Project 3
Worker System[*]

Jiashuo Wang
5100309436

April 21, 2013

Instructor: Ling Gong

I. Objective

This homework revisits the idea of superclass/subclass, polymorphism. And study how to use *Arrays.sort*()**. Remember you need to implement Comparable or Comparator to sort your object array.**

Write a superclass *Worker* and subclasses *HourlyWorker* and *SalariedWorker*. Every worker has a name, a salary rate, and the total salary he got so far from the beginning of his employment. Write a method *computePay*(*int hours*) that computes the weekly pay for every worker. An hourly worker get paid the the hourly wage for the actual number of hours worked, of hours is at most 40. If the hourly worker worked more than 40 hours, the excess is paid at double rate. The salaried worker gets paid the hourly wage for 40 hours, no matter what the actual number of hours is. The total salary is just the accumulated value of all the paid salary for the worker. Beside the *computePay*() method, you can add more methods if you like or think necessary.

After creating an array of Worker, I want to sort the array and print out the sorted result. By default, the workers are sorted by their name. You should also be able to sort it by their salary rate, or by the total salary they've got. This means if I use *Arrays.sort*(*arrayname*) directly, I'm sorting by name. Otherwise I sort by using different comparators.

When print out the information about the workers using *System.out.println*(*worker*), I wish to be able to see the worker's type too, i.e., whether he is *HourlyWorker* or *SalariedWorker*, and with his name, salary rate, and total salary.

II. Algorithm

II.1 Worker Class

There are three different classes about workers here, *Worker*, *HourlyWorker*, *SalariedWorker*. *HourlyWorker* and *SalariedWorker* are extended from *Worker*.

- **Worker**
  It fulfils some basic functions. It can help to store the name, salary rate, total rate of each worker, which is protected that can be used only by the base class and derived classes. And it defines some basic abstract methods, *computePay*(), *toString*() that is used for print the information of the worker, to be implement in the extended classes. When executing the

[*]Designed by LaTeX

constructor to input the information of the worker such as name and salary rate, several check will be made to make sure that all the inputs are legal.

- **HourlyWorker**
  It extends from the *Worker*, which means it implement the undone methods defined in the *Worker* according to the rule.
- **SalariedWorker**
  It also extends form the *Worker*. The differences from the *HourlyWorker* are mainly how to calculate the weekly pay.

II.2 Sort Methods

Sorting the workers is a little easier by using *Arrays.sort*().

As is shown in the description, the workers are sorted by their name by default. That is to say, if I use *Arrays.sort*(*arrayname*) directly, I'm sorting by name. In order to complement it, a rule to compare the two objects by their name is needed. The comparation rule is defined in the the method *compareTo*(), which is in class *Worker* implements from class *Comparable*. In the instantiated *compareTo*() method, the string of name is compared by *compareTo*(). If we get the positive result, we define that this one is larger than the compared one.

And for the sort by their salary rate, or by the total salary they've got, new rules to compare have to defined. In my program I define two class *SalaryRateComparator* and *TotalSalaryComparator* respectively, both of which are implement from class *Comparator*, different from *Comparable*. In these two class, the method *Compare*() is instantiated to create a new rule to compare.

With the three comparation rules shown above, *Arrays.sort*() can be fulfilled successfully. I use *Arrays.sort*(*w*) to sort the workers by names by default. And Sorting by other two is made such as *Arrays.sort*(*w*, *newSalaryRateComparator*()).

II.3 Main Method

In main method, a worker management system is built. First the number of workers is needed, which is used to create an array. Then you can choose to get the weekly salary or to get the worker's information or find a worker or even sort in some orders.

To get the weekly salary, it is simple to use the *computePay*() for each worker. And getting the worker's information is easily fulfilled by using *System.out.println*(*worker*), which is based on *toString*() in each derived class. As to find a worker, just compare the name of each worker, if there is, then use *System.out.println*(*worker*) to print, or the system will tell the user that there is no such person. And the sorting is shown above.

