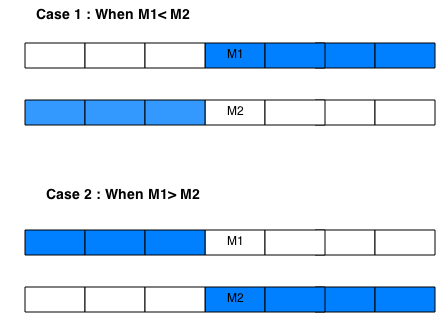
## find median of two sorted arrays: Algorithm

1. Find the middle of two sorted array, if they are combined together.  
2. If length of one array is odd and other even, return middle element.   
3. Else, find middle +1 element in same arrays.  
4. Get the average of (middle and middle + 1) and return as median.



### Median of two sorted array algorithm

1. If length of both arrays is 1, take average and return the value.  
2. If length of both arrays is 2, then return following value (max(array1[0], array2[0]) + min(array1[1], array2[1]) )/2  
3. If length is greater than 2, follow steps below  
 3.a Find median of array A and array B individually. Let them be M1 and M2   
 3.b If M1 == M2 return M1   
 3.c If M1 < M2, follow step 1 onwards with array A[mid...N] and array B[0, mid-1]   
 3.d If M2 < M1, follow step 1 onwards with array B[mid...N] and array B[0, mid-1]

## 1. Sort an Array

To sort an Array, use the **Arrays.sort()**.

String[] fruits = new String[] {"Pineapple","Apple", "Orange", "Banana"};  
  
Arrays.sort(fruits);  
  
int i=0;  
for(String temp: fruits){  
 System.out.println("fruits " + ++i + " : " + temp);  
}

Output

fruits 1 : Apple  
fruits 2 : Banana  
fruits 3 : Orange  
fruits 4 : Pineapple

## 2. Sort an ArrayList

To sort an ArrayList, use the **Collections.sort()**.

List<String> fruits = new ArrayList<String>();  
  
fruits.add("Pineapple");  
fruits.add("Apple");  
fruits.add("Orange");  
fruits.add("Banana");  
  
Collections.sort(fruits);  
  
int i=0;  
for(String temp: fruits){  
 System.out.println("fruits " + ++i + " : " + temp);  
}

Output

fruits 1 : Apple  
fruits 2 : Banana  
fruits 3 : Orange  
fruits 4 : Pineapple

## Sort an Object with Comparable

How about a Java Object? Let create a Fruit class:

public class Fruit{  
  
 private String fruitName;  
 private String fruitDesc;  
 private int quantity;  
  
 public Fruit(String fruitName, String fruitDesc, int quantity) {  
 super();  
 this.fruitName = fruitName;  
 this.fruitDesc = fruitDesc;  
 this.quantity = quantity;  
 }  
  
 public String getFruitName() {  
 return fruitName;  
 }  
 public void setFruitName(String fruitName) {  
 this.fruitName = fruitName;  
 }  
 public String getFruitDesc() {  
 return fruitDesc;  
 }  
 public void setFruitDesc(String fruitDesc) {  
 this.fruitDesc = fruitDesc;  
 }  
 public int getQuantity() {  
 return quantity;  
 }  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
}

To sort it, you may think of **Arrays.sort()** again, see below example :

package com.mkyong.common.action;  
  
import java.util.Arrays;  
  
public class SortFruitObject{  
  
 public static void main(String args[]){  
  
 Fruit[] fruits = new Fruit[4];  
  
 Fruit pineappale = new Fruit("Pineapple", "Pineapple description",70);  
 Fruit apple = new Fruit("Apple", "Apple description",100);  
 Fruit orange = new Fruit("Orange", "Orange description",80);  
 Fruit banana = new Fruit("Banana", "Banana description",90);  
  
 fruits[0]=pineappale;  
 fruits[1]=apple;  
 fruits[2]=orange;  
 fruits[3]=banana;  
  
 Arrays.sort(fruits);  
  
 int i=0;  
 for(Fruit temp: fruits){  
 System.out.println("fruits " + ++i + " : " + temp.getFruitName() +  
 ", Quantity : " + temp.getQuantity());  
 }  
  
 }  
}

Nice try, but, what you expect the **Arrays.sort()** will do? You didn’t even mention what to sort in the Fruit class. So, it will hits the following error :

Exception in thread "main" java.lang.ClassCastException:  
com.mkyong.common.Fruit cannot be cast to java.lang.Comparable  
 at java.util.Arrays.mergeSort(Unknown Source)  
 at java.util.Arrays.sort(Unknown Source)

To sort an Object by its property, you have to make the Object implement the **Comparable** interface and override the **compareTo()** method. Lets see the new Fruit class again.

