

```
/**
 * Generates a binary coin (0 or 1) in strict alternating sequence. With every
 * flip of the coin, the coin changes from 0 to 1 or vice versa. The first call
 * returns a 0, then 1, then 0, and so on.
 *
 * @correspondence |history| is even <==> doneEvenFlips
 * @author paolo
 */
public class AlternatingCoin implements RandomWithParity {

    /**
     * True if and only if an even number of invocations to this AlternatingCoin
     * have been made.
     */
    private boolean doneEvenFlips;

    /**
     * Initially zero flips have been performed, hence
     * doneEvenFlips is true;
     *
     * @ensures doneEvenFlips = true;
     */
    public AlternatingCoin() {
        this.doneEvenFlips = true;
    }

    /**
     * Returns a 0 or a 1 (alternating) and ignoring upperBound parameter.
     *
     * @param upperBound
     *         an integer that is ignored
     * @ensures doneEvenFlips = !#doneEvenFlips
     * @return 0 iff #doneEvenFlips <br>
     *         1 iff !#doneEvenFlips
     * @see RandomWithParity#generateNumber(int)
     */
    public int generateNumber(int upperBound) {
        if (this.doneEvenFlips) {
            this.doneEvenFlips = false;
            return 0;
        }
        this.doneEvenFlips = true;
        return 1;
    }
}
```