Assignment 5

Pseudocode

```
public boolean add(E itemToAdd)
  Check to see if queue array is full
    if it is:
      create new array with bigger size
      populate array with our existing array
      point the variable towards the existing array (make sure to set the Head variable and tai
l variable
      call a garbage collection
  Add the item to the end and adjust the tail

public E remove()
  Set a temp variable to the head element
  nullify the element at the head
  adjust the head
```

SimpleQueue.java

```java
1  import java.util.*;
2  
3  public class SimpleQueue<E> implements Queue<E> {
4  
5    /* Table of contents for SimpleQueue:
6     * Constructor  //Ln. 24
7     * add()        //Ln. 32
8     * remove()     //Ln. 41
9     * poll()       //Ln. 51
10    * clear()      //Ln. 57
11    * size()       //Ln. 64
12    * toArray()    //Ln. 70
13    * peek()       //Ln. 78
14    * element()    //Ln. 83
15    * isEmpty()    //Ln. 89
16    * expandList() //Ln. 94
17    * UnusedMethod //Ln. 105
18    */
19 
20   //Variable declarations
21   private int head, tail, size;
22   private E[] queue;
23 
24   //This is the constructor for SimpleQueue
25   public SimpleQueue() {
26     queue= (E[])new Object[5];
27     size = 5;
28     head = 0;
29     tail = 0;
30   } //End constructor
31 
32   /* This will add the item to the queue
33      Will return true when the item has been sucessfully added. */
34   public boolean add(E itemToAdd) {
35     queue[tail] = itemToAdd;
36     tail = ++tail % size;
37     if(tail == head) expandList();
38     return true;
39   } //End add()
40 
41   /* Pops the item off the queue and adjusts the head accordingly.
42      Returns exception if empty */
43   public E remove() {
    if(isEmpty()) throw new IllegalArgumentException("Empty List");
    E temp = queue[head];
    queue[head] = null;
    head = ++head % size;
    return temp;
  } //End remove()

  //Pops the item off the queue and returns null if empty
  public E poll() {
    if(isEmpty()) return null;
    return remove();
  } //End poll()

  //This will clear the array
  public void clear() {
    queue = (E[]) new Object[size];
    head = 0;
    tail = 0;
  } //End clear()

  //This will return the array size as an int
  public int size() {
    if(isEmpty()) return 0;
    return (tail > head) ? (tail - head) : (size - head + tail);
  } //End size()

  //This will return the array
  public E[] toArray() {
    if(isEmpty()) return (E[])new Object[size()];
    E[] temp = (E[])new Object[size()];
    for(int i=0; i < size(); i++) temp[i] = queue[(head + i) % size];
    return temp;
  } //End toArray()

  //Return but not remove the head. Returns null if empty
  public E peek() {
    return queue[head];
  } //End peek()

  //Returns but not remove the head. Throws exception if empty
  public E element() {
    if(isEmpty()) throw new NoSuchElementException("Empty queue");
    return peek();
  } //End element()

  //Returns true if the array is empty, false if not
  public boolean isEmpty() {
    return queue[head] == null;
  } //End isEmpty()

  //Expands the list.
  private void expandList() {
    E[] temp = (E[]) new Object[size*2];
    for(int i=0; i < size(); i++) temp[i] = queue[(head + i) % size()];
    tail = size();
    head = 0;
    size *= 2;
    queue = temp;
    System.gc(); //Reccomend a g.collection
  } //End expandList()

  /* This is going to be used for testing.
   * Will add itemToAdd to list defined by times
   * Will be commented out on code submission
   */
  public void stressTest(int times, E itemToAdd) {
    if(times == 0) return;
    add(itemToAdd);
```

```java
      stressTest(times - 1, itemToAdd); // :^)
    } //End stressTest()

    //These are the unused methods
    public boolean remove(Object arg0)
      {throw new UnsupportedOperationException();}
    public boolean removeAll(Collection<?> arg0)
      {throw new UnsupportedOperationException();}
    public Iterator<E> iterator()
      {throw new UnsupportedOperationException();}
    public boolean addAll(Collection<? extends E> arg0)
      {throw new UnsupportedOperationException();}
    public boolean containsAll(Collection<?> arg0)
      {throw new UnsupportedOperationException();}
    public boolean retainAll(Collection<?> arg0)
      {throw new UnsupportedOperationException();}
    public boolean offer(E arg0)
      {throw new UnsupportedOperationException();}
    public <T> T[] toArray(T[] arg0)
      {throw new UnsupportedOperationException();}
    public boolean contains(Object o)
      {throw new UnsupportedOperationException();}
    //End unused methods

  } //End SimpleQueue()
```

