CSCI 2251 – Programming Assignment
Human Resources – Part 1 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

<u>List of classes that you will write:</u>

- `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 1

For part 1 you need to create three classes: `Person`, `PersonList`, and `PersonSet`. A mostly-blank Main.java has been provided, but you'll need to fill it in **<u>AND</u>** you must answer the five questions at the top of the document.

1. Write a class named `Person`. This will be a very basic class with three attributes for storing name, height, and weight information. This class should also have a `toString` method that returns the `Person` data in a database-ready String format.
2. Write an **interface** named `PersonList`. The interface should have two abstract methods:
   A. `add` – This method takes a `Person` as input and returns void.
   B. `get` – This method takes an `int` as input and returns a `Person` at the corresponding index of the input `int`.
3. Write a class named, `PersonSet`, that **implements** the interface `PersonList`. Use an `ArrayList` and fill in the `add` and `get` methods. You may not use any built in Set-type Java classes.
4. In addition to implementing `add` and `get` methods, `PersonSet` must make sure that no duplicate `Person`s are added. If you want to use the `ArrayList`'s built-in `contains` method to make this easier, you will need to add an `equals` method to `Person`. See below for more details.
5. In the `main` method in `Main`:

A. Instantiate a single `Person` object as a test. You can make up the data passed to the constructor.

B. Instantiate a `PersonSet` object as a test.

C. Read data in from the file `hr.txt` and display it on the command prompt.

If you want to use the `ArrayList`'s `contains` method to see if a `Person` is already in the set, then you need to make sure that `Person` overrides the default `equals` method. To do so, fill in the following comment outline and also refer to this resource for more information:

https://www.geeksforgeeks.org/overriding-equals-method-in-java/

```
//Equals method outline
@Override
public boolean equals(Object o)
{
     //if Object o is null then return false

     //if Object o == this then return true

     //if Object o is not an instance of Person then return false

     //Declare a new variable of type Person (perhaps named p)
     //   and assign it to Object o cast as type Person

     //if Person p has the same name, height, and weight as
     //   this then return true
     //else return false
}
```

UML Diagram for HumanResources Part 1

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

The pound sign or hashtag in the next diagram indicates 'protected', which is important so that the ordered set can easily sort the ArrayList in part 2.

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

Person
- name : string - height : double - weight : double
<<constructor>> Person + getHeight() : double + getWeight() : double + setHeight(height : double) : void + setWeight(weight : double) : void + toString() : String

Compilation and Execution

I will test your program as follows:

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