CS 1302 – HW 2

*1-many Association Implemented with an Array*

Contents

[1 Overview 2](#_Toc176419141)

[2 Setup Project with Jar File 3](#_Toc176419142)

[3 Notes about the Test Code 4](#_Toc176419143)

[4 Steps to Complete 4](#_Toc176419144)

[4.1 Phase 1 – *addEmployee* 4](#_Toc176419145)

[4.2 Phase 2 - *toString* 5](#_Toc176419146)

[4.3 Phase 3 – *getEmployee* 5](#_Toc176419147)

[4.4 Phase 4 – *getTotalHours* 5](#_Toc176419148)

[4.5 Phase 5 – *getTotalPay* 5](#_Toc176419149)

[4.6 Phase 6 – *removeEmployee* 5](#_Toc176419150)

[4.7 Phase 7 – *getEmployeeWithName* 5](#_Toc176419151)

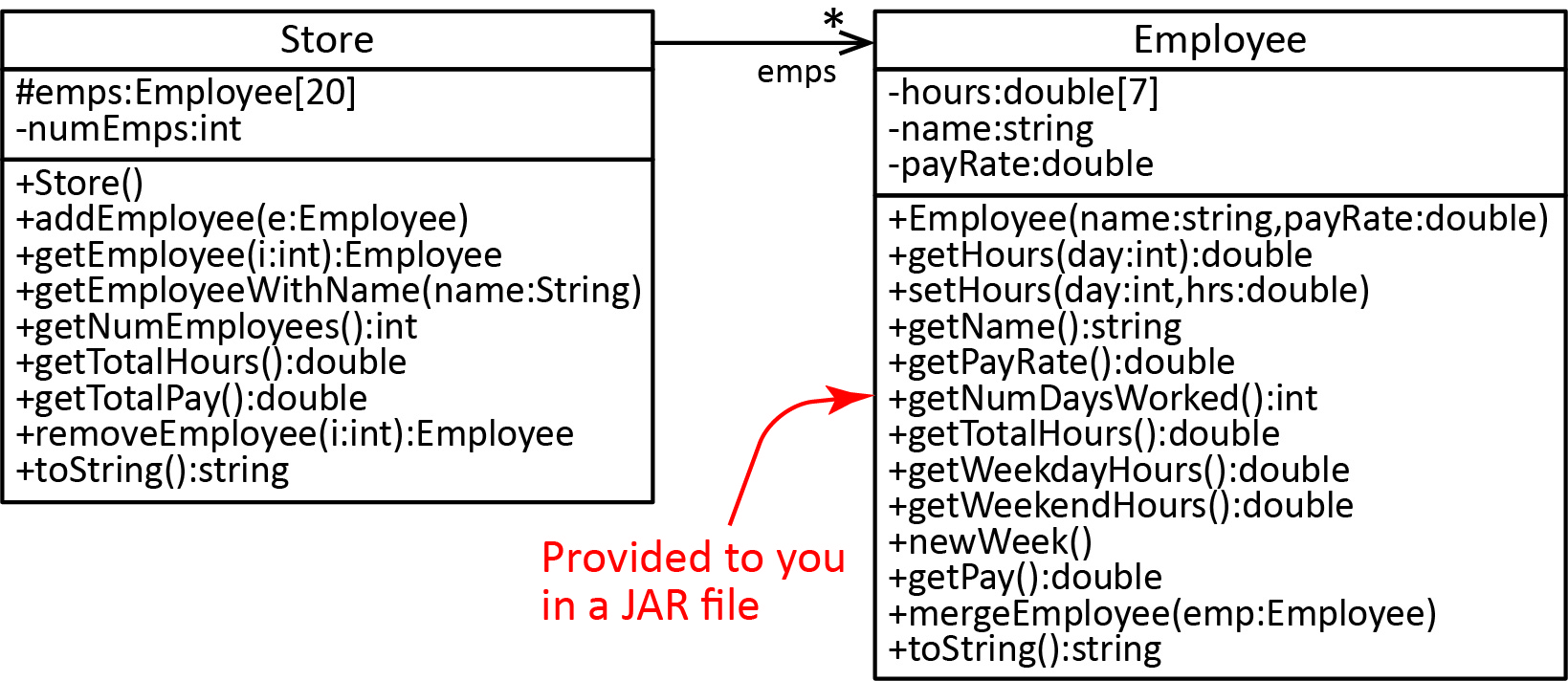
[4.8 Phase 8 – *HW02CompileTest* 6](#_Toc176419152)

[5 Grading Criteria 6](#_Toc176419153)

[6 Submission Requirements 6](#_Toc176419154)

