CS 373 SPRING 2016: HW 1 ORACLE QUEUES

Assigned: 3/30/2016, **Due:** 4/6/2016

1 Introduction

The purpose of this assignment is to get you programming as soon as possible and to try out the array queue structure. The application is a set of oracles (oracle_0, oracle_1, ... oracle_{n-1}) whose job is to answer questions. These oracles are not very bright; each one of them has just one answer (answer[0], answer[1], ... answer[n-1]). The questions and answers are both read from files. The questions are then each put into queues. There are n queues, one for each oracle, and a random number generator determines which of the n queues each question goes into. Once all the questions are enqeueued, the oracles start removing questions from their own queues and answering in what is called *round robin* fashion. This means that oracle_0 answers its first question, then oracle_1 answers its first question, until each has answered (or not answered if its queue is empty) its first question. Then they go on to the next question in each queue. This continues till all queues are empty.

2 Problem Definition/Examples

This homework assignment is an application of array queues (also known as rotating queues) and is designed to help familiarize you with basic data structures. The assignment revolves around the list of questions read in from a file that need to be assigned, at random, to an oracle. After the questions are assigned, the oracles should each take turns answering questions as described above.

Each of these oracles will answer each question the same way, that is each oracle has an answer that it will provide to any question provided to it. Your program should output the oracle's response and the original question on a single line.

For example: if the question "What is your quest?" is submitted to the oracle whose response is "To seek the holy grail", the program should output

```
What is your quest?:  To seek the holy grail
```

Your program should successfully submit all questions to an oracle and print their answers in order.

2.1 Array Queue

To accomplish this task, we ask that you use array queues, one per oracle. A queue is a simple data structure that allows elements to be enqueued and dequeued in a way that preserves the order in which they were input. For whatever order the elements are enqueued, that will be the order that the elements are dequeued. An array queue is a queue that stores its data

in an array (rather than a linked list, which is also common). This assignment requires your queue to utilize an array.

This is accomplished by keeping track of the front and back of the queue in the array. The front of the queue is the next element to be dequeued and the back of the queue is the last element in the queue, after which the next new element will be enqueued. This property means that when the queue is empty, front will have passed back in the array (circularly). It is also important to note, when using an array queue, **do not move the elements in the array**. Enqueue and dequeue operations should be done by advancing the corresponding front/back pointers. Because these pointers may overlap and move from the end of the array to the beginning, array queues are also known as rotating queues.

Array queues have an additional important difference from the traditional linked-list queue. Array queues can become completely full if the array becomes full and no elements are dequeued. Make sure that your code can recognize when the queue is full, especially when enqueueing additional elements. There is no need in this assignment to create an array queue which increases the size of its array, a simple error message when adding to a full queue is sufficient. Recognizing when the queue is empty is equally important, especially when dequeuing. There are different approaches to check if the queue is full or empty. Using a size variable is the easiest way to recognize full and empty queues.

3 Given Material

Three java files and two text files are provided as part of the assignment. They are commented for the Eclipse IDE to explain the behavior of the provided code.

3.1 Utility.java

The Utility file is provided to deal with file reading and random number generation because they are not part of the course. This java code should not be modified.

3.1.1 init

init initializes the Utility and sets up the file reader and random number generator. This is sufficient initialization and calling a constructor on Utility is unnecessary.

3.1.2 readQuestions/readAnswers

These functions will return the files "questions.txt" and "answers.txt" as arrays of strings.

3.1.3 random

The random function will return a random number between zero and the parameter number. This will be useful for assigning questions to oracles. You will call it to find out to which oracle a given question is assigned. (oNum = Utility.random(numOracles))

3.2 Executor.java

The Executor file is the main program that you will execute to run the code. It will involve all of your code that is not in ArrayQueue. As provided, Executor only initializes Utility and reads the files into string arrays. There are comments to tell you what to do.

3.3 ArrayQueue.java

The ArrayQueue file requires student work to operate. As provided, it is only the structure of the data structure. All functions will need to be written by the student and class variables will need to be added as necessary. Some functions currently have temporary return values, which can be removed when you write your code.

3.4 Input files (questions.txt and answers.txt)

These are the questions and answers. The number of oracles should be the number of answers in the text file. These files are read into string arrays by the Utility class, so you don't have to do it.

4 Expected Results

4.1 Executor.java

The Executor class will need to assign the questions to oracles using ArrayQueue and Utility.random. Then, it will need to go through the oracles in round-robin fashion and print the response to each question.

4.2 ArrayQueue.java

All functions of the array queue need to be implemented and tested by the student. The functions that should be implemented are already in the ArrayQueue.java class and their desired behavior is in comments above. This class **must** use the String array provided as storage for data.

4.3 Executor output

Questions should be assigned to a random oracle in the order they are placed in the text file. Once each question is assigned to an oracle, the oracles should answer one question at a time in the order their answers appear in the text file. If an oracle does not have any questions in its queue when it is its turn to answer, nothing should be printed.

5 Grading

- Program runs and prints results: 10 points

- All queue functions operate correctly: 10 points
- Readability (code and comments): 5 points