You use socket because your program needs to provide information to another program or vice versa.

So there must be a sequence on how to exchange the program.

The sequence is called protocol.

The agreement use on application is called an application protocol.

TCP/IP protocol transports bytes of user data without modyfying them.

Protocols made up from sequence of fields.

The application protocol specifies these fields very carefully.

Message can be sequence of bytes.

When you build a program to exchange via socket, you are doing either one of two things:

Designing/Writing the program on both sides

Or interpreting a protocol that someone else had already specified.

3.1 Encoding Information

Bytes of Information can be transmitted through a socket by writting them to an OutputStream or

encapsulating them in datagram packet.

The only data these operations can applied are bytes and array of bytes.

Therefore, we must do some conversion using Java built-in facilities.

3.1.1 Primitive integers

We can encode larger primitive integer types, but user and sender must agree on serveral things first.

One is the size of each integer being transmitted.

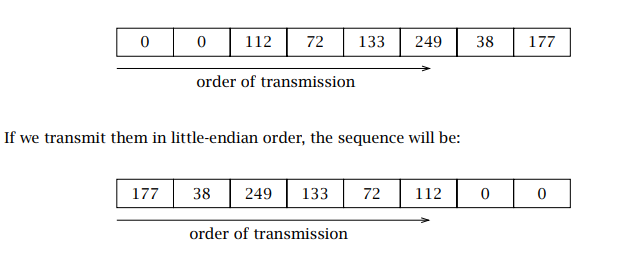
Ex: int value in Java is 32 bit-quantity, we can transfer any variables or constraints of type Int using 4-bytes

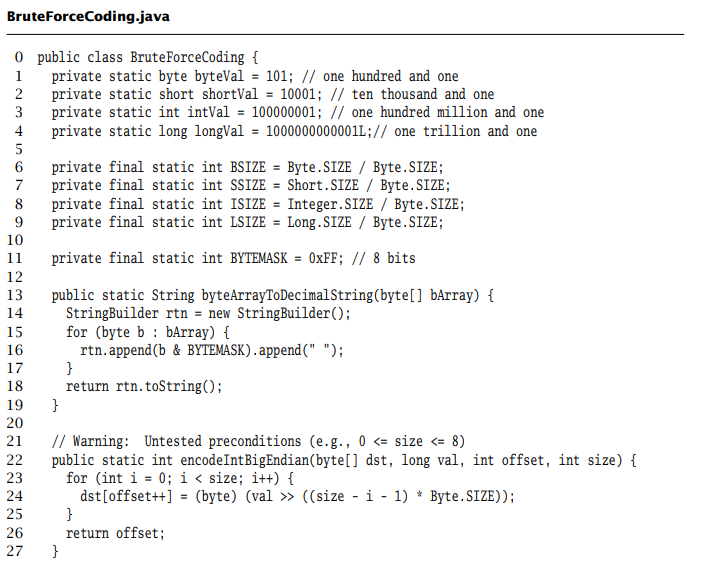
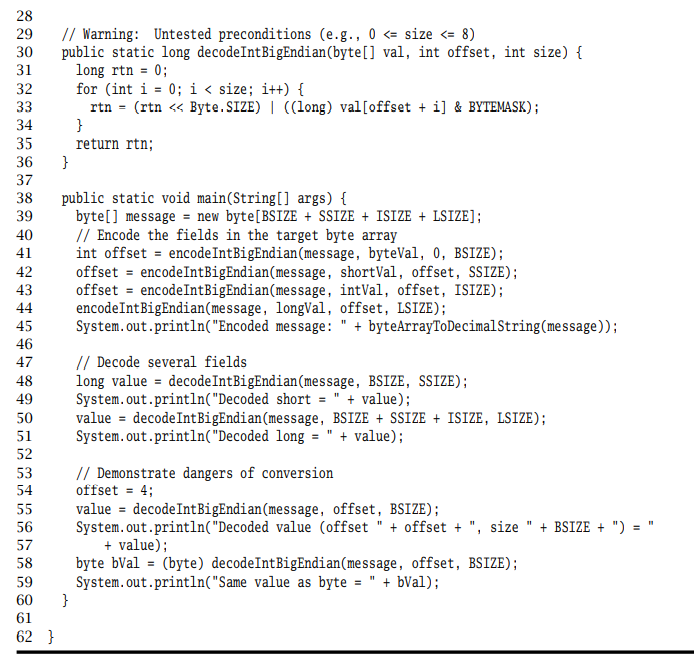
To encrypt at least 4 integer 4 integer values, we must use at least 15 bytes:

1 for byte, 2 for short, 4 for int and 8 for long.