II.4 What's New

According to the basic task, I think more about this worker system. As is shown above, I have added some new:

- By hierarchical framework, the system is easily used for one can only choose the function according to the prompt.
- I add a new function to find some worker, which can be useful practically.
- Considering that the money can only have up to two decimal places, I use the round-off to print the money in a better form by using the class *BigDecimal* and method *setScale*, which is in the java.math.*.

II.5 More details

The format of the input need to be considered, such as name and money. And the choose of menu should also be considered. All of these can be solved by the method, *matches*, which is quite useful. As to names, W Wang or Wang is allowed, while WW or ww or Wang W is not allowed.

III. Results and Conclusions

III.1 Environment

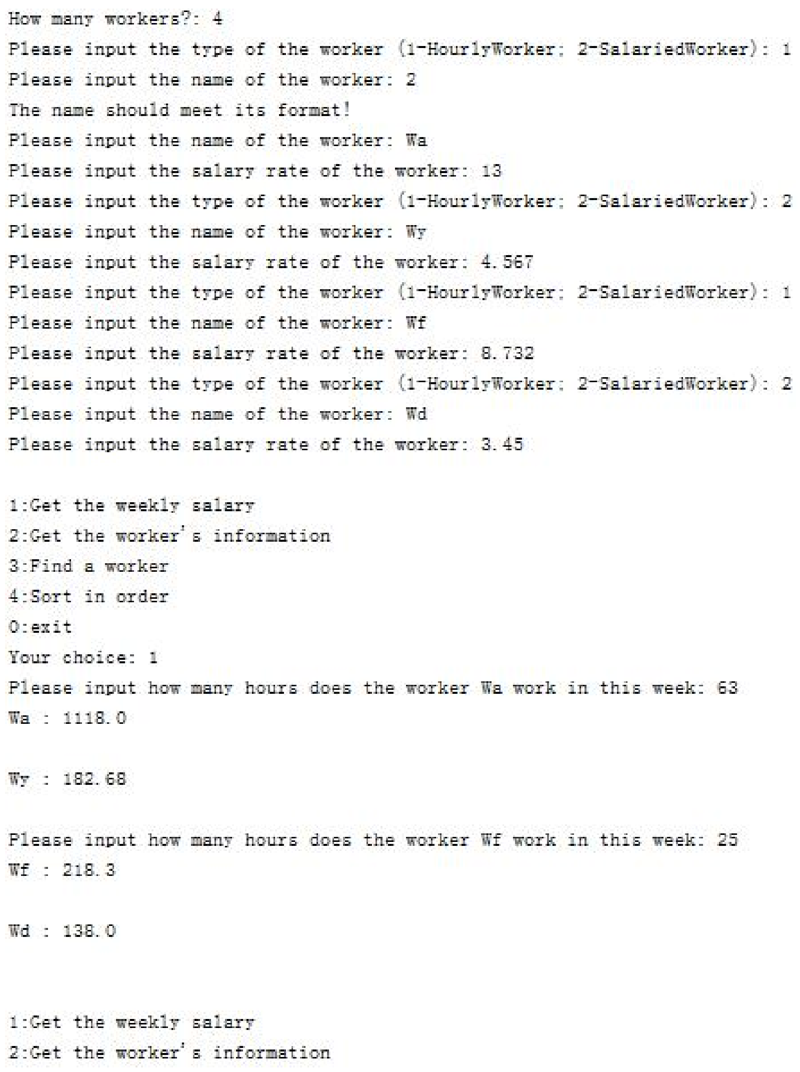
- Windows 8
- NetBeans IDE 7.3

III.2 Screenshots of the result

Use JVM to compile and execute the program in Figure 1 - 4.