public class Fruit implements Comparable<Fruit>{  
  
 private String fruitName;  
 private String fruitDesc;  
 private int quantity;  
  
 public Fruit(String fruitName, String fruitDesc, int quantity) {  
 super();  
 this.fruitName = fruitName;  
 this.fruitDesc = fruitDesc;  
 this.quantity = quantity;  
 }  
  
 public String getFruitName() {  
 return fruitName;  
 }  
 public void setFruitName(String fruitName) {  
 this.fruitName = fruitName;  
 }  
 public String getFruitDesc() {  
 return fruitDesc;  
 }  
 public void setFruitDesc(String fruitDesc) {  
 this.fruitDesc = fruitDesc;  
 }  
 public int getQuantity() {  
 return quantity;  
 }  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
 public int compareTo(Fruit compareFruit) {  
  
 int compareQuantity = ((Fruit) compareFruit).getQuantity();  
  
 //ascending order  
 return this.quantity - compareQuantity;  
  
 //descending order  
 //return compareQuantity - this.quantity;  
  
 }  
}

The new Fruit class implemented the **Comparable** interface, and overrided the **compareTo()** method to compare its quantity property in ascending order.

The **compareTo()** method is hard to explain, in integer sorting, just remember

1. this.quantity – compareQuantity is ascending order.
2. compareQuantity – this.quantity is descending order.

To understand more about compareTo() method, read this [Comparable documentation](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Comparable.html).

Run it again, now the Fruits array is sort by its quantity in ascending order.

fruits 1 : Pineapple, Quantity : 70  
fruits 2 : Orange, Quantity : 80  
fruits 3 : Banana, Quantity : 90  
fruits 4 : Apple, Quantity : 100

## Sort an Object with Comparator

How about sorting with Fruit’s “fruitName” or “Quantity”? The Comparable interface is only allow to sort a single property. To sort with multiple properties, you need **Comparator**. See the new updated Fruit class again :

import java.util.Comparator;  
  
public class Fruit implements Comparable<Fruit>{  
  
 private String fruitName;  
 private String fruitDesc;  
 private int quantity;  
  
 public Fruit(String fruitName, String fruitDesc, int quantity) {  
 super();  
 this.fruitName = fruitName;  
 this.fruitDesc = fruitDesc;  
 this.quantity = quantity;  
 }  
  
 public String getFruitName() {  
 return fruitName;  
 }  
 public void setFruitName(String fruitName) {  
 this.fruitName = fruitName;  
 }  
 public String getFruitDesc() {  
 return fruitDesc;  
 }  
 public void setFruitDesc(String fruitDesc) {  
 this.fruitDesc = fruitDesc;  
 }  
 public int getQuantity() {  
 return quantity;  
 }  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
 public int compareTo(Fruit compareFruit) {  
  
 int compareQuantity = ((Fruit) compareFruit).getQuantity();  
  
 //ascending order  
 return this.quantity - compareQuantity;  
  
 //descending order  
 //return compareQuantity - this.quantity;  
  
 }  
  
 public static Comparator<Fruit> FruitNameComparator  
 = new Comparator<Fruit>() {  
  
 public int compare(Fruit fruit1, Fruit fruit2) {  
  
 String fruitName1 = fruit1.getFruitName().toUpperCase();  
 String fruitName2 = fruit2.getFruitName().toUpperCase();  
  
 //ascending order  
 return fruitName1.compareTo(fruitName2);  
  
 //descending order  
 //return fruitName2.compareTo(fruitName1);  
 }  
  
 };  
}

The Fruit class contains a static **FruitNameComparator** method to compare the “fruitName”. Now the Fruit object is able to sort with either “quantity” or “fruitName” property. Run it again.

1. Sort Fruit array based on its “fruitName” property in ascending order.

Arrays.sort(fruits, Fruit.FruitNameComparator);

Output

fruits 1 : Apple, Quantity : 100  
fruits 2 : Banana, Quantity : 90  
fruits 3 : Orange, Quantity : 80  
fruits 4 : Pineapple, Quantity : 70

2. Sort Fruit array based on its “quantity” property in ascending order.

Arrays.sort(fruits)

Output

fruits 1 : Pineapple, Quantity : 70  
fruits 2 : Orange, Quantity : 80  
fruits 3 : Banana, Quantity : 90  
fruits 4 : Apple, Quantity : 100

The **java.lang.Comparable** and **java.util.Comparator** are powerful but take time to understand and make use of it, may be it’s due to the lacking of detail example.

## My thoughts…

In future, Arrays class should provides more generic and handy method – **Arrays.sort(Object, String, flag)**.

To sort a object array by its “fruitName” in ascending order.

Arrays.sort(fruits, fruitName, Arrays.ASCENDING);

To sort a object array by its “quantity” in ascending order.

Arrays.sort(fruits, quantity, Arrays.DESCENDING);