CECS 277 LAB INTERFACES & POLYMORPHISM

OBJECTIVE: Give you experience building and using interface types.

INTRODUCTION: Please remember the coding standards here.

Generally speaking, complex objects are hard to compare to each other simply because there are so many possible ways to compare them. You might compare two automobiles by their EPA estimated gas mileage, their average sticker price, the horsepower of their engines just to name a few. The `Comparable` interface just demands that the implementing class has **some** way to compare two instances of that same class. The beauty of implementing the `Comparable` interface is that allows other class methods that count on the `Comparable` interface to manipulate `Comparable` objects without knowing anything else about them. A case in point is the `Arrays` class `parallelSort` static method will allow us to sort an array of `Comparable` objects. For more information on that method, see the documentation here.

One detail about implementing the `Comparable` interface deserves mention. The `Comparable` interface provides a means to compare two instances of **anything** that supports that interface. In our case, it will be two instances of Person. But, it could be two instances of Question, two instances of Country, really anything that you can imagine. To accommodate that, we have to tell the compiler the class that we will be comparing **to** in the implements construct. To do that, we will use the same syntax that we used to create an `ArrayList`: `<class name>`. The resulting syntax looks like: `public class Person implements Comparable <Person>`.

The `Measurable` interface provides a similar abstraction role. This interface only requires that a class that implements `Measurable` provide a method called getMeasure() to retrieve **some** double precision value that represents a measure of interest for that particular class. In our case, we will use a person's age as the value that `Measurable` returns. We will then use that interface to search an array of `Measurable` objects to find the smallest one.

PROCEDURE:

1. Build a new project named "CECS 277 Lab Interfaces and Polymorphism".
    a. Name the corresponding module "cecs277LabInterfacesAndPolymorphism".
    b. Create a package within that project with the same name as the module.
2. Within the cecs277LabInterfacesAndPolymorphism package, create your `Measurable` interface.
    a. The interface has just one method: `getMeasure` that takes no arguments and returns a double precision value.
3. Create a Class called Data, from the code that you find here.

4. Create a Class called Person, from the code that you find here.
5. Write a driver class, call it InterfaceAndPolymorphismRunner:
    a. Load up a small array with instances of the Person class.
    b. Display that array of Persons.
    c. Use the Arrays class parallelSort method to sort the Persons.
    d. Display the sorted array of Persons.
    e. Use the average method in the Data class to find the average age.
    f. Print out that average age.

WHAT TO TURN IN:

- Your new version of Person.java.
- Measurable.java
- InterfaceAndPolymorphismRunner.java
- Your console output as **console.txt**.
- Do not forget to demonstrate your code to me before you leave.