Seneca College
Applied Arts & Technology
SCHOOL OF COMPUTER STUDIES

JAC444 **Submission date:** **August 11, 2020**

Workshop 10

Description:

The tasks in this workshop is a recap of some concepts from previous workshops and some new concepts like Comparable Interface and Cloneable Interface.

Task 1:

Define a class named **Time** for encapsulating a time. The class contains the following:

1. A data field of the long time that stores the elapsed time since midnight, Jan 1, 1970.
2. A no-arg constructor that constructs a Time for the current time.
3. A constructor with the specified hour, minute, and second to create a Time.
4. A constructor with the specified elapsed time since midnight, Jan 1, 1970.
5. The **getHour()** method that returns the current hour in the range 0-23.
6. The **getMinute()** method that returns the current minute in the range 0-59.
7. The **getSecond()** method that returns the current second in the range 0-59.
8. The **getSeconds()** method that returns the elapsed total seconds.
9. The **toString()** method that returns a string such as "1 hour 2 minutes 1 second" and "14 hours 21 minutes 1 second".
10. Implement the Comparable<Time> interface to compare this Time with another one based on their elapse seconds. The compareTo method returns the difference between this object's elapse seconds and the another's.
11. Implement the Cloneable interface to clone a Time object.

Write a test program that produces the following sample run:

Sample Run - 1

```
Enter time1 (hour minute second): 331 34 674

19 hours 45 minutes 14 seconds

Elapsed seconds in time1: 1194314


Enter time2 (elapsed time in seconds): 93889345

16 hours 22 minutes 25 seconds
```

```
Elapsed seconds in time2: 93889345


time1.compareTo(time2)? -92695031


time3 is created as a clone of time1
time1.compareTo(time3)? 0
```

End Sample Run - 1

Sample Run - 2

```
Enter time1 (hour minute second): 1 2 3

1 hour 2 minutes 3 seconds

Elapsed seconds in time1: 3723


Enter time2 (elapsed time in seconds): 193032 <Enter>

5 hours 37 minutes 12 seconds

Elapsed seconds in time2: 193032


time1.compareTo(time2)? -189309


time3 is created as a clone of time1
time1.compareTo(time3)? 0
```

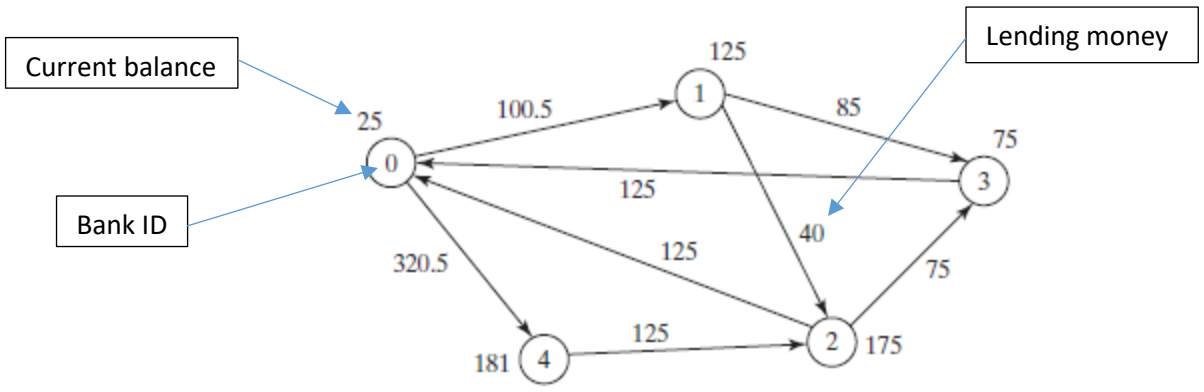
End Sample Run - 2

Task 2:

Write a simulation of Banks. Banks trade to other bank by lending and borrowing the money to each other. In tough economic times, if a bank goes bankrupt, it may not be able to pay back the loan.

A bank's total assets are its current balance plus its loans to other banks.

- The diagram below shows five different banks (0 to 4).
- The banks' current balances are 25, 125, 175, 75, and 181 million dollars, respectively.
- The directed edge from node 1 to node 2 indicates that bank 1 lends 40 million dollars to bank 2.



