```
import java.io.IOException;
import java.io.OutputStream;

/**
 * Methods to transform an output byte stream into a stream of
 * bits. Because they are written to an underlying byte stream, the
 * end of the stream is padded with 0's up to a multiple of 8 bits.
 */

public final class BitOutputStream {

    // Underlying byte stream to write to.
    private OutputStream output;

    // The accumulated bits for the current byte. Always an int in the
    // range 0 to 255.
    private int currentByte;

    // The number of accumulated bits in the current byte. Always
    // between 0 and 8, inclusive.
    private int numBitsInCurrentByte;


    // Creates a bit output stream based on the given byte output
    // stream.
    public BitOutputStream(OutputStream out) {
if (out == null)
    throw new NullPointerException("No output stream given");
output = out;
currentByte = 0;
numBitsInCurrentByte = 0;
    }


    // Writes a bit to the stream. The specified bit must be 0 or 1.
    public void writeBit(int b) throws IOException {
if (!(b == 0 || b == 1))
    throw new IllegalArgumentException("Argument must be 0 or 1");
currentByte = currentByte << 1 | b;
numBitsInCurrentByte++;
if (numBitsInCurrentByte == 8) {
    output.write(currentByte);
    output.write(currentByte);
    numBitsInCurrentByte = 0;
}

    }



    // Writes an int to the stream.
    public void writeInt(int b) throws IOException {
int bitsWritten = 0;
while (bitsWritten < 32){
    writeBit(b >>> (31-bitsWritten) & 1);
    bitsWritten++;
}
    }



    // Closes this stream and the underlying OutputStream. If called
    // when this bit stream is not at a byte boundary, then the
    // minimum number of "0" bits (between 0 and 7 of them) are
    // written as padding to reach the next byte boundary.
    public void close() throws IOException {
while (numBitsInCurrentByte != 0)
    writeBit(0);
output.close();
    }

}
```