# Overview

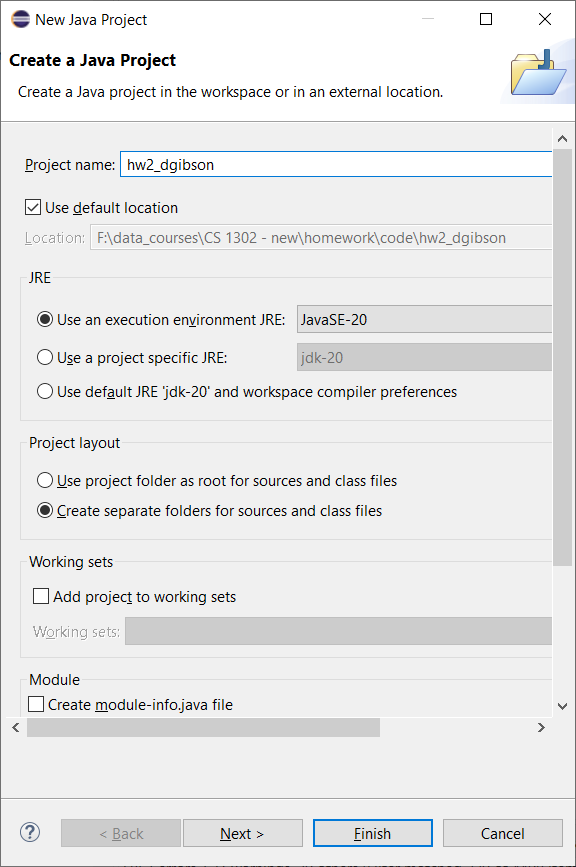
1. You will write the *Store* class shown in the class diagram below. The *Employee[[1]](#footnote-1)* class has been provided to you in a JAR file[[2]](#footnote-2). Instructions for using the JAR file are in the next section. Documentation for the *Store* and *Employee* classes is provided on the Homework page.



1. You have been provided the *StoreTest* class. It contains 20 test methods, 5 of which have been written. You will write the rest of them.
2. Ensure that the provided *HW02CompileTest.java* compiles. This ensures that you have properly implemented the signature of the methods which allows my grading program to operate on your code.
3. Type this Academic Honesty statement followed by your full name, as a comment at the top of the *Store* class:

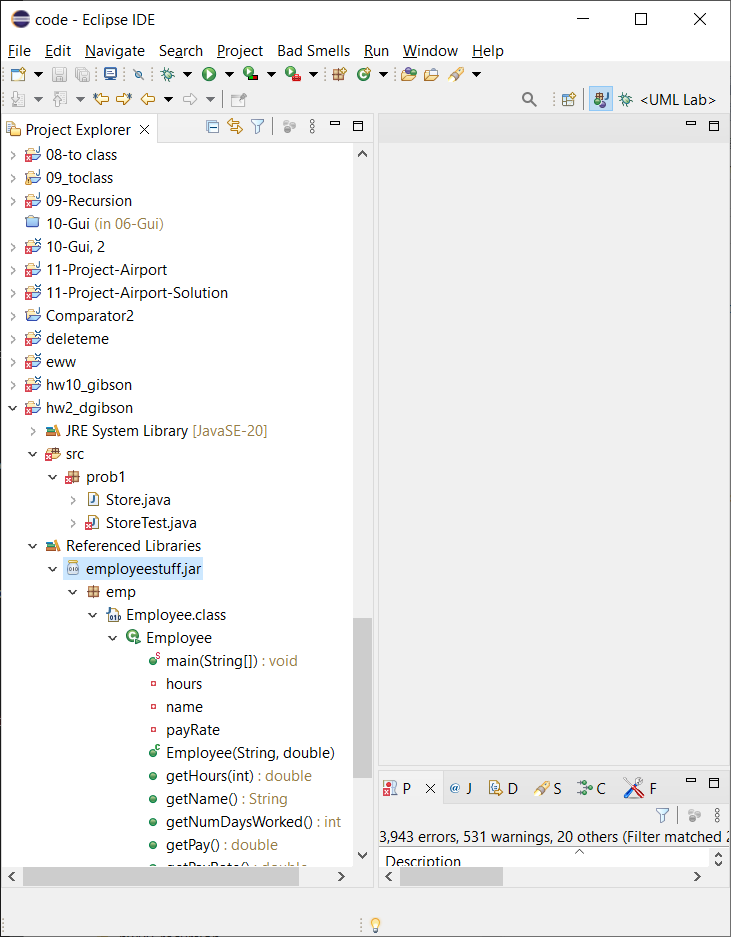
“This homework represents my own work. I understand that I may receive help, but I did not copy any portion of this assignment from anywhere. I understand that a violation of this will result in a Report of Academic Dishonesty.—YOUR FULL NAME HERE”

# Setup Project with Jar File

Follow the steps below to setup your HW 2 project to use a Jar file.

