Polymorphism - Pair Exercises

In order to practice applying polymorphism, your task is to create interface definitions and class implementations for the exercises defined below. In any of the cases, you may add attributes and other supporting methods to the classes in order to fully implement the interface.

Worker Class
create an array of
workers

Application Program

~~Worker Interface~~

Attribute Name	Type	Get	Set
`firstName`	`string`	X	
`lastName`	`string`	X	

In Main()
1. Define an ArrayyList of Workders
2. Put a numver of different workders in array.
3. for-each to test all methods
calculate hourly pay (dummy method)

Method	Return Type
`calculateWeeklyPay(int hoursWorked)`	`double`

Salary Worker

Every worker will have a different pay

Create a `SalaryWorker` class that implements `Worker` interface.

the loop will test woker pay

Attribute Name	Type	Get	Set
`annualSalary`	`double`	X	

All inherit form worker
all have weekly pay
define worker
have for-each to test methods

Constructor
`SalaryWorker(String firstName, String lastName, double annualSalary)`

A salaried employee "is-a" worker. hoursWorked express the number of hours for which the salaried employee worked in a given week. Since salary employees are based on a 40 hour week, any hours above or below are ignored and the following formula is used to calculate their weekly salary:

Salary worker calculate

```
pay = annual salary / 52
```

Hourly Worker

Create an `HourlyWorker` class that implements `Worker` interface.

Attribute Name	Type	Get	Set
`hourlyRate`	`double`	X	

Constructor
`HourlyWorker(String firstName, String lastName, double hourlyRate)`

An hourly employee "is-a" worker whose hoursWorked express the number of hours for which the hourly employee is being paid.

```
pay = hourly rate * hoursWorked
overtime = hoursWorked - 40
```

```
pay = pay + (hourly rate * overtime * .5)
```

Volunteer Worker

Create a `VolunteerWorker` class that implements the `Worker` interface.

Constructor
`VolunteerWorker(String firstName, String lastName)`

A volunteer "is-a" worker who's hoursWorked express the number of hours a volunteer gave. There is no monetary amount associated with volunteers, but the hours are recorded as pay.

```
pay = hoursWorked * 0
```

Program

Following the approach that the morning's examples has led, create a List that represents a company payroll and holds a collection of workers in it. The objective will be to:

- output each employee, their number of hours worked (you can use a random number generator), and their weekly pay using the interface
- at the end show the sum of total hours worked and total weekly payroll

Sample Output

```
Employee             Hours Worked          Pay
Mouse, Mickey        90                    $750.00
Geef, George (Goofy) 90                    $0.00
Duck, Daisy          110                   $750.00
Mouse, Minnie        20                    $2100.00

Total Hours: 310
Total Pay: $3600.00
```