```
import java.io.FileInputStream;
import java.io.FileOutputStream;

// Test program to exercise BitInputStream and BitOutputStream. Run it
// on some small text file (containing at least 4 bytes) by
//
//   java Test infilename outfilename
//
// This should copy the infile to outfile, while adding the character
// '@' to the end of it (after any newline, if the file ends with
// this), and along the way write some multi-digit integer to the
// screen (read comments below to understand WHY this should be the
// behaviour).

public class Test {
    public static void main(String[] args) throws Exception {

// Open input and output byte streams to/from files.
FileInputStream inFile = new FileInputStream(args[0]);
FileOutputStream outFile = new FileOutputStream(args[1]);

// Wrap the new bit streams around the input/output streams.
BitInputStream in = new BitInputStream(inFile);
BitOutputStream out = new BitOutputStream(outFile);

// First read a full int (i.e., four bytes) from input file
// using the library method readInt().
int i = in.readInt();

// Print the value on screen (probably multi-digit integer, if
// run on a textfile, as four bytes representing some chars
// are ususally the bit pattern of some big integer).
System.out.println(i);

// Write the int again to output file (as same four bytes, so
// bytes should appear again in output file exactly they were
// in input file) using the library method writeInt().
out.writeInt(i);

// Now read last bits of input file. Do this bit by bit using
// the library method readBit(). At the same time write them
// again on the output file using the library method
```

```
// again on the output file using the library method
// writeBit(). Note the while-expression going through the
// file until no more bits are available (signaled by

// returning -1 instead of 0 or 1).
int bit;
while ( (bit = in.readBit()) != -1 )
    out.writeBit(bit);

// Write two bits MORE to output file. This will be padded by
// the library with six 0 bits when closing the output stream,
// hence give the character '@' (which in ASCII has pattern
// 01000000).
out.writeBit(0);
out.writeBit(1);

// Close the streams cleanly (automatically padding output
// streams with 0 bits until a multiple of bytes have been
// written, as described above).
in.close();
out.close();

    }
}
```