There are two choices to send the byte in: at the right end order (little endian) or at the left end order (big endian).



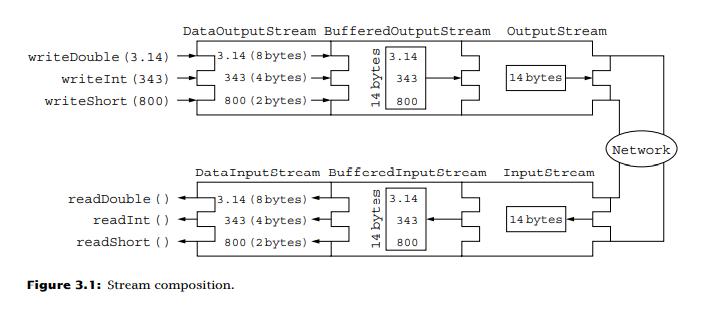
For multibyte integrity quantity, the sender must agree on which one to use, little endian or big endian.  
The last detail is whether the numbers transmitted will be signed or unsigned.  
  
The four primitves values all are signed and stored in two’s complement representation.  
  
Below is a Java program which can helps implement the correct values into the byte array of the message.  
  
  




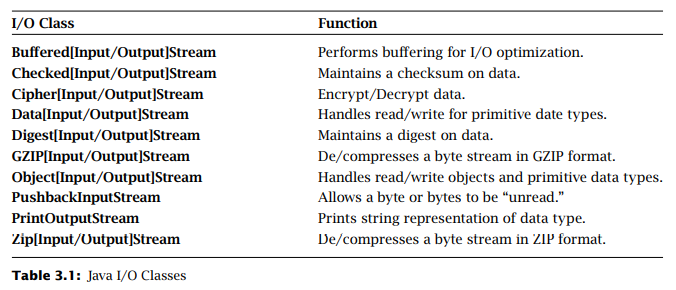
Data items to encode: Line 1 to 4  
Number of bytes in Java integer primitives: lines 6 – 9  
byteArraytoDecimalStrings: lines 13-19  
dsdencodeIntBigEndian: lines 22-27  
decodeIntBigEndian: lines 30-36  
  
Demonstrate Methods: lines 38-60  
(( Prepare array for receiving integers: lines 39  
 Encode Items: lines 40-44   
 Print contents of encoded array: line 45  
 Decode fields from encoded bytes array: line 47 -51  
 Conversion problems: lines 53 -59))  
  
  
Brute force like this requires a lot of work.  
It is not the recommended approach, because Java has built-in mechanism that are easier to use.  
A more relatively easier way is to use the two classes DataOutput Stream and ByteArrayOutput Stream.  
It provides writeByte( ). writeInt( ), writeShort( ), writeLong( )  
And by running code implementing this it given out the same result.  
  
3.1.2 Strings and Text   
  
If we can learn how to encode text, we can do it with all kind of data.  
  
We can represent numbers and boolean values as String, and a String can be converted to a Byte using the getBytes( )method.  
  
Text is made up of symbols and characters. In fact every String instances corresponds to a sequence ( array ) of characters .   
  
Char value in Java could be internally understood as an Integer.  
Ex: The letter “a” corresponds to the integer 97.  
  
A mapping between a set of symbols and a set of characters is called a coded character set, such as ASCII.  
  
The sender and the receiver must agree on how these integers will be presented as byte sequences, that is an encoding scheme.  
  
The combination of a coded character set and an encoding scheme is called a charset.   
  
On many platform the default charset is UTF-8, but that just one but many standarized character set of the world.   
   
3.1.3 Bit-Didling: Encoding Booleans  
  
Bitmaps are a very compact way to encode boolean information, which is often be used in protocols.   
  
Each bits of integer type can encode one boolean values – typically with 0 representing false, and 1 representing true.  
  
