**public** **class** PeopleSortByBirthday **implements** Comparator<Person>

{

**public** **int** compare(Person a, Person b)

{

**if** (a.birthday.getMonth() < b.getMonth())

{

**return** -1;

}

**else** **if**(a.birthday.getMonth() > b.birthday.getMonth)

{

**return** 1;

}

**else**

{

**if**(a.birthday.getDay() < b.birthday.getDay)

{

**return** -1;

}

**else** **if** (a.birthday.getDay() > b.birthday.getDay())

{

**return** 1;

}

**else** **return** 0;

}

}

**public** **class** HW2Comparator<T **extends** Comparable<T>> **implements** Comparator<List<T>>

{

**public** **int** compare(List<T> v1, List<T> v2)

{

**int** i = 0;

**while**(i<v1.size() || i<v2.size())

{

**if**(v1.get(i) == v2.get(i))

{

i++;

}

**else**

{

**return** v1.get(i).compareTo(v2.get(i));

}

}

}

}

Part 2

public interface LivingBeing<B extends LivingBeing<B>>

{

public LivingBeing Parent(B b, Boolean male); //a method that returns the father of b, "male" specifies 1 for male, 2 for female

public LivingBeing[] Siblings(B b, Boolean male); //method that returns all siblings that are male if "male" is true, female if Boolean is false

//check both the cousin and the current LivingBeings parent's siblings and if any are equal then

//they are cousings BOOM

public LivingBeing[] Offspring(B b); //this is just for the second set

//this works because the previous two methods can be used for

//set one in order to check a given person as a cousin, but in

//order to FIND that cousin only knowing one "node" of a tree

//you would need a method to get offspring of the Parent's Sibling's Offspring (cousins).

}