```java
/*--- HW1.java ---*/
import java.util.*;
/**
 * Class HW1 that only has a main.
 * @author Thompson, Joshua 206360
 */
public class HW1
{
  /**
   * main method that creates a Tree object of type integer. Adds 10 integers to th
at tree and prints them out
   * @param args Command line arguments are ignored
   */
  public static void main(String[] args) {
    Random rand = new Random(System.currentTimeMillis());

    TreeSet<Integer> tr = new TreeSet<Integer>();

    for(int i = 0; i < 10; i++)
      tr.add(rand.nextInt(1000));

    Iterator<Integer> i = tr.iterator();

    while(i.hasNext())
      System.out.println(i.next());
  }
}

/*--- Queue.java ---*/
/**
 * Class Queue used to create Queues of Strings
 * @author Thompson, Joshua - 206360
 */
public class Queue {
  /**
   * enqueue method that adds a new node to the end of the queue.
   * @param s Input string that the user would like to enqueue
   */
  public synchronized void enqueue(String s) {
    if (head == null) {
      head = tail = new Node(s, null);
    } else {
      tail.next = new Node(s, null);
      tail      = tail.next;
    }
  }

  /**
   * Dequeue method that returns the first item in the queue and takes it out of th
e queue.
   * There is an additional condition where if the head is already null, then simpl
y return and empty string.
   * This is incase there are multiple threads working on the same object and deque
uing.
   * @return String from the dequeued node
   */
  public synchronized String dequeue() {
    Node t = head;
    if(head == null)
      return "";
    head = head.next;

    if (head == null) {
      tail = null;
    }
    return t.data;
  }

  /**
   * If the queue is empty return true
   * @return Returns whether the queue is empty or not
   */
  public synchronized boolean empty() {
    return head == null;
  }

  /**
   * Class node that is the backbone of the queue. Each node
   * is used to make up the queue
   */
  private class Node {
    public String data;
    public Node   next;

    /**
     * Constructor for the node
     * @param d String for the data portion of the node
     * @param n A node object to point to. Next node in the queue
     */
    public Node(String d,
                Node   n) {
      data = d;
      next = n;
    }
  }
  private Node head = null, tail = null;
}
```