* **Lab Assignment One**

|  |  |
| --- | --- |
| Attached Files: | * + [File](https://nuigalway.blackboard.com/bbcswebdav/pid-381209-dt-content-rid-1623366_1/xid-1623366_1) PayrollSystem.zip (2.843 KB) |

– Modify the supplied payroll system to include private instance variable ***joinDate*** in class ***Employee***to represent when they joined the company. Use the java.util.***Date*** for this variable.

– Use a static variable in the **Employee** class to help automatically assign each new employee a unique (incremental) id number.

– Assume the payroll is processed once per month. Create an array of ***Employee*** variables to store references to the various employee objects.

– In a loop, calculate the payroll for each ***Employee***, and add a €100.00 bonus to the person’s payroll if they joined the company over 10 years ago.

For submitting the assignment, please submit all .java files via Blackboard/TurnItIn. You do not need to write up a report.

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Two**

Using the classes supplied with Assignment One:  
  
- Change the Earnings() method in Employee and all sub-classes to throw a generic Exception if the total earnings would be less than zero. The exception should have a message with the employee's name and the error encountered.  
  
- Modify the Test class to be able to handle exceptions. When an exception is encountered calculating an employee's earnings, the Test class should print out the error message and continue as normal with the next employees.  
  
- Test this by changing the Test class so that two of the employees will have negative earnings.  
  
Note: you may have to remove the "if less than zero, set to zero" [wagePerHour > 0 ? wagePerHour : 0] parts of the subclasses.

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Three**

Write a Java class called Rational for performing arithmetic with fractions.  
  
Use integer variables to represent the two private variables of the class: the numerator and the denominator, and corresponding Getter methods.  
  
Also provide public methods for addition, subtraction, multiplication and division of two fractions, and use the formatted output functions to print out the result to the console.  
  
  
Write a short driver program that could be used to test your class. Pass the various runtime parameters as strings in the form of x/y, use the Enhanced For Loop to loop through them, and then use a StringTokenizer to divide the inputted fractions into numbers. Note that StringTokenizer(string, "//" ) will tokenize the string using backslashes as the divider instead of spaces.  
  
For Eclipse, the runtime (or command line) arguments can be set in the Run Configuration window (Program Arguments).  
  
Still in that loop, use the methods created earlier to add, subtract, multiply and divide the current fraction with the one prior. The first fraction inputted can be ignored.  
  
  
Put the fraction-related code in a new package called lab3code, and the test code in a package called lab3test. Remember - the difference between a public method and a method with no modifier is that only one of them can be accessed outside of its package.

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Four**

1. Write a Java application that prompts the user to input their Name, Address, Date of Birth and Student ID number using the standard input - this information should then be saved to a file named *studentData.*The program should use the FileWriter class and an appropriate processing stream to handle the data output.

2. Create a separate class to read in that file, line by line, and output it to the console with the line number. The program should create a **FileReader** object and pass this in the constructor of a **LineNumberReader** object to handle the file reading required

The LineNumberReader class has two useful methods (that could be used):

**public String readLine() throws IOException;**

This method reads a line of text. It returns a String containing the contents of the line, not including any line termination characters, or null if the end of the stream has been reached.

**public int getLineNumber();**

This method returns the current line number.

*System.in - this information should then be saved to a file named* studentData. The program should use the *FileWriter class and an appropriate processing stream to handle the data output.****FileReader object and pass this in the constructor of a****LineNumberReader object to handle the file reading required.*

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Five**

* + Write a simple Java program that creates an array of *StudentData* objects (name, Date Of Birth, and ID), including a toString override to print out all variables. It then serializes the array (and not the StudentData objects individually) and saves it to a file named *studentdata.txt*. It should then read the array from the file and reconstruct a new array object from this data, and print out each item in the array to the console.
  + Copy and rename these two classes. This time, use custom readObject() and writeObject() override methods in StudentData to read and write the object's variables. Do *not* use defaultWriteObject or defaultReadObject methods to do this.
  + Unrelated to the previous two parts, write a simple program that uses a RandomAccessFile object to open an ordinary text file and to then append an extra line of content onto the end of the file.
* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Six**