1. In Eclipse, create a Java Project named *hw2\_lastname*. **As shown on the right at the bottom, Uncheck, “Create module-info.java file”.**
2. Download and unzip: *hw2\_code.zip*. You will find a *prob1* folder that contains two files: *StoreTest.java* and *employeestuff.jar.*
3. Drag the *prob1* folder into the *src* node in the Package Explorer
4. Create a class (with a *main*) in the *prob1* package named *Store.*
5. Right-click *employeestuff.jar* and choose: Build Path, Add to Build Path.

Notice (see figure below) that a *Referenced Libraries* node has been added to the project and *employeestuff.jar* has been moved there, as shown in the figure on the right. If you expand the *employeestuff.jar* node you will see the *emp* package and the *Employee* class as well as the class members.



1. Add the import below to the *Store* class. This import the *Employee* class in the Jar file

import emp.Employee;

**If you get a compile error then start completely over with a new project.**

1. Test by creating an *Employee.* Add this line to *main* in the *Store* class*:*

Employee e = **new** Employee("Will", 22.33);

System.***out***.println(e);

This should compile and run properly; if it doesn’t, start over. Note that *StoreTest* does not yet compile.

# Notes about the Test Code

The names of the 20 test methods in *StoreTest* are shown below. The highlighted ones are written for you. Method stubs[[3]](#footnote-3) have been provided for the others with comments above each to describe what the test method should do.

*testAddEmp\_Add\_1*();

*testAddEmp\_Add\_3*();

*testAddEmp\_Add\_20*();

*testAddEmp\_Add\_21*();

*testGetEmp\_5\_Emps\_Loc\_Minus\_2*();

*testGetEmp\_5\_Emps\_Loc\_0*();

*testGetEmp\_5\_Emps\_Loc\_2*();

*testGetEmp\_5\_Emps\_Loc\_4*();

*testGetEmp\_5\_Emps\_Loc\_5*();

*testGetTotalHours\_3\_Employees*();

*testGetTotalPay\_3\_Employees*();

*testRemoveEmployee\_With\_5\_Employees\_Loc\_Minus\_2*();

*testRemoveEmployee\_With\_5\_Employees\_Loc\_0*();

*testRemoveEmployee\_With\_5\_Employees\_Loc\_2*();

*testRemoveEmployee\_With\_5\_Employees\_Loc\_4*();

*testRemoveEmployee\_With\_5\_Employees\_Loc\_5*();

*testRemoveEmployee\_With\_20\_Employees\_Loc\_19*();

*testGetEmployeeWithName\_Found*();

*testGetEmployeeWithName\_NotFound*();

*testToString*();

Towards the bottom of *StoreTest,* there are four helper methods (fully implemented) which you can use:

* *createTestEmployee* – Creates and returns an *Employee* object with 5 hours Monday-Friday.
* *createStoreWith3Employees* – Creates and returns a *Store* object with 3 *Employee* objects with varying hours.
* *createStoreWith5Employees* – Creates and returns a *Store* object with 5 *Employee* objects with varying hours.
* *createStoreWith20Employees* – Creates and returns a *Store* object with 20 *Employee* objects.

4 of the test methods that are written for you use one of these helper methods. You should take a look at these helper methods as they will save you time.

# Steps to Complete

The development work is broken into Phases below to emphasize: writing a method, then write code to test it, and evaluate the results.

## Phase 1 – *addEmployee*

1. Add the three instance variables to the *Store* class. **MAKE THE *hours* INSTANCE VARIABLE PROTECTED, NOT PRIVATE.**

**protected** Employee emps[] = **new** Employee[20];

1. Add the no-arg constructor to that initializes the number of employees to 0.
2. Write the *getNumEmployees* method.
3. Write the *addEmployee* method.
4. In *StoreTest* write these test methods:

*testAddEmp\_Add\_1* (provided)

*testAddEmp\_Add\_3*

*testAddEmp\_Add\_20*

*testAddEmp\_Add\_21*.

|  |
| --- |
| **Reminder** (this pertains to all of the test methods you will write):   * Each test method should have a comment explaining what you are testing (this is already provided in the *StoreTest*). * *main* should simply call each of the test methods one after the other (this is already provided in the *StoreTest*). * Each test method should be stand-alone and independent of other test methods. For example, each test method will create a *Store* object, starting from scratch, and building the *Store* object so that it can be tested. It is useful, sometimes, to create helper methods to create things. As previously mentioned, several helper methods are provided in *StoreTest*. * Each test method should display nicely formatted output to the console that shows that the expected and actual results that can be used to verify that the method work correctly. * **Do not expect full credit for this unless you have written thorough, organized, self-documenting test code that produces meaningful output.** |

1. Run the add tests, verify out, debug as necessary.

## Phase 2 - *toString*

1. Write the *toString* method. The *toString* method can be made over complicated, when in fact most of the work is already done for you. You just display 3 values about the store, and then loop over the *employees* array and call *toString* on each of the employees. In other words, the *Employee*’s *toString* returns a “Pay Stub”. See the documentation for the exact format the return should be in.
2. In *StoreTest*, run *testToString* (method is provided), verify output, debug as necessary.

