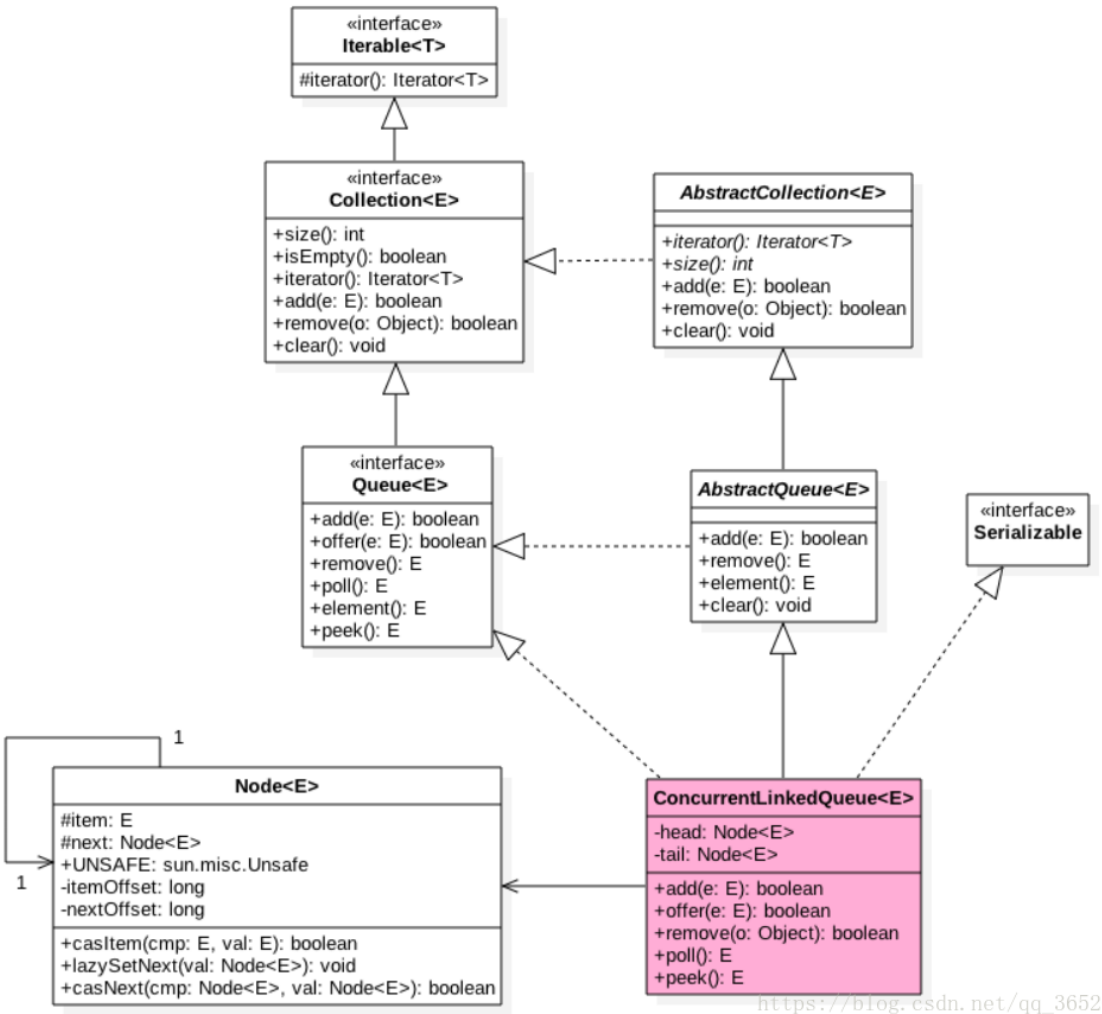
ConcurrenLinkedQueue:

1.concurrentlinkedqueue的简单了解和用途

继承的是AbstractQueue接口，实现的是Queue接口。

使用的是单向链表的形式来实现队列

```
 * @since 1.5
 * @author Doug Lea
 * @param <E> the type of elements held in this collection
 */
public class ConcurrentLinkedQueue<E> extends AbstractQueue<E>
        implements Queue<E>, java.io.Serializable {
    private static final long serialVersionUID = 196745693267521676L;
```



定义一个Node方法体，使用volatile进行修饰，这样就可以修改到主内存，全部线程可见。

```java
private static class Node<E> {
    volatile E item;
    volatile Node<E> next;

    /**
     * Constructs a new node.  Uses relaxed write because item can
     * only be seen after publication via casNext.
     */
    Node(E item) { UNSAFE.putObject( o: this, itemOffset, item); }

    boolean casItem(E cmp, E val) { return UNSAFE.compareAndSwapObject( o: this, itemOffset, cmp, val); }

    void lazySetNext(Node<E> val) { UNSAFE.putOrderedObject( o: this, nextOffset, val); }

    boolean casNext(Node<E> cmp, Node<E> val) { return UNSAFE.compareAndSwapObject( o: this, nextOffset, cmp, val); }

    // Unsafe mechanics

    private static final sun.misc.Unsafe UNSAFE;
    private static final long itemOffset;
    private static final long nextOffset;

    static {
        try {
            UNSAFE = sun.misc.Unsafe.getUnsafe();
            Class<?> k = Node.class;
            itemOffset = UNSAFE.objectFieldOffset
                (k.getDeclaredField( name: "item"));
            nextOffset = UNSAFE.objectFieldOffset
                (k.getDeclaredField( name: "next"));
        } catch (Exception e) {
            throw new Error(e);
        }
    }
}
```

从上图可以看到，在Node构造函数中，使用了unsafe类，分别设置、比较并替换了item和next域的值。但是对next域使用了unsafe.putOrderedObject方法，导致对next域的修改并不会对其他线程立即可见。使用了unsafe类的cas算法保证了出入队列的一致性。CAS其实是一条CPU原子指令，其作用是让CPU先进行比较两个值是否相等，然后原子的更新某个位置的值，**即cas是基于硬件平台的，JVM封装了汇编调用，AtomicInteger类则使用了这些封装后的接口(atomic底层提供的是volatile和cas来实现的对数据的更改**，volatile实现了数据修改的主内存可见，禁止重排序，cas比较并替换实现了数据更新的原子性)。

而java的原子类是通过Unsafe类实现的，所以简单了解一下unsafe类。主要提供的是执行级别较低、且不安全操作的方法(java没有为我们提供unsafe类的对外API),例如：访问和管理系统内存资源。