* + Modify the Rational class so that it correctly implements the comparable interface i.e. each object should be able to compare itself to another object of the same type.   (3 marks)
  + Write a Java program that uses an *ArrayList* to store a collection of 10 Rational objects. Write a simple algorithm that manually iterates through the contents of the list and places them in ascending (i.e. smallest Rational to the biggest Rational) order into a new list. You can use a simple bubble sort or similar algorithm to do the sorting – do NOT use the built-in sort algorithm at this stage. (6 marks)
  + Print out the contents of the list at each stage to verify the results. (1 marks)
* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Assignment 7**

* + Use the Rational class that you modified in the previous assignment, so that it correctly implements the comparable interface.
  + Write a Java program that uses an *ArrayList* to store a collection of Rational objects. Then use the built-in Collections.sort() method to sort the contents of the list. Print out the contents of the list to verify the results.
  + Modify the program to use your own Comparator object i.e. write a class that implements the java.util.Comparator interface and then use the version of the sort() method that allows you to pass your own Comparator object (the sort() method will take 2 arguments). Print out the contents of the list to verify the results.
  + Use the Collections.shuffle() method to shuffle the elements in your list, print out the results and then sort them again (using Collections.sort(), again printing to confirm)
  + Modify the program to use the built-in Collections.binarySearch() method to check if a particular fraction is contained within the list. Print out your results. Use a JOptionPane input for this part.
* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Eight**

Develop a simple GUI-based Java program that may be used to control a lightbulb. Use suitable Swing components to allow the lightbulb controller to perform the following functions:

1. Switch the lightbulb system on or off.

2. Choose a scheduled time from a list - either morning, evening, or all day long.

3. Light intensity buttons - 20W, 40W or 60W.

4. Display the current status of the lightbulb (aka:, display the currently selected settings).

–  Show the top-level design of the GUI, including any Panels and related Layout Manager objects that you propose to use.

–  For each of the components you’ve chosen above, write the code to construct the component, add the component to a container and then setup simple event handling for the component (for those that generate events).

–  The event handlers need only print out a message indicating that they have been called.

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Nine**

* + Write a simple BankAccount class that may be safely accessed by multiple threads of execution at the same time (include deposit/withdraw methods). Write a driver program that creates an instance of the BankAccount class and then creates several concurrent threads that use the various methods of the same bank account object. Print the output from your program to see how it operates.
  + Create a Java class called FlashingLabel that takes a text string and displays it centered in an Applet. The text should blink on and off at a rate determined by the sleepTime parameter switching between 2 strings each time. The font, the text string to be displayed, and the sleepTime (in milliseconds) should be passed as parameters from html. - (tip: the getParameter methods should be called in the Init method, or a method called after it, and not at the start of the file.)
  + Extend the basic applet to allow a left mouse button click start and stop the flashing.
* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Ten**

* + **Modify the IntBuffer class (in the notes) to use the Java Concurrency API.**
    - **In particular, use the ReentrantLock and Condition classes to implement similar functionality.**
  + **Run the modified program to verify that it still works as before i.e. facilitates safe access for multiple threads to the IntBuffer object.**

Note: The aim of this assignment is comprehension of the flow of control

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Eleven**

Create server and client programs that allows files to be uploaded and downloaded to the server.  
  
The server should be multi-threaded, and have one thread per connection.  
  
The client should be able to specify whether to upload or download a file, as well as the file name (or file directory).  
  
*Note that due to the two-week timeframe, this assignment is notably longer than the other assignments - it is strongly recommended that you do not leave it to the last week. Also that an IP address may be used instead of a host name (and 127.0.0.1 is to connect to the same computer), and remember that FileInputStream and FileOutputStream are used to convert bytes into a file.*

* https://nuigalway.blackboard.com/images/ci/sets/set08/assignment_on.gif

**Lab Assignment Twelve**

 Re-implement the file transfer application, but this time using Datagram sockets and multiple packets. This means using a UDP protocol instead of TCP.  
  
This means that your application may receive packets out of order, and some packets may be dropped. You must make the code be able to organise the incoming packets into the proper order, and ensure that all information has been received - unreceived packets should be re-sent. For this, you should have sequence numbers and make good use of timers.