```
import java.io.IOException;
import java.io.InputStream;

/**
 * Methods to transform an input byte stream into a stream of
 * bits.
 */

public class BitInputStream {

    // Underlying byte stream to read from.
    private InputStream input;

    // Buffer of up to 8 bits from the most recently read byte of the
    // underlying byte input stream. Is an int in the range 0 to 255
    // if bits are available, or is -1 if the end of stream is
    // reached.
    private int nextBits;

    // Always between 0 and 8, inclusive.
    private int numBitsRemaining;

    private boolean isEndOfStream;



    // Creates a bit input stream based on the given byte input stream.
    public BitInputStream(InputStream in) {
if (in == null)
    throw new NullPointerException("No input stream given");
input = in;
numBitsRemaining = 0;
isEndOfStream = false;
    }



    // Reads a bit from the stream. Returns 0 or 1 if a bit is
    // available, or -1 if the end of stream is reached. The end of
    // stream always occurs on a byte boundary.
    public int readBit() throws IOException {
if (isEndOfStream)
    return -1;
if (numBitsRemaining == 0) {
```

```
    nextBits = input.read();
    if (nextBits == -1) {

isEndOfStream = true;
return -1;
    }
    numBitsRemaining = 8;
}
numBitsRemaining--;
return (nextBits >>> numBitsRemaining) & 1;
    }



    // Reads an int from the stream. Throws IOException if 32 bits are
    // not available.
    public int readInt() throws IOException {
int output = 0;
int nextBit;
int bitsAdded = 0;
while(bitsAdded < 32){
    nextBit = readBit();
    if (nextBit == -1)
throw new IOException("Not enough bits while trying to read int");
    output = output << 1 | nextBit;
    bitsAdded++;
}
return output;
    }



    // Closes this stream and the underlying InputStream.
    public void close() throws IOException {
input.close();
    }

}
```