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)

  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.

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: 𝑖=0 𝑛−1 ∑ 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).

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. 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).