Interface Callback

saacsos

ตัวอย่าง Interface java.lang.Comparable

```
interface Comparable {
   public int compareTo(Object o);
}
```

- Class ที่จะใช้ Comparable ต้องมี attribute ที่เปรียบเทียบได้
- ถ้าค่า attribute มากกว่าค่า attribute ของ o ให้ return 1
- ถ้าค่า attribute น้อยกว่าค่า attribute ของ o ให้ return -1
- ถ้าค่า attribute เท่ากับ ค่า attribute ของ o ให้ return 0

ตัวอย่าง Student

```
class Student implements Comparable {
    private double score;
    @override
    public int compareTo(Object o) {
        Student other = (Student) o;
        if (score < other.score) return -1;
        if (score > other.score) return 1;
        return 0;
    }
}
```

ตัวอย่าง GameCharacter

```
class GameCharacter implements Comparable {
    private int hp;
    @override
    public int compareTo(Object o) {
        GameCharacter other = (GameCharacter) o;
        if (hp < other.hp) return -1;
        if (hp > other.hp) return 1;
        return 0;
    }
}
```

Arrays.sort()

Interface Comparable นำไปใช้ประโยชน์ใน Arrays.sort ได้

```
Student[] students = new Student[3];
students[0] = new Student("Mr.A", 20.0);
students[1] = new Student("Mr.B", 90.5);
students[2] = new Student("Mr.C", 77.6);

Arrays.sort(students);
```

Arrays.sort()

Interface Comparable นำไปใช้ประโยชน์ใน Arrays.sort ได้

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10);
players[1] = new GameCharacter("Player 2", 20);
players[2] = new GameCharacter("Player 3", 5);

System.out.println(Arrays.toString(players));
Arrays.sort(players);
System.out.println(Arrays.toString(players));
```

หาค่าที่มากที่สุด โดยใช้ Comparable

```
class Data {
  public static Comparable max(Comparable a, Comparable b) {
     if (a.compareTo(b) > 0) return a;
     return b;
  }

  public static Comparable max(Comparable[] comparables) {
     if (comparables.length == 0) return null;
     Comparable m = comparables[0];
     for (int i = 1; i < comparables.length; i++ )
         m = max(m, comparables[i]);
     return m;
  }
}
```

หาค่าที่มากที่สุด

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10);
players[1] = new GameCharacter("Player 2", 20);
players[2] = new GameCharacter("Player 3", 5);

System.out.println(Data.max(players));
```

ข้อจำกัดของ Interface

- เราสามารถเพิ่ม Interface ให้กับ Class ที่เราสร้างขึ้นเองเท่านั้น
- ไม่สามารถเพิ่ม Interface ให้ Library Class หรือ package ที่คนอื่นสร้างขึ้น
- Interface ที่ถูก implements ไปใน Class จะทำงานได้แบบเดียว

  อยากเปรียบเทียบ hp ของ GameCharacter

  และอยากเปรียบเทียบ damage ของ GameCharacter ด้วย

  จะเขียน Interface อย่างไร

Interface Callback

- ลดข้อจำกัดของ Interface
- Interface Callback มี method ที่รับ Object มาใช้งาน
- Method นั้นคืนค่าบางอย่าง เพื่อให้ตัวที่เรียกใช้ไปทำงานต่อได้

ตัวอย่าง Interface Callback java.util.Comparator

```
interface Comparator {
   public int compare(Object o1, Object o2);
}
```

- ถ้าค่า attribute `o1` มากกว่าค่า attribute `o2` ให้ return 1
- ถ้าค่า attribute `o1` น้อยกว่าค่า attribute `o2` ให้ return -1
- ถ้าค่า attribute `o1` เท่ากับ ค่า attribute `o2` ให้ return 0
- จะไม่มี Model Class ใดที่ `implements Comparator`

hpComparator

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10, 30);
players[1] = new GameCharacter("Player 2", 20, 10);
players[2] = new GameCharacter("Player 3", 5, 25);
Comparator hpComparator = new Comparator() {
    @Override
    public int compare(Object o1, Object o2) {
        GameCharacter p1 = (GameCharacter) o1;
        GameCharacter p2 = (GameCharacter) o2;
        if (p1.getHp() > p2.getHp()) return 1;
        if (p1.getHp() < p2.getHp()) return -1;
        return 0;
    }
});
Arrays.sort(players, hpComparator);
System.out.println(Arrays.toString(players));
```

GameCharacter ไม่ต้อง implements Comparable หรือ Comparator

damageComparator

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10, 30);
players[1] = new GameCharacter("Player 2", 20, 10);
players[2] = new GameCharacter("Player 3", 5, 25);
Comparator damageComparator = new Comparator() {
    @Override
    public int compare(Object o1, Object o2) {
        GameCharacter p1 = (GameCharacter) o1;
        GameCharacter p2 = (GameCharacter) o2;
        if (p1.getDamage() > p2.getDamage()) return 1;
        if (p1.getDamage() < p2.getDamage()) return -1;
        return 0;
    }
});
Arrays.sort(players, damageComparator);
System.out.println(Arrays.toString(players));
```

Anonymous Class

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10, 30);
players[1] = new GameCharacter("Player 2", 20, 10);
players[2] = new GameCharacter("Player 3", 5, 25);
Arrays.sort(players, new Comparator() {
    @Override
    public int compare(Object o1, Object o2) {
        GameCharacter p1 = (GameCharacter) o1;
        GameCharacter p2 = (GameCharacter) o2;
        if (p1.getHp() > p2.getHp()) return 1;
        if (p1.getHp() < p2.getHp()) return -1;
        return 0;
    }
});
System.out.println(Arrays.toString(players));
```

หาค่าที่มากที่สุด โดยใช้ Comparator

```
class Data {
  public static Object max(Object a, Object b, Comparator c) {
     if (c.compare(a, b) > 0) return a;
     return b;
  }

  public static Object max(Object[] objects, Comparator c) {
     if (objects.length == 0) return null;
     Object m = objects[0];
     for (int i = 1; i < objects.length; i++ )
         m = max(m, objects[i], c);
     return m;
  }
}
```

หาค่า HP ที่มากที่สุด โดยใช้ Comparator

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10, 30);
players[1] = new GameCharacter("Player 2", 20, 10);
players[2] = new GameCharacter("Player 3", 5, 25);

System.out.println(Data.max(players,  new Comparator() {
    @Override
    public int compare(Object o1, Object o2) {
        GameCharacter p1 = (GameCharacter) o1;
        GameCharacter p2 = (GameCharacter) o2;
        if (p1.getHp() > p2.getHp()) return 1;
        if (p1.getHp() < p2.getHp()) return -1;
        return 0;
    }
}));
```

หาค่า Damage ที่มากที่สุด โดยใช้ Comparator

```
GameCharacter[] players = new GameCharacter[3];
players[0] = new GameCharacter("Player 1", 10, 30);
players[1] = new GameCharacter("Player 2", 20, 10);
players[2] = new GameCharacter("Player 3", 5, 25);

System.out.println(Data.max(players,  new Comparator() {
    @Override
    public int compare(Object o1, Object o2) {
        GameCharacter p1 = (GameCharacter) o1;
        GameCharacter p2 = (GameCharacter) o2;
        if (p1.getDamage() > p2.getDamage()) return 1;
        if (p1.getDamage() < p2.getDamage()) return -1;
        return 0;
    }
}));
```