Important

There are general homework guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 8**. This includes unused code, so don't submit extra files that don't compile.
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.

Stacks and Queues

You are to code a stack and a queue. Each will be backed by an array. You must order your array in the way that we specify. Failure to do so will result in lost points. The PowerPoints from the book (located under Resources in T-Square) are an excellent resource for understanding how the array-backed stack and queue should work. Note that these are just a guide though and there will be differences between how you are supposed to implement the stack and queue and how the book implements them. Refer to this PDF and the interfaces provided for the specifics on how to implement your stack and queue.

Enqueue and Dequeue

Enqueue and dequeue are the add and remove methods for queues. Enqueue should add an item to the end of a queue and dequeue should remove an item from the front of the queue. **You must null out the data in the array when dequeueing.**

Push and Pop

Push and pop are the add and remove methods for stacks. Push should add an item to the end of a stack and pop should remove an item from the end of the stack. **You must null out the data in the array when popping.**

Resizing

You should only resize the backing array for your queue or stack if it is completely full and you want to add another item.

You are to treat your backing array as a circular array for a queue.. This means that if you want to add an item to the end of the array, but the last spot in the array is filled, if there is space at the front of the array you should circle around to it.

The stack will use a standard linear array, as you do not need to go to the front.

Whenever you resize your backing array, you end with a backing array that is filled, starting at index 0, with the elements of the old array. The only gap immediately after resizing your backing array should be at the back of the array. When you resize your backing array, you should **double its size.**

Exceptions

When throwing exceptions, you must include a message by passing in a String as a parameter. For example:

```
throw new PDFReadException("Did not read PDF, will lose points.");
```

Style and Formatting

It is important that your code is not only functional but is also written clearly and with good style. We will be checking your code against a style checker that we are providing. It is located in resources along with instructions on how to use it. We will take off a point for every style error that occurs. If you feel like what you wrote is in accordance with good style but still sets off the style checker please email Jonathan Jemson (jonathanjemson@gatech.edu) with the subject header of "CheckStyle XML".

Javadocs

Javadoc any helper methods you create in a style similar to the Javadocs for the methods in the interface.

Forbidden Statements

You may not use these in your code at any time in CS 1332.

- break may only be used in switch-case statements
- continue
- package
- System.arrayCopy()
- clone()
- assert()
- Arrays class
- Array class
- Collections class
- Reflection APIs

Debug print statements are fine, but should not print anything when we run them. We expect clean runs - printing to the console when we're grading will result in a penalty. If you use these, we will take off points.
Arrays must be copied **manually using a loop of some sort. Do not** use built in array copy functions, or points will be deducted.

Provided

The following file(s) have been provided to you. There are several, but you will only edit two of them.

1. `QueueADT.java` This is the interface you will implement when designing your queue. All instructions for what the methods should do are in the javadocs. **Do not alter this file.**
2. `StackADT.java` This is the interface you will implement when designing your stack. All instructions for what the methods should do are in the javadocs. **Do not alter this file.**
3. `ArrayQueue.java` This class is where you will be implementing your queue. **Do not add any public methods to this file.**
4. `ArrayStack.java` This class is where you will be implementing your stack. **Do not add any public methods to this file.**
5. `QueueStackTestStudent.java` This is the test class that contains a set of tests covering the basic operations on the `QueueADT` and the `StackADT`. It is not intended to be exhaustive and does not guarantee any type of grade. **Write your own tests to ensure you cover all edge cases.**

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interfaces, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc.

1. `ArrayQueue.java`
2. `ArrayStack.java`

You may attach each file individually or submit them in a zip archive.