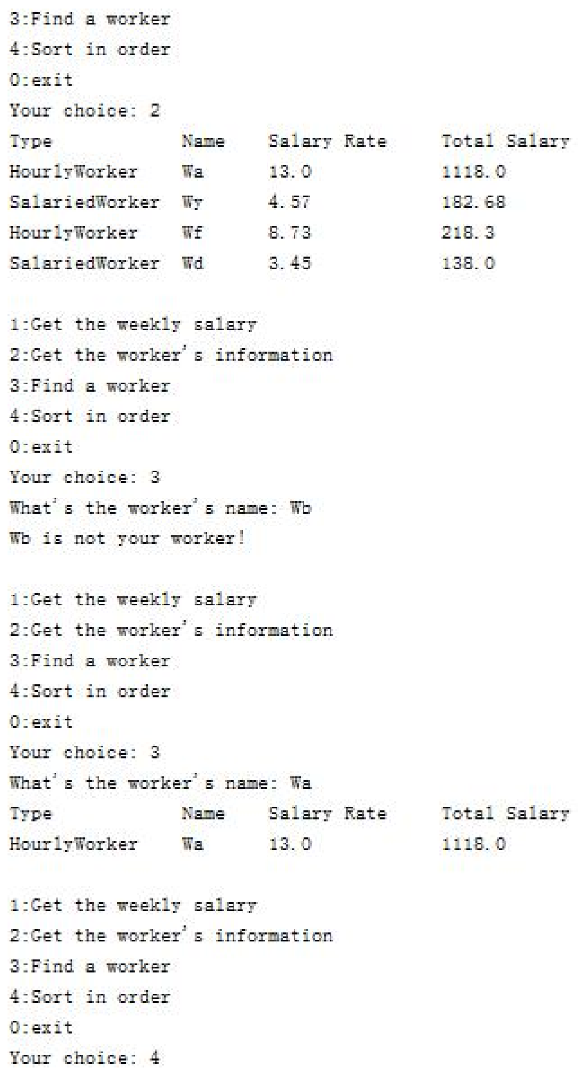
III.3 Thoughts

Thinking more to add some new functions is interesting! And solving problems from something new is interesting, too.



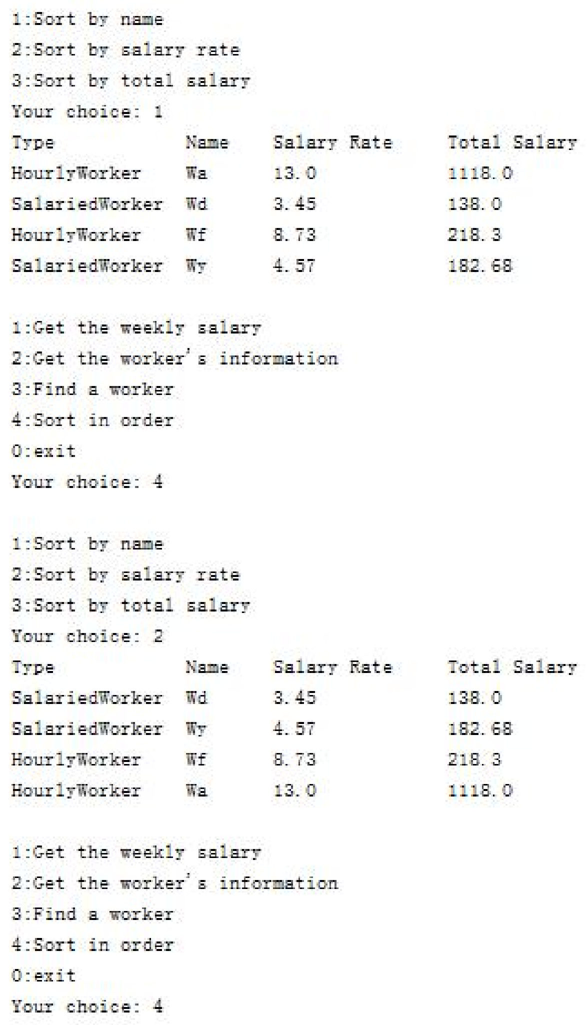
```
How many workers?: 4
Please input the type of the worker (1-HourlyWorker; 2-SalariedWorker): 1
Please input the name of the worker: 2
The name should meet its format!
Please input the name of the worker: Wa
Please input the salary rate of the worker: 13
Please input the type of the worker (1-HourlyWorker; 2-SalariedWorker): 2
Please input the name of the worker: Wy
Please input the salary rate of the worker: 4.567
Please input the type of the worker (1-HourlyWorker; 2-SalariedWorker): 1
Please input the name of the worker: Wf
Please input the salary rate of the worker: 8.732
Please input the type of the worker (1-HourlyWorker; 2-SalariedWorker): 2
Please input the name of the worker: Wd
Please input the salary rate of the worker: 3.45

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 1
Please input how many hours does the worker Wa work in this week: 63
Wa : 1118.0

Wy : 182.68

Please input how many hours does the worker Wf work in this week: 25
Wf : 218.3

Wd : 138.0


1:Get the weekly salary
2:Get the worker's information
```

