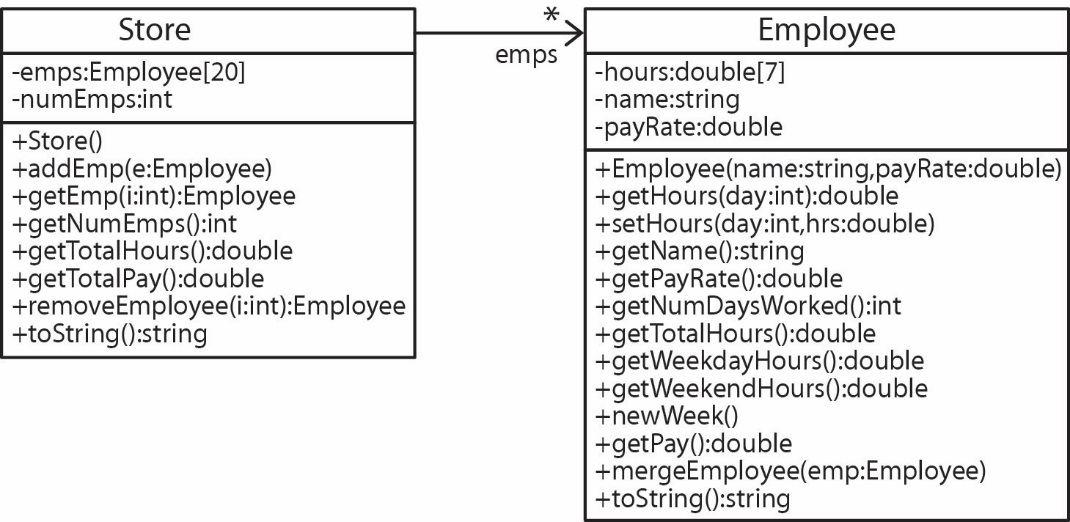
**Coding Assignment 2**

You can work individually or in groups of 2.

**Problem**

Consider the *Store* and *Employee* classes below. The *Employee* class was considered in a previous assignment and you can consider it to be fully tested. You will use the category-partition method to construct a test specification and JUnit tests for the two *Store* methods: *addEmp* and *removeEmployee.*



1. *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.
2. *numEmps* – An instance variable that stores the total number of Employee objects in the *emps* array. Initially this value is 0.
3. *addEmp(e:Employee)* – Adds the Employee, *e* in the next available position. If there are already 20 employees and there is an attempt to add another then this method should do nothing, but should not crash.
4. *getEmp(i:int):Employee* – Returns theemployee at position *i* if there is one, otherwise returns *null.*
5. *getNumEmps():int* – Returns the number of employees.
6. *getTotalHours():double* **–** Returns the total number of hours worked over all the employees.
7. *getTotalPay():double* **–** Returns the total pay worked over all the employees.
8. *removeEmployee(i:int):Employee* – Removes theemployee at position *i* if there is one and returns it. All other employees to the right should be moved over one position to the left. If *i* is out of range then return null.
9. *toString():string*– Returns a message with this format: like the one shown below (for example, if there are 3 employees).

**Steps**

1. You will keep a Time Log for this (and all) assignments. Please see the *Time Log* section below.
2. Use the category-partition method to develop a test specification for *addEmp.* Document the steps you took to arrive at this providing rationale as appropriate. The more I can see your thought process the better. Type your results in a document in the format shown in Appendix A.
3. Repeat Step 2 using *removeEmployee.*
4. Find the *code* link on the Schedule and download *ca2.zip.* Drag the *Store* class into an Eclipse project. Drag in the *Employee* class from CA 1.
5. Develop JUnit tests for each method from the specification.

**Deliverables**

1. Code – zip the *prob1* package which includes your JUnit test class, *Store* class, *Employee* class, time log(s), and Test Derivation document, and possibly your time log(s) if not contained in Test Derivation Document. Name the zip file: LastName1\_LastName2.zip and submit on Blazeview in the dropbox named, *CA 2*.

**Time Log**

If working in a group, each person should make a Time Log. They can be submitted as separate files or both on one page. If submitting with separate files, name the file: your\_name\_time\_log.docx

**Track your time using this table. Delete empty rows, add if needed. Put the total time at bottom. Delete everything above and submit (in your *prob1* folder) everything below saved as a word document.**

**Coding Assignment 2 - Time Log**

Name:

|  |  |
| --- | --- |
| **Time (hrs)** | **Task** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **Total Time (hrs)** |

**Appendix A**

Document template for submission. Delete this line and everything above it before submitting. Name file: ca2\_lastName1\_lastName2\_testDerivation.docx.

**Coding Assignment 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Name(s) |  |  |  |

***addEmp(e:Employee)* Test Derivation**

[Format anyway you choose. Remove this comment before submitting.]

***removeEmployee(pos:int):Employee* Test Derivation**

[Format anyway you choose. Remove this comment before submitting.]

***Time Logs***

[Can include the Time Log(s) here or as separate files. Remove this comment before submitting.]