CS 284: Homework Assignment 4 Optional, extra credit
Due: November 19, 11:55pm

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. If some code was shown in class, it can be used, but it must be obtained from moodle, the instructor or the TA.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

Assignment. Implement a **Priority Queue** using a **MaxHeap** associated with the following UML diagrams. The purpose of this data structure is to organize credit card offers you have received. Specifically, the inputs for each credit card offer will be: (i) annual membership fee, (ii) cash back rate and (iii) annual percentage rate (APR). The cash back rate is the fraction of your monthly expenses that is returned to you. The APR is an annual interest rate on the balance you owe to the credit card provider. Interest is charged to you if do not pay your balance in full at the end of each month. For this assignment you will assume that you may carry a balance of 10% of your annual expenses that will be charged interest at the APR rate.

Based on these and your expected monthly charges, you will compute a score for each credit card offer as shown below. The priority queue will help you insert the offers and then retrieve the one with the highest score at any time.

Add your own classes, data members and methods to meet the following requirements. The code must implement the following UML diagrams precisely. Deviations will be penalized. Your grade will be 0 if your code does not compile.

1. Your class must be named MyPriorityQueue and should contain a MaxHeap implemented by you.
2. Each node of the MaxHeap will be of user-defined type, named creditOffer, that has three data members: membership fee in dollars per year, cash back rate (CBR) which is typically a few percentage points, and the APR as described above.
3. The score of a creditOffer should be $score = cashBack - membershipFee - APRamount$.
4. MyPriorityQueue should provide a constructor that reads in an array of creditOffer nodes and organizes them into a MaxHeap.

5. MyPriorityQueue is essentially the interface with the user and its methods delegate the work to the corresponding methods of MaxHeap.
6. The user must be able to retrieve the creditOffer with the maximum score at all times and also insert new offers at any time, while the heap property is preserved.
7. The main() method should open a text file named "offers.txt" and read credit offers that have been stored as three doubles on each line of the file. See the sample code on the last page for an example of how to read from such a file.
8. MyPriorityQueue should accept a parameter for you **expected monthly charges** to this card which will determine the scores.

creditOffer
+ membershipFee : double + CBR : double + APR: double
+ creditOffer (in fee : double, in cbr : double, in apr : double) // Creates a node storing the input data

MyMaxHeap
- theHeap : creditOffer [] - size : int - capacity : int

+ MyMaxHeap(in monthlyExpenses : double) // Creates an empty heap with capacity=5
+ MyMaxHeap(in arr: creditOffer [], int n: int, in monthlyExpenses : double)
// Creates a heap by inserting the elements of the input array arr
+ size() : int
+ add(in job : creditOffer) : boolean // Adds element and reheaps
+ removeMax() : creditOffer // Returns the node with the highest score and reheaps
+ showList() : void // Prints all the elements in the order they are stored
- swap(in i: int, in j: int) : void // swaps nodes with indices i and j
- compare(in i: int, in j: int) : int // compares nodes with indices i and j

MyPriorityQueue
- theQueue: MyMaxHeap

+ MyPriorityQueue(in monthlyExpenses : double) // creates an empty priority queue
+ MyPriorityQueue(in arr : creditOffer [], in n: int, in monthlyExpenses : double)
//creates a priority queue out of an array of n creditOffer nodes
+ offer(in co: creditOffer) : boolean //adds a new element to the queue and returns true
+ remove() : creditOffer // removes the first node in the queue and returns it
// or returns null if the queue is empty
+ peek(): creditOffer //returns the item in the front of the queue without removing it
// or null if the queue is empty
+ isEmpty(): boolean // returns true if the queue is empty
+ showList() : void // Prints all the elements in the queue in the order they are stored

Skeleton code for file reader:

```
public static void main(String[] args) {
  // declare local variables here
  double number;

  try{
     BufferedReader in= new BufferedReader(new
          FileReader(new File("offers.txt")));

     for(String inputLine= in.readLine();
          inputLine!=null; inputLine=in.readLine())
     {
        // split string and only process first number read
        String[] splits = inputLine.split(" ");
        System.out.println("Read: "+ inputLine);

        if(splits[0].length()!=0){
           for(int j=0;j<splits.length; j++){
               number = Double.parseDouble(splits[j]);
               System.out.println("Next int: "+ number);
           }
        }
     }
     in.close();
  }
  catch(IOException e){
     System.out.println("File I/O error");
     System.exit(1);
  }
}
```