COMP251 - Assignment 3
Deadline: Dec 9th, 2021

Goal:

Using priority queues to implement a loan approval system for a financial institute that funds startups.

Problem:

Our financial institute receives many loan applications and rejects them only if the applicant doesn't qualify (so generous!) otherwise the applications will be added to a list (called active list) and will be approved if the budget is available. The loan approval system allows the following commands:

- **Load the applications**: Reads the information of the applications from a file and store them as active applications if they qualify (if an application does not meet the requirements it will be added to the rejected application list)
- **Set the budget**: Update the current budget (the input for this command is some amount of budget that will be added to the current budget of the institute. At the beginning the current budget should be initialized as 0).
- **Make a decision**: given the current budget, make a decision about active applications (approve as many applications as possible: i.e remove them from the active application list and add them to the approved list).
- **Print**: print the list of active applications (no decision made so far), approved applications and rejected applications in three separate log files (approved.txt, rejected.txt, active.txt).
- **Update an application**: Applicants can update their application later by providing more documents. This may only affect active applications. For this command you can assume the information is given in a file (exactly the same format as input applications for command *Load*)

Application:

New applications are stored in a file which is the input of *load* or *update* command. Each application contains different fields (separated by tab '\t') and will be stored as a separate line in the input file. The following are the application's fields:

- Applicant's full name (can be stored as a string)
- Loan amount that is requested
- Years of relevant education
- Years of relevant experience
- Estimated annual profit: it is a list of estimated profit for the financial institute from the customers' payback in each year (for up to 30 years). Note that different applicants may differ in the length of estimated annual profit.

Approval process:

The financial institute will keep applications in three lists. Once the applications are read from the input file, they will be checked to see if they meet the requirements or not and based on that added to the active application list or rejected list. If the sum of education and experience years is smaller than 10, the application will be rejected. Otherwise the application will get some score and will be added to the active list. The following formula is used to calculate the score of active applications:

$$score: \sum_{i=0}^{n-1} \frac{estimatedAnnualProfit\ [i]}{(i+1)}$$

where *estimatedAnnualProfit*[i] is the estimation of annual profit for year i after approval (it is a part of the application form).

Note that in the case of update command, the score of the application (if in active list) will be calculated again and it will be updated in the active list (you do not need to search in the rejected list to update an application. if the application was rejected before it is like a new application).

Sample Input

The following is an example of an input file:

```
Maryam Siahbani      150000  10       3       0       0      24000   40000   50000   24000
John Smith    150000  6       3       0       0       25000   50000   50000   25000
Larry Page    150000000       8       25       0       150000000
Some Guy      120000  4       13      0       60000   90000   40000
Some Gal      120000  7       10       0       60000   90000   40000
```

Each line is one application and different fields of each application are separated by tab ("\t"). For example, in the first application (the first line), the applicant's full name is "Maryam Siahbani", the requested loan is 150000$, years of relevant education is 10, and years of relevant

experience is 3, and estimated annual profit is [0, 0, 24000, 40000, 50000, 24000]. So for this applicant the score will be calculated as 0*1/1 + 0* 1/2 + 24000 * 1/3 + 40000* 1/4 + 50000 * 1/5 + 24000* 1/6 = 32000.
A sample input file will be posted to Blackboard as an example, make sure you follow the input format.
Small hint on reading file: When you read each line of the file you can split it based on “\t” to get all the tokens as a list.

Note that the approval of the applications depends on their priority scores and the available budget. For example in the above example, assume that the institute budget is set to 300000, then the input file is given to the loan-approval application and then command “make decision” is requested and finally command print. The result in the `active.txt, approved.txt` and `rejected.txt` files will be as follows:

active.txt:

```
Larry Page   150000000    75000000
Maryam Siahbani     150000   32000
```

approved.txt:

```
Some Gal     120000
Some Guy     120000
```

rejected.txt

```
John Smith   150000
```

Note that although the priority score of “Larry Page” was higher than others it was not approved, due to lack of budget. The applications in the active list will be printed based on the priority score (sorted in decreasing order). The last field in each line of active.txt is the priority score.

Guideline:

You need to implement and submit at least 4 classes:

- `Applicant`: this class keeps the information for each applicant, name, education, experience,... . This class should also have a field called `score` (see the Approval process for the details). This class should implement interface `Comparable` (define method `compareTo()` which compares regarding the value of `score`).
- `PriorityQueue`: your priority queue should be a max heap (you did this in lab8). I posted a max heap on Blackboard along with this assignment. You can use that. Study all the methods in this class properly.

- `Loan`: This is the main class for loan approval. This class should keep three lists: active applications, approved applications and rejected applications. The approved and rejected lists can be a regular ADT List (you can use `ArrayList` in Java API). The active application list should be a priority queue (an object of `PriorityQueue` class). In class Loan you should have a method called `run()` which prints the options for the user and executes them.
- `test`: This class is just for testing. It will include method main, just create an object of class `Loan` and test it. I posted an example test file (You can use it as is - do not change it).

Requirements:

Make sure your final submission passes all the following requierments to get the full mark:

- Your code should run without error.
- Your loan application properly communicates with the user (print the possible commands, takes the commands and executes them).
- Your loan application has all the commands mentioned above (properly working).

Delivery

- Please record a short video in which you explain your work. Run through your code and show how it works. The design of your code and all the methods (just a brief description, algorithms and data structure you use). If you could not finish all parts of the assignment, mention them in your video.
- **Important**: post your video on youtube and set the **publish time to be three days after the deadline** of the corresponding assignment.
- Submit: Create a directory called src and copy all your **source files** (.java files) and a **readme** file (a simple .txt file which you should write the link to your video on youtube) in it, zip it and submit it to the Blackboard.
- **Important**: I'll randomly choose some students for **oral presentation**, and they will be notified by email. If you get selected, I'll schedule a time slot and **you should present your assignment to me and answer some questions.**

Important Remarks

It's very important that you make sure your program compiles without any problem (you may get 0 if your program doesn't compile)

For this project, you must work alone!

By alone rule: All code that you submit should be written by you alone, except for small snippets that solve tiny subproblems (of course you are allowed to use the source codes we discussed in class or I post on blackboard).

Do Not Possess or Share Code: Before you've submitted your final work for a project, you should never be in possession of solution code that you did not write. You will be equally culpable if you distribute such code to other students or future students of COMP251 (within reason). DO NOT GIVE ANYONE YOUR CODE – EVEN IF THEY ARE DESPERATELY ASKING. DO NOT POST SOLUTIONS TO PROJECTS ONLINE (on GitHub or anywhere else)! If you're not sure what you're doing is OK, please ask.

Permitted:

- Discussion of approaches for solving a problem.
- Giving away or receiving significant conceptual ideas towards a problem solution. Such help should be cited as comments in your code. For the sake of other's learning experience, we ask that you try not to give away anything juicy, and instead try to lead people to such solutions.
- Discussion of specific syntax issues and bugs in your code.
- Using small snippets of code that you find online for solving tiny problems (e.g. googling "uppercase string java" may lead you to some sample code that you copy and paste into your solution). Such usages should be cited as comments in your hw, lab, and especially project code!

Absolutely Forbidden:

- Possessing another student's project code in any form before a final deadline, be it electronic or on paper. This includes the situation where you're trying to help someone debug. Distributing such code is equally forbidden.
- Possessing project solution code that you did not write yourself (from online (e.g. GitHub), staff solution code found somewhere on a server it should not have been, etc.) before a final deadline. Distributing such code is equally forbidden.