```java
//*****************************************************************************
//
// File:    Cricket.java
// Package: ---
// Unit:    Class Cricket
//
//*****************************************************************************

/**
 * This class models a cricket that will chirp at time t + 2 if it hears a chirp
 * at time t. It inherits from vertex so that it can be connected to other
 * crickets through undirected edges.
 *
 * @author Jimi Ford (jhf3617)
 * @version 3-31-2015
 */
public class Cricket extends Vertex {

    private boolean[] chirp = new boolean[2];
    private boolean willChirp;
    private int currentTick = 0;
    private final CricketObserver observer;

    /**
     * Construct a cricket
     * @param n the unique integer identifier
     * @param o the cricket observer this cricket should report to
     */
    public Cricket(int n, CricketObserver o) {
        super(n);
        this.observer = o;
    }

    /**
     * force a cricket to chirp at the next time tick
     */
    public void forceChirp() {
        willChirp = chirp[0] = true;
    }

    /**
     * will chirp only if it is being forced to, or if it has heard a chirp
     * 2 time ticks ago
     */
    public void emitChirp() {
        if(willChirp) {
            willChirp = false;
            int n = super.degree();
            for(int i = 0; i < n; i++) {
                edges.get(i).other(this).hearChirp();
            }
            observer.reportChirp(currentTick, super.n);
        }
    }

    /**
     * hear another chirp from an adjacent cricket
     */
```

```java
    private void hearChirp() {
        chirp[1] = true;
    }

    /**
     * simulate time passing by letting the cricket know what time it is
     *
     * @param tick the current time tick for this cricket
     */
    public void timeTick(int tick) {
        currentTick = tick;
        willChirp = chirp[0];
        chirp[0] = chirp[1];
        chirp[1] = false;
    }

    /**
     * determine if a given cricket is directly connected to this cricket
     * @param other the given cricket to check
     * @return true if this cricket as a single edge that connects the two
     */
    public boolean directFlight(Cricket other) {
        boolean retval = false;
        if(equals(other)) return true;
        int e = super.degree();
        Cricket o;
        for(int i = 0; i < e && !retval; i++) {
            o = super.edges.get(i).other(this);
            retval = o.equals(other);
        }
        return retval;
    }

    /**
     * determine if another object is equal to this cricket
     * @param o the other object
     * @return true if the other object is equal to this cricket
     */
    public boolean equals(Object o) {
        if( !(o instanceof Cricket)) {
            return false;
        }
        if(o == this) {
            return true;
        }
        Cricket casted = (Cricket) o;

        return casted.n == this.n;
    }
}

```