Banks lend money to each other

- If a bank's total assets are under a certain limit, the bank is unsafe.
- The money it borrowed cannot be returned to the lender, and the lender cannot count the loan in its total assets.
- Consequently, the lender may also be unsafe, if its total assets are under the limit.

Write a program to find all the unsafe banks. Your program reads the input as follows.

1. It first reads two integers **n** and **limit**,
    a. **n** indicates the number of banks
    b. **limit** is the minimum total assets for keeping a bank safe from the user.
2. It then reads **n** lines that describe the information for **n** banks with IDs from **0** to **n-1**.

3. The first number in the line is the bank's balance, the second number indicates the number of banks that borrowed money from the bank, and the rest are pairs of two numbers.

4. Each pair describes a borrower. The first number in the pair is the borrower's ID and the second is the amount borrowed.

For example, the input for the five banks in above picture is as follows (**note that the limit is 201**):

Number of banks: 5
Minimum asset limit: 201
For Bank # 0
    Balance: 25
    Number of banks Loaned: 2
        Bank ID who gets the loan: 1
        Loaned Amount: 100.5
        Bank ID who gets the loan: 4
        Loaned Amount: 320.5

For Bank # 1
Balance: 125
Number of banks Loaned: 2
Bank ID who gets the loan: 2
Loaned Amount: 40
Bank ID who gets the loan: 3
Loaned Amount: 85
For Bank # 2
Balance: 175
Number of banks Loaned: 2
Bank ID who gets the loan: 0
Loaned Amount: 125
Bank ID who gets the loan: 3
Loaned Amount: 75
For Bank # 3
Balance: 75
Number of banks Loaned: 1
Bank ID who gets the loan: 0
Loaned Amount: 125
For Bank # 4
Balance: 181
Number of banks Loaned: 1
Bank ID who gets the loan: 2
Loaned Amount: 125

The total assets of bank 3 are (75 + 125), which is under 201, so bank 3 is unsafe. After bank 3 becomes unsafe, the total assets of bank 1 fall below (125 + 40). Thus, bank 1 is also unsafe.

Note: Program should take inputs from the user like Number of banks, Minimum asset limit and then all other inputs

The output of the program should be

Unsafe banks are Bank 3 and Bank 1
Bank 3 got bankrupted and it got the loan from Bank 1, because Bank 3 is unsafe due to under the limit Bank 1 is also unsafe due to lower limit.

Workshop Header

/***
Workshop #
Course:*<subject type> - Semester*
Last Name:*<student last name>*
First Name:*<student first name>*
ID:*<student ID>*
Section:*<section name>*
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:*<submission date>*
***/

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Document all the classes properly
- Do Not have any debug/ useless code and/ or files in the assignment
- Do not have everything in the *main method*.
- Have a separate TestClass with the main method in it.
- Check your inputs if the user is not entering garbage inputs.
- Use exceptional handling or other methods to let the user know if the inputs are incorrect.

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- You are supposed to create video/ record voice/ detailed document of your running solution. **(50%)**
  - Screen Video captured file should state your last name and id, like Ali_123456.mp4 (or whatever the extension of the file is)
  - Record voice clip should also include a separate word file with the screen shots of your program's output, state your last name and id, like Ali_123456.mp3 (or whatever the extension of the file is)

  - Detailed document should include screen shots of your output, have your name and id on the top of the file and save the file with your last name and id, like Ali_123456.docx (or whatever the extension of the file is)
- A word/ text file which will reflect on learning of your concepts in this workshop. Also include the instructions on how to run your code. **(30%)**
  - Should state your Full name and Id on the top of the file and save the file with your last name and id, like Ali_123456.txt
- Submission of working code. **(20%)**
  - Make sure your follow the "**Code Submission Criteria"** mentioned above.
  - You should zip your whole working project to a file named after your Last Name followed by the first 3 digits of your student ID. For example, **Ali123.**zip.
- Your marks will be deducted according to what is missing from the above-mentioned submission details.
- Late submissions would result in additional 10% penalties for each day or part of it.
- Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any source.