A mask is an integer value that has one or more specific bits set to 1, and all others cleared.  
  
Below is the work dealing with int-sized bit maps:  
  
//Set   
  
  
  
   
//Clear



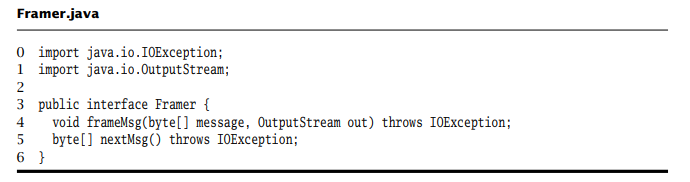
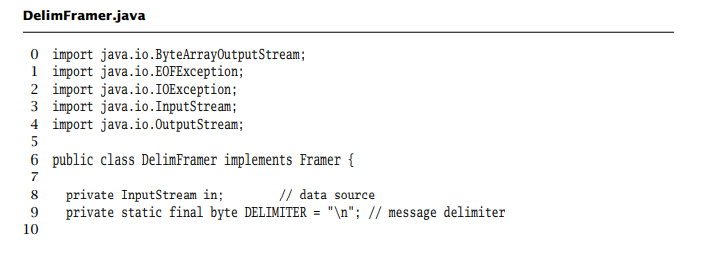
// Set and Clear at the same time

3.2 Composing I/O Streams  
  
Java’s stream classes have powerful capibilities.  
We can wrap the OutputStream of a Socket instance in a BufferedOutputStream to improve performance by buffering bytes and flush them down to underlying channels.   
Then wrap those instances in DataOutputStream to send primitive data types.   
  
Here is the code for that composition:

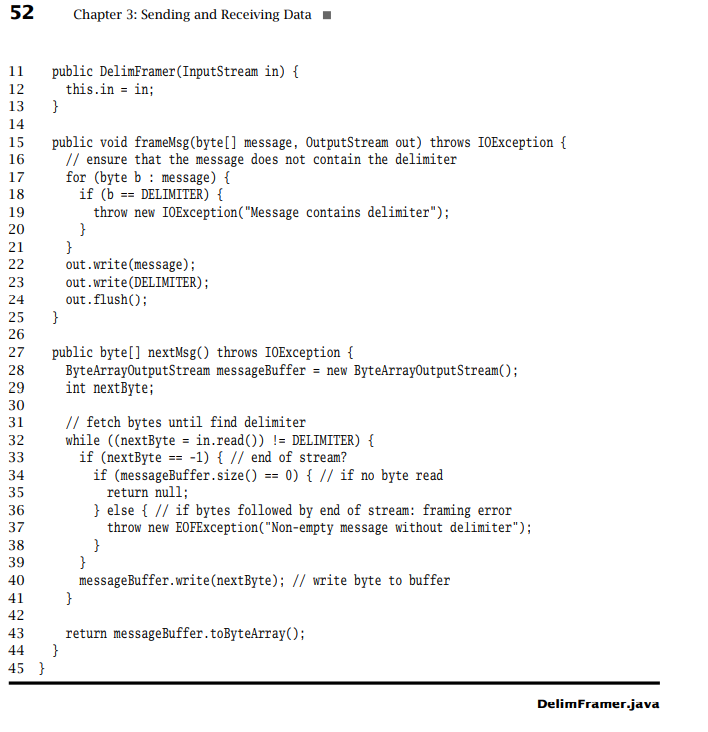
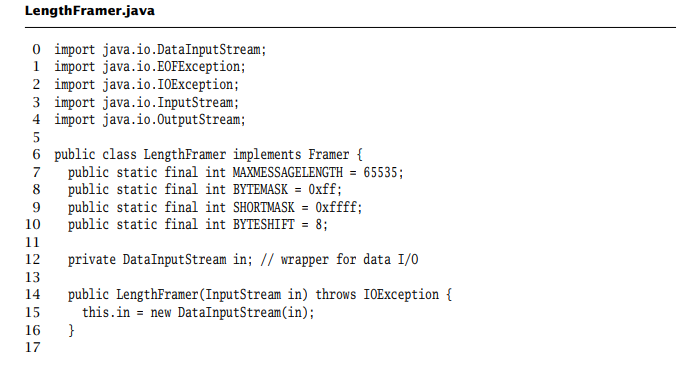
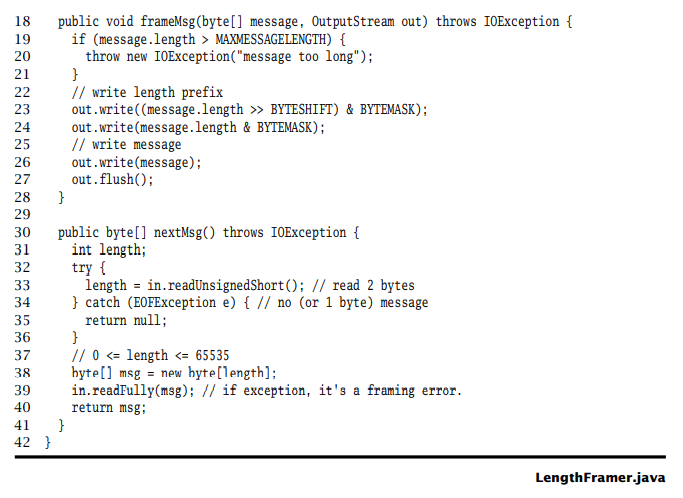






