Aggregation

- **Aggretion** →

  school
  ↑
  student

  } อันบน need อันล่าง !

- **Composition** →

  Dear ♥
  ↑
  Pluem ♥

  } ทั้งคู่ต้องการกัน แยกกันไม่ได้

- **Implements** - - - - - →

  • ใช้กันคนละ type

  - class - - - → interface

  * Interface - - → class ไม่ได้น้า : o

  Ex.

- **Extends** →

  • ใช้กับ type เดียวกัน

  - class ⟶ class
  - interface ⟶ interface

  Ex.

  พ่อ
  ↑
  ลูก

- **Asscociation** →

  Class
  ↑
  Test

Arrays

count = 4

- ++ Count (บวกค่าก่อนแล้วค่อยใส่)
- Count ++ (Count จะบวกค่าไปทีละ 1)

Remove

- จะลบ pluem[1] = null ;

∴ จะต้อง loop โดยการ for (int i=0 ; i<6 ; i++)

จะได้เป็น pluem[removingIndex] = null ; Count -- ;

```
for (int i = removingIndex ; i < Count ; i++) {
    pluem[i] = pluem[i+1];
    pluem[i+1] = null ; }
```

* removingIndex คือ ค่า (ตำแหน่งที่ต้องการลบ)

Search

count = 9 หา letter = "e";

```
public int search (String letter) {
    for (int i=0 ; i < count ; i++) {
        if (name[i].equals(letter) {
            return i ;
        }
    }
    return -1 ;   * -1 = return นอก for loops
}
```

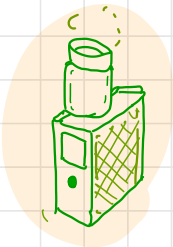
Abstract Class

• ไม่มี Body ก็ได้. / ใช้ abstract ในการเขียน interface !

```
public abstract Animal {
    public void sound ();
}
public Pig implements Animal {
    public void sound () {
        sout.("อื๊ด") ;
    }
}
```

• Inheritance

• การสืบทอด



• Enum

เอาไว้เก็บข้อมูลโดยมีกฎค่า

```
public enum ชื่อ Class {

}
```

Ex:
```
public enum month {
    JAN, FEB, MARCH, ... ;
}
```

เวลาเรียกใช้ `if ( เดือน == month.FEB )`

เดือน = Attribute ของ class นั้น

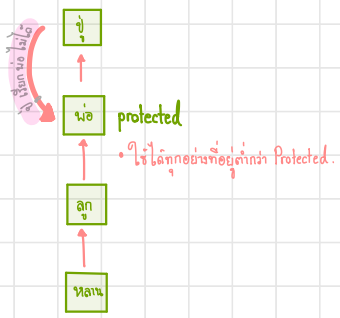
เอาไว้ check.

• Visibility Modifiers

• Public ทุกที่ (+)

• Private ได้ใน class ตัวเอง! (-)

• Protected (*)



• ใช้ได้ทุกอย่างที่อยู่ต่ำกว่า Protected.

Generic

- public void name () <T> {

feel เดียวกับ parameter เลย

ประกาศ generic.

- public class Test <T> {

```
T obj ;

public Test ( T obj ) {
    this. obj = obj ;
}

public T get obj () {
    return obj ;
}
```

- public class main {

```
public static void main (String ags[] ) {

    Test < Integer > i Obj = new < Integer > ( 15)
      sout. (iobj. getObj() );  * ได้ 15

    Test < string > sObj = new < String > (" D")
      sout. ( sObj. getObj() );  * ได้ D
  }
}
```

Comparable → java.lang.comparable

ใช้ implement เพราะ Comparable เป็น Interface

- Public class Movie implement Comparable <Movie> {

```
private double rating ;
private String name ;
private int year ;

Public int compareTo ( Movie m ) {
    return this.year - m.year ;
}
```

ถ้ามาก return +
น้อย return -
เท่ากัน return 0

```
Public class main {
   psvm {
      Movie [] m1 = new Movie [10] ;

      ArraysSort(m1); ใช้เมื่อจัดเรียงค่าตามลำดับ.

      for ( Movie movie : m1) {
          sout ( movie.toString() );
        }
      }
   }
```

ตารางค่า

"teletubbies"	85	1998	
"Dora"	7.5	2008	สลับกัน
"Terminator"	7.5	1998	
"Oliv"	10.0	2020	

Comparator → java.util.Comparator

- Public class Movie implement Comparable <Movie> {

```
    private double rating ;
    private String name ;
    private int year ;

    public int compareTo (Movie m) {... }

    public Movie (String name, double rating, int year) {... }
}
Class RatingCompare implement Comparator <Movie> {

    Public int compare (Movie m1, Movie m2) { ... }
}
```

เปรียบเทียบ parameter 2 ตัว

Ex

```
Class nameCompare implements comparator <Movie> {

    Public int compare (Movie m1, Movie m2) {
        return m1.name.compareTo (m2.name) ;
    }
}
```

- public static Comparator < Ex. > ratingCompare = new Comparator < Ex > () {

  new Comparetor ใหม่

  @Override
  public int Compare (. . . .) { เขียนไง }
  };

public static Comparator yearCompare () {
return new Comparator < Ex > () {
} ทำเป็นในรูป Method.

@Override
public int compare (. . . .) { เขียนไง }
};

• Iterator

ชื่อ array.

จำนวนช่องที่มี Count

Contact list

int Count = 4

data type

```
Public Iterator < Contact > iterator () {
    return new Iterator < Contact > () {
        private int index ;
        @Override
        public boolean hasNext () {
            return index < count ;
        }
        @Override        data type เหมือน parameter
        public contact next () {
            if ( index == count ) { Do Something }
            return list [ index ++ ] ;
    } {
```