## Phase 3 – *getEmployee*

1. Write the *getEmployee* method.
2. In *StoreTest* write these test methods below. Run tests, verify output, debug as necessary.

*testGetEmp\_5\_Emps\_Loc\_Minus\_2* (provided)*,*

*testGetEmp\_5\_Emps\_Loc\_0,*

*testGetEmp\_5\_Emps\_Loc\_2,*

*testGetEmp\_5\_Emps\_Loc\_4,*

*testGetEmp\_5\_Emps\_Loc\_5*

## Phase 4 – *getTotalHours*

1. Write the *getTotalHours* method.
2. In *StoreTest* write this test method: *testGetTotalHours\_3\_Employees*. Run test, verify output, debug as necessary.

## Phase 5 – *getTotalPay*

1. Write the *getTotalPay* method.
2. In *StoreTest* write this test method: *testGetTotalPay\_3\_Employees.* Run test, verify output, debug as necessary.

## Phase 6 – *removeEmployee*

1. Write the *removeEmployee* method.
2. In *StoreTest* write these test methods below. Run tests, verify output, debug as necessary.

*testRemoveEmployee\_With\_5\_Employees\_Loc\_Minus\_2,*

*testRemoveEmployee\_With\_5\_Employees\_Loc\_0* (provided)*,*

*testRemoveEmployee\_With\_5\_Employees\_Loc\_2,*

*testRemoveEmployee\_With\_5\_Employees\_Loc\_4,*

*testRemoveEmployee\_With\_5\_Employees\_Loc\_5,*

*testRemoveEmployee\_With\_20\_Employees\_Loc\_19*

## Phase 7 – *getEmployeeWithName*

1. Write the *getEmployeeWithName* method.
2. In *StoreTest* write these test methods below. Run tests, verify output, debug as necessary.

*testGetEmployeeWithName\_Found* (provided)

*testGetEmployeeWithName\_NotFound*

## Phase 8 – *HW02CompileTest*

If *HW02CompileTest* compiles and runs, you are done.

If not, any compile error in this code means that the signature (spelling of method, return type, number of parameters, type of parameters) of the method YOU wrote is incorrect. Usually, the spelling is incorrect, but sometimes you get data types wrong. **Change your code so that *HW02CompileTest* compiles and runs.**

* If you didn't implement a method, then add a "stub" to your code so that this class compiles and to make the grading process go more smoothly on my end. Some example stubs for various methods:

|  |  |
| --- | --- |
| **public** **double** getPay() {  **return** Double.MAX\_VALUE;  } | **public** **void** mergeEmployee(Employee e) {  } |

**public** Employee getEmployeeWithMostHours(Employee[] emps) {

**return** null;

}

* If you need to correct the spelling of a method, be sure and use: Refactor/Rename, which changes all occurrences in all files. See Lab 2 if needed.

# Grading Criteria

|  |  |
| --- | --- |
| **Grading Criteria** | **Points** |
| *Store* | 75 |
| *StoreTest* | 25 |
| **Total** | **100** |

# Submission Requirements

Checklist:

|  |  |  |
| --- | --- | --- |
|  | **Complete?** | **Requirement** |
| 1. |  | *HW02CompileTest* compiles. |
| 2. |  | The Academic Honesty statement followed by your full name, appears as a comment at the top of the *Store* class. |
| 3. |  | All java files (including test classes and JAR file) are in the *prob1* package. |
| 4. |  | Your *prob1* folder is zipped into a file name: *hw2\_yourLastName.zip.*   * See Lab 2, Stage 9 for exact instructions. * Do not zip your workspace folder * Do not zip your *src* folder. * Do not zip just the java files * Do zip just your *prob1* folder |
| 5. |  | Submit in the *hw2* dropbox on Blazeview by the deadline. |

1. This is the *Employee* class from HW 1; however, you will use my version of the class to prevent any errors you may have had in yours from affecting this HW. [↑](#footnote-ref-1)
2. A Jar file is the standard way to distribute Java class files. You can only use the *Employee* class, you cannot modify the code, <https://en.wikipedia.org/wiki/JAR_(file_format)>, [↑](#footnote-ref-2)
3. A [*method stub*](https://en.wikipedia.org/wiki/Method_stub)is a method with no code, or if the method returns something, it returns an arbitrary value. [↑](#footnote-ref-3)