CS 1302A – HW 2 (Spring 2016)
Total: 100 pts
Due: Wednesday, Feb. 3, 2016 11:59 pm

This homework has three problems which all deal with classes and associations between classes.

Eclipse Reminder

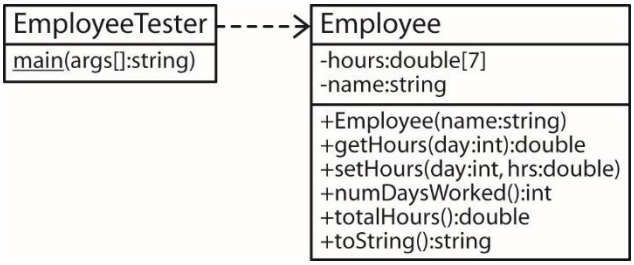
- You will create a Java Project in Eclipse with the name: *hw2_FLastname*
- You will have three packages: *prob1*, *prob2*, *prob3*

Problem 1

Do problem 9.7 from the text.

Problem 2

Write the classes indicated for the following class diagram:



1. Employee class:

    a. The *Employee* class represents an employee of a store and keeps track of the number of hours worked for each day of a week. The hours worked can be set and retrieved for any day of the week. The name cannot be changed. Various other derived information is available: the number of days worked and the total number of hours worked.

    b. *hours* – An instance variable that is a an array of 7 elements. It holds the number of hours worked for each day of the week. Assume the week starts on Monday, *e.g.* position 0 is hours worked on Monday, position 1 is hours worked on Tuesday, *etc.*

    c. *name* – An instance variable that holds the name of the employee.

    d. *Employee(name:string)* – A constructor that accepts a name as an argument.

    e. *getHours(day:int):double* – A method which returns the number of hours worked for *day*, where *day* takes on values 0, 1, ..., 6. For example, *getHours(4)* would return the number of hours worked on Friday.

f. *setHours(day:int, hours:double)* – A method which sets the number of *hours* worked for the indicated *day*. The day must be a value between 0 and 6, otherwise, the method should do nothing. For example, *setHours(2,6.5)* would set the total hours worked for Wednesday to 6.5.

g. *numDaysWorked():int* – A method which returns the total number of days worked. If a day has 0 hours, then it is not counted. For example, if *hours* held the values: 8,8,8,0,8,4,0 then this method would return 5.

h. *totalHours():double* – A method which returns the total hours worked for the week. For example, if *hours* held the values: 8,8,8,0,8,4,0 then this method would return 36.0

i. toString( ):string – A method which returns a message like this:

   Chen worked 5 days for a total of 36.0 hours.

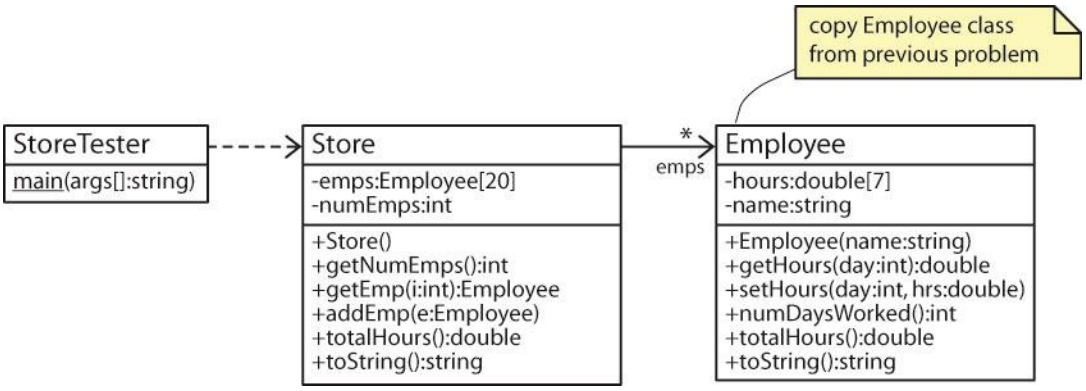
2. EmployeeTester class:

- Write a *main* method that produces a dialog like this:

```
-->Enter a name : Chen
-->Enter hours worked (7 values separated by spaces): 5 2 1 0 0 3 3

Output:
Chen worked for 5 day(s) for a total of 14.0 hours.

-->Enter a day of the week (0-6): 2

Output:
Hours worked on day 2 is 1.0
```

Problem 3

Write the classes indicated for the following class diagram. **Copy** your Employee class from problem 2.



