Sorting:

public interface Comparable<T> {

public int compareTo(T o);

}

import java.util.\*;

public class Name implements Comparable<Name> {

private final String firstName, lastName;

public Name(String firstName, String lastName) {

if (firstName == null || lastName == null)

throw new NullPointerException();

this.firstName = firstName;

this.lastName = lastName;

}

public String firstName() { return firstName; }

public String lastName() { return lastName; }

public boolean equals(Object o) {

if (!(o instanceof Name))

return false;

Name n = (Name) o;

return n.firstName.equals(firstName) && n.lastName.equals(lastName);

}

public int hashCode() {

return 31\*firstName.hashCode() + lastName.hashCode();

}

public String toString() {

return firstName + " " + lastName;

}

public int compareTo(Name n) {

int lastCmp = lastName.compareTo(n.lastName);

return (lastCmp != 0 ? lastCmp : firstName.compareTo(n.firstName));

}

}

import java.util.\*;

public class NameSort {

public static void main(String[] args) {

Name nameArray[] = {

new Name("John", "Smith"),

new Name("Karl", "Ng"),

new Name("Jeff", "Smith"),

new Name("Tom", "Rich")

};

List<Name> names = Arrays.asList(nameArray);

Collections.sort(names);

System.out.println(names);

}

}

What if you want to sort some objects in an order other than their natural ordering:

public interface Comparator<T> {

int compare(T o1, T o2);

}

public class Employee implements Comparable<Employee> {

public Name name() { ... }

public int number() { ... }

public Date hireDate() { ... }

...

}

import java.util.\*;

public class EmpSort {

static final Comparator<Employee> SENIORITY\_ORDER =

new Comparator<Employee>() {

public int compare(Employee e1, Employee e2) {

return e2.hireDate().compareTo(e1.hireDate());

}

};

// Employee database

static final Collection<Employee> employees = ... ;

public static void main(String[] args) {

List<Employee> e = new ArrayList<Employee>(employees);

Collections.sort(e, SENIORITY\_ORDER);

System.out.println(e);

}

}

If there was a clash on hiredate:

static final Comparator<Employee> SENIORITY\_ORDER =

new Comparator<Employee>() {

public int compare(Employee e1, Employee e2) {

int dateCmp = e2.hireDate().compareTo(e1.hireDate());

if (dateCmp != 0)

return dateCmp;

return (e1.number() < e2.number() ? -1 :

(e1.number() == e2.number() ? 0 : 1));

}

};