3.3 Framing and Parsing  
  
The original info must be recovered at the receiver from the transmitted sequence of bytes.  
  
Framing refers to the problem of enabling the receiver at the start or end of a message.   
  
Under all circumstances, the application protocol must specify how the receiver of a message can determine when it has received all of the message.   
  
There are some problems need to be dealt with, such as: TCP has no notion of message boundaries.   
  
There are two techniques enable a receiver to unambigously find the end of a message:  
  
Delimiter-based: The end of the message is indicated by a unique marker, an explicit byte sequence that is not occur in the data.  
Explicit length: The variable-length field or message is preceded by a fixed-size length field that tells how many bytes it contains.   
  
Delimiter-based method can be used for the last message sent on TCP connection and often used with message encoded with text: A particular character or a sequence character is marked at the end of the text.   
  
The drawback is that the delimeter must not be contained in the message itself.  
  
There are stuffing techique which makes the message does not regconize them as such.  
  
The disadvantages of this is that both sides must scan for the message.  
  
Length based approach is much simpler but requires the upper bound to be known.   
  
To demonstrate those tecnique, we can use the interface Framer, which has two method framMsg( ) and nextMsg ( ):  
  
  
The class DelimFramer implements delimeter-based framing using the “new-line” character.  
  


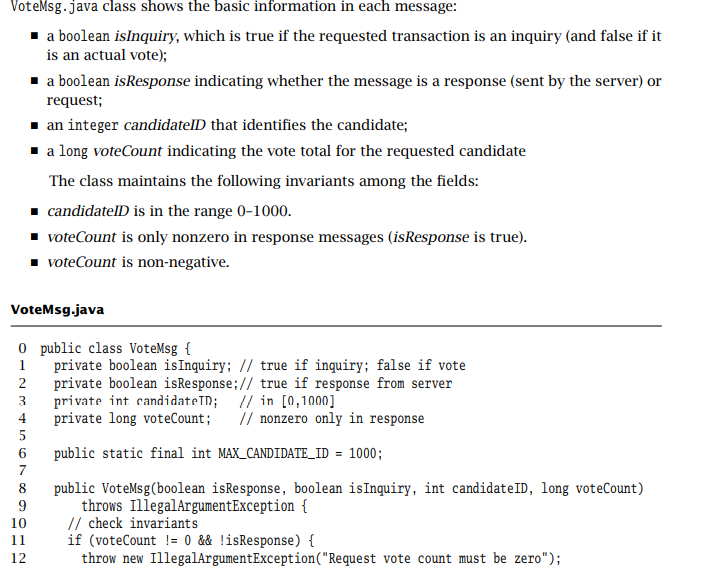


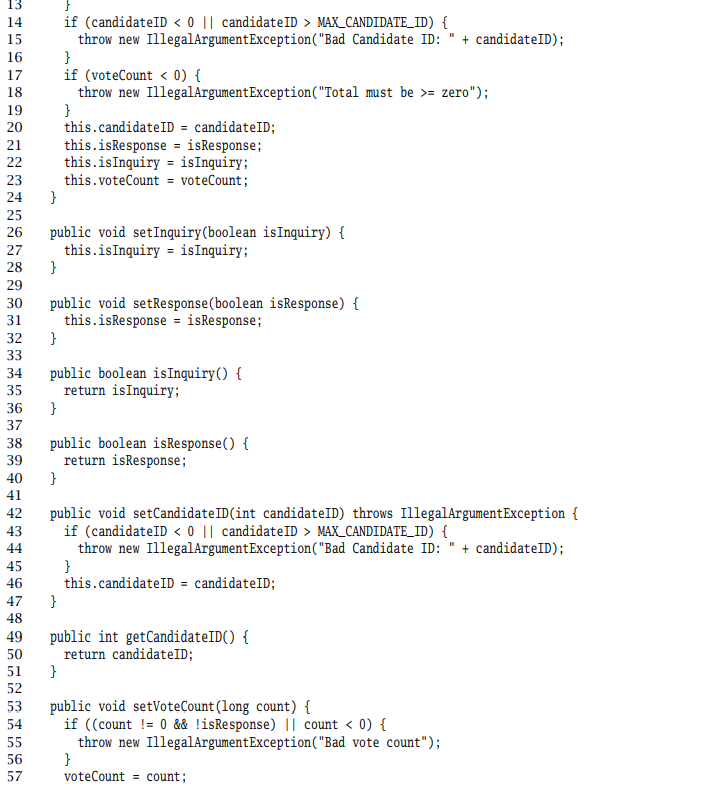
Constructor: line 11-13  
