporn
count = 2
Any person?(y/n)y
Enter name : Benjamas
count = 3
Any person?(y/n)n
name : Matinee
name : Prapaporn
name : Benjamas
count = 3
```