Figure 1: *Screenshots of Worker System(1)*



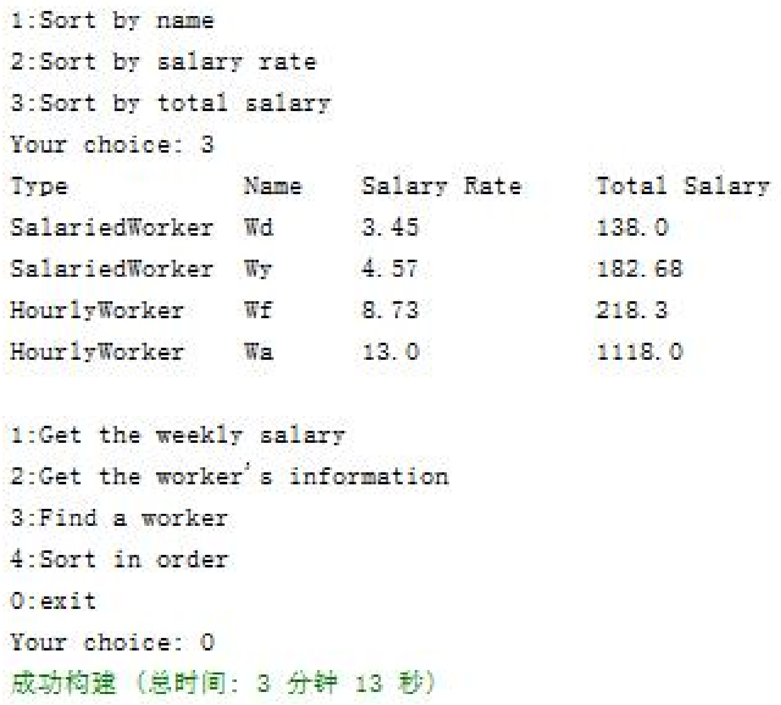
```
3:Find a worker
4:Sort in order
0:exit
Your choice: 2
Type            Name    Salary Rate     Total Salary
HourlyWorker    Wa      13.0            1118.0
SalariedWorker  Wy      4.57            182.68
HourlyWorker    Wf      8.73            218.3
SalariedWorker  Wd      3.45            138.0

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 3
What's the worker's name: Wb
Wb is not your worker!

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 3
What's the worker's name: Wa
Type            Name    Salary Rate     Total Salary
HourlyWorker    Wa      13.0            1118.0

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 4
```

Figure 2: *Screenshots of Worker System(2)*



```
1:Sort by name
2:Sort by salary rate
3:Sort by total salary
Your choice: 1
Type            Name    Salary Rate     Total Salary
HourlyWorker    Wa      13.0            1118.0
SalariedWorker  Wd      3.45            138.0
HourlyWorker    Wf      8.73            218.3
SalariedWorker  Wy      4.57            182.68

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 4

1:Sort by name
2:Sort by salary rate
3:Sort by total salary
Your choice: 2
Type            Name    Salary Rate     Total Salary
SalariedWorker  Wd      3.45            138.0
SalariedWorker  Wy      4.57            182.68
HourlyWorker    Wf      8.73            218.3
HourlyWorker    Wa      13.0            1118.0

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 4
```

Figure 3: *Screenshots of Worker System(3)*



```
1:Sort by name
2:Sort by salary rate
3:Sort by total salary
Your choice: 3
Type            Name    Salary Rate     Total Salary
SalariedWorker  Wd      3.45            138.0
SalariedWorker  Wy      4.57            182.68
HourlyWorker    Wf      8.73            218.3
HourlyWorker    Wa      13.0            1118.0

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 0
成功构建 (总时间: 3 分钟 13 秒)
```

Figure 4: *Screenshots of Worker System(4)*