Week 13 – Due April 29

You will build both stacks and queues. You will use an array as your storage container for a set of `String`s, and your classes `myStack` and `myQueue` will be given a size limit when they are constructed.

What to build

Your tasks will be to construct two new classes `myStack` and `myQueue` with the following descriptions. You will create two files: `myStack.java` and `myQueue.java`.

`myStack`

Your class should contain the following attributes:

- `private String[] data` -- this will be used to hold all of the information in the stack
- `private int top` -- this will hold the index *of* the last element stored in `data`. (If `top == -1` there are no elements in the data structure.)

Your class should have the following constructors:

- `myStack(int size)` -- creates a new instance with `data.length == size`.
- `myStack()` -- created a new instance so `data.length == 10`.

Your class will have the following methods:

- `public boolean push(String s)` -- pushes an element into the stack at `top` and increments the pointer. If `top` is out of bound and the element cannot be inserted return false, otherwise return true.
- `public String pop()` – pops the top element in the data structure (i.e. the one at `top`) and updates the value of `top`. Returns `null` if no elements are in the stack.
- `public String peek()` — accesses the top element in the data structure (i.e. the one at `top`) but leaves it in place. Returns `null` if no elements are in the stack.

`myQueue`

Your class should contain the following attributes:

- `private String[] data` -- this will be used to hold all of the information in the stack.
- `private int head (a.k.a. front)` -- this will contain the index of the front element of the queue (`head = -1` when the queue is empty).
- `private int numElements` -- keeps track of how many elements are currently in the queue.

Your class should have the following constructors:

- `myQueue(int size)` -- creates a new instance with `data.length == size`.
- `myQueue()` -- created a new instance so `data.length == 10`.

You class will have the following methods:

- `public boolean enqueue(String s)` -- adds `s` to the end of the queue and increments `numElements`. The location to insert the element will be determined based on the position of `head` and the value `numElements` (you will need to use modulous, `%`). This method returns `true` if the element is added, and `false` if the element cannot be added because the queue is full.
- `public String dequeue()` -- dequeues the `head` element in the data structure (i.e. the one at `head / front`) and updates the value of `head and numElements`. Returns `null` if there is nothing to dequeue.
- `public String peek()`—accesses the `head` element in the data structure (i.e. the one at `head`) but leaves it in place. Returns `null` if no elements are in the stack.

Both `myStack` **and** `myQueue`

Both classes will contain the following methods, though the implementations will be slightly different.

- `public int getSize()` -- returns the number of elements in the data structure.
- `public boolean isEmpty()` -- returns `true` if no elements are present in the data structure, returns `false` otherwise.
- `public boolean isFull()` -- returns `true` if the number of elements in the data structure is equal to the size of `data`.

What to turn in

You will submit 3 files:

- `myStack.java`,
- `myQueue.java`, and
- `StackQueueTester.java`, which will contain at least 5 test cases for each of the new classes.

Grading

- `myStack.java` -- 45 pts + 5 extra points
  - constructors -- 10 pts.
  - `push` -- 10 pts.
  - `pop` -- 10 pts.
  - `peek` -- 5 pts.
  - `getSize`, `isEmpty`, `isfull` -- 5 pts. each
- `myQueue.java` -- 45 pts + 5 extra points
  - constructors -- 10 pts.
  - `enqueue` -- 10 pts.
  - `dequeue` -- 10 pts.
  - `peek` -- 5 pts.
  - `getSize`, `isEmpty`, `isfull` -- 5 pts. each
- `StackQueueTester.java` -- 10 pts (1 pt each for 10 test cases)

Due Date

You assignment should be submitted on GitHub by **Wednesday, 29 April 2020, before lab.**

Recommended Schedule

- Monday, 20 April
  - test cases
  - stack & queue constructors
- Wednesday, 22 April
  - rest of the methods
- Monday, 27 April
  - final testing
- Wednesday, 29 April
  - polished work ready to submit