CSCI 230 Data Structures
Instructor: Dr. Sebastian van Delden
Email: vandeldensa@cofc.edu

ASSIGNMENT 2 – INDIVIDUAL WORK ONLY!!!

Implement the **Eight Piece Puzzle** Program that utilizes the **A* Searching Algorithm** that uses the **Manhattan Distance Heuristic**, and a **Priority Queue** that stores board states that are yet to be considered. You ***must*** follow the below details.

PROGRAM GUIDELINES THAT MUST BE FOLLOWED

The Main Class:

- Simply create a new eightPuzzle object

The EightPuzzle Class:

- Constructor:	- Passed a 1x9 array of ints representing a board state. (Assume that the initial state is 0 1 2 3 4 5 6 7 8) - Call *checkReachable* method to see if goal state is reachable - If it is reachable then call *solve* method, else prompt for another goal state.
- checkReachable:	-Takes the goal state, a 1D array of ints of size 9, and return true if it is reachable
- solve:	- Performs Eight Puzzle Algorithm which uses the PriorityQueue and LinkedList data structures - Uses the getChildren method to generate a LinkedList containing the board state's children. - When goal is found, print the path to the goal using the printPath method.
- printPath:	- MUST be recursive. Prints the path to the goal.
- getChildren:	- Takes a 1D array board state and returns a linked list containing its children states.
- Manhattan:	- Takes two 1x9 arrays of ints (board states) and returns the Manhattan distance between them.

The BoardState Class:	- Implements the Comparable interface - must include the compareTo() method. - Should have attributes for its board state, g, h, and reference to parent BoardState. - Overwrites Object's equals() method which returns true if both BoardStates have the same state. - Overwrites Object's toString() method which should return a String containing the board state.
The PriorityQueue Class :	- Should extend the Queue class that is provided to you with this assignment. - Do not modify the linkedlist/stack/queue/etc classes that are provided to you. - Need to create a priorityEnqueue and find method. Use generics to indicate that the Object to be enqueued or found implements the Comparable interface:

```
public void priorityEnqueue(Comparable<Object> item);
public Comparable<Object> find(Comparable<Object> item);
```