1. Store class:

a. The *Store* class is used to manage the collection of *Employee* objects that exist for the store. In other words, a store may have 5 employees so that its *emps* array would have the first 5 elements filled with references to these employees.

b. *emps* – An instance variable, which is an array that can hold up to 20 Employee objects. Employees are stored sequentially, with no gaps, starting with position 0. Employees can be added but not removed.

c. *numEmps* – An instance variable that stores the total number of Employee objects in the *emps* array. Initially this value is 0.

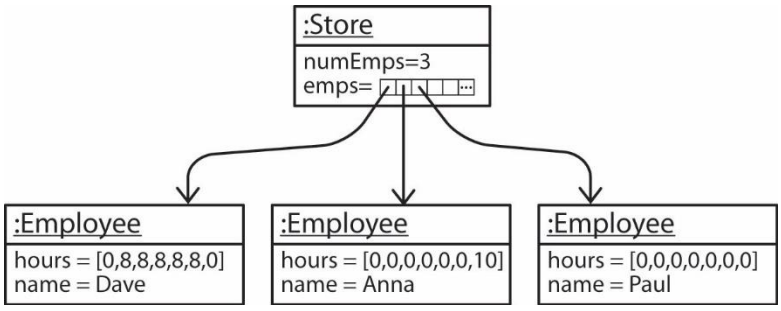
d. *getNumEmps():int* – Returns the number of employees. Getter for *numEmps*.

e. *getEmp(i:int):Employee* – Returns the employee at position *i* if there is one, otherwise returns *null.* For example to return the 3rd employee, you would return *emps[2]*.

f. *addEmp(e:Employee)* – Adds the Employee, *e* in the next available slot. Does nothing if there are no more slots. For example if there are 3 employees then *e* will be stored in *emps[3]* and the number of employees will become 4.

g. *totalHours():double* – Returns the total number of hours worked over all the employees. Hint: loop through the *emps* and get the total hours for each one and add them up.

h. *toString():string* – Consider this object diagram as an example:



The *toString* method should return a message like this:

Num Employees: 3
Dave worked 5 days for a total of 40.00 hours
Anna worked 1 days for a total of 10.00 hours
Paul worked 0 days for a total of 0.00 hours
Total Hours Worked = 50.00

2. StoreTester class – use this *main* method exactly

```
public static void main(String[] args) {
        Store store = new Store();
        Employee e = new Employee("Dave");
        for(int i=1; i<6; i++) e.setHours(i, 8);
        store.addEmp(e);

        e = new Employee("Anna");
        e.setHours(6, 10);
        store.addEmp(e);

        e = new Employee("Paul");
        store.addEmp(e);
```

```
        System.out.println (store);
        System.out.println (store.getEmp(1));
    }
```

The main method should print out the following:

```
Num Employees 3
Dave worked for 5 day(s) for a total of 40.0 hours.
Anna worked for 1 day(s) for a total of 10.0 hours.
Paul worked for 0 day(s) for a total of 0.0 hours.
Total Hours Worked= 50.0
Anna worked for 1 day(s) for a total of 10.0 hours.
```

Submission

Follow the directions in Lab 1 to zip the folder *hw2_FLastname*.

Make sure you zip the ENTIRE folder *hw2_FLastname*!! Submit your zip file to Blazeview by the due date. The name of your file should be: *hw2_FLastname.zip*.

Grading

I will RANDOMLY select one problem for grading. For example, if problem 3 is selected, I will only go to the package/folder *prob3* for grading. So make sure all your Java codes are in the correct packages/folders.

Additional Requirements:

1) No late submission will be accepted.
2) Please exactly follow the naming rules described above. You will be deducted 10 points for incorrect naming.
3) Write comments at the beginning of your Java source file(s) to indicate your name, student ID, "CS 1302-A Homework 2", and the due date.
4) Make sure that your programs are well-documented, readable, and user friendly. Make sure you document your programs by adding comments to your statements/blocks so that I can understand your code. You will get 0 to 10 points based on the helpfulness of your comments. You will be deducted 10 points for no comments.
5) It is your responsibility to make sure your programs can compile and run correctly. Please be aware that programs that do not compile may receive ZERO credit.
6) When grades are returned to you via BlazeView, you have 7 days to meet with the instructor to discuss on grade changes. After 7 days, the grades are written in stone and can't be changed after that point.