QueueTester.java

```java
import java.util.*;

public class QueueTester {
  public static void main(String[] args) {

    //Step 1: Constructing various queues
    SimpleQueue intQueue = new SimpleQueue<Integer>();
    SimpleQueue stringQueue = new SimpleQueue<String>();

    //Step 2: trying functions on empty queues
    System.out.println("====EMPTY QUEUE TESTS====");
    System.out.println("Queue should be empty: " + intQueue.isEmpty());
    System.out.println("Polling empty queue: " + intQueue.poll());
    System.out.println("Peeking at empty array: " + intQueue.peek());
    System.out.println("Size of current array: " + intQueue.size());
    System.out.println("Clearing blank array...");
    intQueue.clear();
    System.out.println("Returning blank array " +
                      Arrays.toString(intQueue.toArray()));
    try {
      System.out.print("Trying remove: ");
      intQueue.remove();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    try {
      System.out.print("Trying element: ");
      intQueue.element();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    System.out.println("");

    //Step 3: adding/removing elements to the queue
    System.out.println("====ADDING ELEMENTS====");
    System.out.println("Adding some elements to intQueue");
    intQueue.add(1);
```

```
        System.out.println("Trying isEmpty(): " + intQueue.isEmpty());
        intQueue.add(2);
        intQueue.add(3);
        System.out.println("Size: " + intQueue.size());
        System.out.println("Queue is now: " +
                           Arrays.toString(intQueue.toArray()));
        System.out.println("Lets move the array around...");
        System.out.println("Removed: " + intQueue.remove());
        System.out.println("Polled: " + intQueue.poll());
        System.out.println("Adding...");
        intQueue.add(4);
        intQueue.add(5);
        intQueue.add(6);
        System.out.println("Size: " + intQueue.size());
        System.out.println("Queue is now: " +
                           Arrays.toString(intQueue.toArray()));
        System.out.println("Peeking: " + intQueue.peek());
        System.out.println("Element: " + intQueue.element());
        System.out.println("Adding a ton of numbers to intQueue");
        intQueue.stressTest(10, 1337);
        System.out.println("Queue is now: " +
                           Arrays.toString(intQueue.toArray()));
        System.out.println("Clearing the queue: ");
        intQueue.clear();
        System.out.println("Queue is now: " +
                           Arrays.toString(intQueue.toArray()));
        System.out.println("");

        //Step 4: String tests
        System.out.println("====String Queue test====");
        System.out.println("Adding string null");
        stringQueue.add("null");
        System.out.println("Queue: " +
                           Arrays.toString(stringQueue.toArray()));
        System.out.println("isEmpty(): " + stringQueue.isEmpty());
        System.out.println("Adding some strings...");
        stringQueue.add("Space");
        stringQueue.add("Cowboy");
        stringQueue.add("Cyber Decker");
        System.out.println("Adding strange \0 string");
        stringQueue.add("\0");
        System.out.println("Queue: " +
                           Arrays.toString(stringQueue.toArray()));
        System.out.println("Removing null");
        stringQueue.remove();
        System.out.println("Queue: " +
                           Arrays.toString(stringQueue.toArray()));
        System.out.println("Clearing");
        stringQueue.clear();
        System.out.println("Queue: " +
                           Arrays.toString(stringQueue.toArray()));
    } //End main()
} //End QueueTester()
```

Output

```
====EMPTY QUEUE TESTS====
Queue should be empty: true
Polling empty queue: null
Peeking at empty array: null
Size of current array: 0
Clearing blank array...
Returning blank array []
Trying remove: Empty List
Trying element: Empty queue

====ADDING ELEMENTS====
Adding some elements to intQueue
Trying isEmpty(): false
Size: 3
Queue is now: [1, 2, 3]
Lets move the array around...
Removed: 1
Polled: 2
Adding...
Size: 4
Queue is now: [3, 4, 5, 6]
Peeking: 3
Element: 3
Adding a ton of numbers to intQueue
Queue is now: [3, 4, 5, 6, 1337, 1337, 1337, 1337, 1337, 1337, 1337, 1337, 1337, 1337]
Clearing the queue:
Queue is now: []

====String Queue test====
Adding string "null"
Queue: [null]
isEmpty(): false
Adding some strings...
Adding strange "\0" string
Queue: [null, Space, Cowboy, Cyber Decker, ]
Removing "null"
Queue: [Space, Cowboy, Cyber Decker, ]
Clearing
Queue: []
```