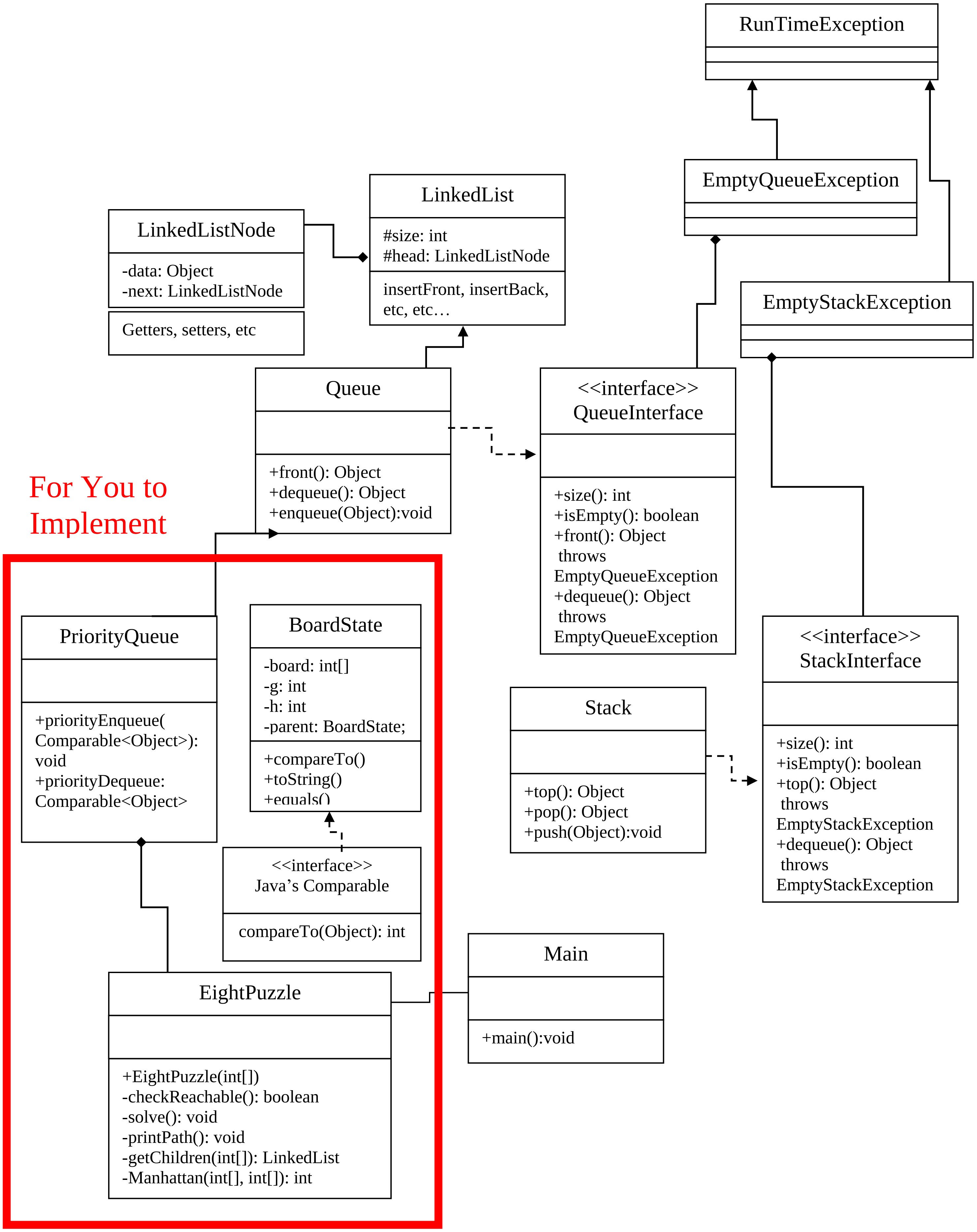
OTHER DETAILS THAT MUST BE FOLLOWED

- A board state will be stored as a one dimensional array of nine integers: 0 through 9. Zero will represent the blank space on the board. At no point in the program should you use a 3 by 3 array of integers to store a board state. Doing so will result in the loss of 10 points for each usage.
- You can assume the start state will always be: 0 1 2 3 4 5 6 7 8 i.e.

```
0 1 2
3 4 5
6 7 8
```

SUBMISSION DETAILS

Upload entire Eclipse project folder in a ZIP file to OAKS by the due date. ***THIS ASSIGNMENT IS INDIVIDUAL WORK***. A score of zero will be assigned to all programs which contained portions of code that have been duplicated.


RunTimeException
EmptyQueueException
LinkedList
#size: int
#head: LinkedListNode
insertFront, insertBack, etc…
LinkedListNode
-data: Object
-next: LinkedListNode
Getters, setters, etc
EmptyStackException
Queue
+front(): Object
+dequeue(): Object
+enqueue(Object):void
<<interface>> QueueInterface
+size(): int
+isEmpty(): boolean
+front(): Object throws EmptyQueueException
+dequeue(): Object throws EmptyQueueException
For You to Implement
PriorityQueue
+priorityEnqueue(Comparable<Object>): void
+priorityDequeue: Comparable<Object>
BoardState
-board: int[]
-g: int
-h: int
-parent: BoardState;
+compareTo()
+toString()
+equals()
<<interface>> StackInterface
+size(): int
+isEmpty(): boolean
+top(): Object throws EmptyStackException
+dequeue(): Object throws EmptyStackException
Stack
+top(): Object
+pop(): Object
+push(Object):void
<<interface>> Java's Comparable
compareTo(Object): int
Main
+main():void
EightPuzzle
+EightPuzzle(int[])
-checkReachable(): boolean
-solve(): void
-printPath(): void
-getChildren(int[]): LinkedList
-Manhattan(int[], int[]): int