CSCI 2251 – Programming Assignment
Human Resources – Part 2 of 2

This assignment has the following objectives:

1. apply inheritance to objects
2. implement an interface
3. apply polymorphism to the class hierarchy
4. read data from file and create new output files

Problem Description

Nintendo's human resources data is disorganized, full of duplicates, and in metric! The information is stored in a database file, `hr.txt` and it's your task to create two new versions of it:

- One version will be in alphabetical order
- One version will be converted from metric to imperial units
- And both versions will have no duplicates

List of classes that you will write:

- `Main – contains the main method.`
- `Person – stores HR information`
- `PersonList – an interface`
- `PersonSet – a class implementing the interface`
- `PersonImperialSet – a class inheriting from PersonSet`
- `PersonOrderedSet – another class inheriting from PersonSet`

Instructions for Part 2

You need to write two new classes for part 2: `PersonImperialSet` and `PersonOrderedSet`.

1. Add a toString method to `PersonSet` that loops through the ArrayList, concatenating the Persons data to a String variable, which is then returned. The format needs to match the format of `hr.txt`.
2. Write a class named, `PersonOrderedSet`. This class should **extend** `PersonSet` and override the `add` method to add Persons in alphabetical order by name.
3. Write a class named, `PersonImperialSet`. This class should **extend** `PersonSet` and override the `add` method to convert the height measurement from centimeters to inches and the weight from kilograms to pounds. (Look up the conversions online.)
4. Modify `Main` to
   A. instantiate a `PersonOrderedSet` and a `PersonImperialSet`, instead of the `PersonSet`.
   B. Read in the data from the file, use it to populate both set objects with Persons, and then write out the data into two separate output files (one ordered and one imperial).
   C. I recommend adding methods to the classes to get the data in a text format for writing to file. You should think about which class is most appropriate for this method (or methods) to be implemented in order to reduce code duplication.
   D. Output the formatted data with header to two separate files named: hr_imperial_set_output.txt and hr_ordered_set_output.txt

E. Lastly, output the ordered data and the imperial data to the screen/console, nicely formatted in rows and labeled columns (this nice formatting should already be in use if you wrote your toString methods well).

Various Important Notes:

Neither `PersonOrderedSet` nor `PersonImperialSet` should contain duplicate data, but the code that takes care of such things should be implemented in the add method of `PersonSet`.

You must use `super.add(p)` as the final step in `PersonImperialSet.add` after the conversion has been completed. (Assuming that `p` is a variable referring to a `Person` object to be added to the set.)

You are strongly encouraged to use `super.add(p)` in `PersonOrderedSet.add` and then perform a sort of the data to make sure everything is in order. You are allowed to use the built-in `Collections.sort` method, but in order to make it work, `Person` will need to implement the `Comparable` interface and implement a `compareTo` method (You're encouraged to look this up.). You may also look up and modify a bubblesort or other sorting technique for solving this problem.

Objects are passed by reference, not by value. If you add the same `Person` to `PersonOrderedSet` and `PersonImperialSet` you will see that the units get changed from metric to imperial in both. This is NOT what you want. One solution to this problem is to instantiate two new `Person` objects and pass the same name, height, weight info to each, and then pass one to the imperial set and the other to the ordered set. Another solution is to overload the `Person` constructor so that you can pass a `Person` type variable to the constructor and the constructor will take care of copying the data. Both solutions require two separate instantiations of `Person` in main.

UML Diagram for HumanResources Part 2

<<Interface>> PersonList
+ add(p : Person) : void + get(index : int) : Person

PersonSet <<implements>> PersonList
people : ArrayList<Person>
+ add (p : Person) : void + get (index : int) : Person + toString() : String

Person <<implements>> Comparable
- name : string - height : double - weight : double

<<constructor>> Person + getHeight() : double + getWeight() : double + setHeight(height : double) : void + setWeight(weight : double) : void + toString() : String + equals(o : Object) : Boolean + compareTo(p : Person) : int

PersonOrderedSet <<extends>> PersonSet
+ add(p : Person) : void - sort() : void

PersonImperialSet <<extends>> PersonSet
+ add(p : Person) : void - convertHeight(height : double) : double - convertWeight(weight : double) : double

Compilation and Execution

I will test your program as follows:

```
  javac *.java
  java Main hr.txt
```