framMsg( ) adds framing information: line 15-25  
Write message: line 22  
Write delimeter: line 23  
Read each byte in the stream until the delimeter is found: line 32  
nextMsg( ) extracts message from input: lines 33-39  
Write non-delimeter byte to message buffer: line 40  
Return contents of message buffer as byte array: line 43  
  
The class LengthFramer.java implements length-based framing for messages up to 65,536 bytes (2^16 -1) in length.   
Note that, with this framing method, the sender does not have to inspect the content of the message being framed; it needs only to check that message does not exceed the length limit.   
  




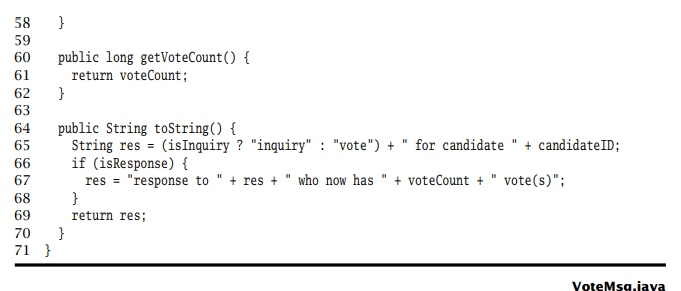
Contrustor: lines 14-16  
frameMsg( ) adds framing information: lines 18-28   
Verify length: lines 19-21  
Ouput length field: lines 23-24  
Output message: line 26  
  
nextMsg ( ) extracts next frame from input: lines 30-41  
 Read the prefix length: lines 32-36  
Read the specific number of bytes: lines 38-39  
Return bytes as message: line 40   
  
3.4 Java Specific Encodings  
  
When you build sockets, you are building a program on both protocols.  
When you know that both end of the communication will be implemented in Java, you can make use of Java’s bulit in facilities like the Serializable interface or the Remote Method Invocation (RMI) facility.  
  
For some reasons, they are not the best solutions, such as Serializable can not be use when a different format has been specified.   
  
Therefor, it is much more convenient to “build your own”.

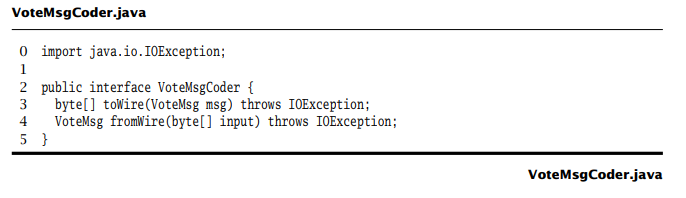
3.5 Constructing and Parsing Protocol Message.   
  
Example of a implement a protocol specified by someone else. Below is a voting protocol.

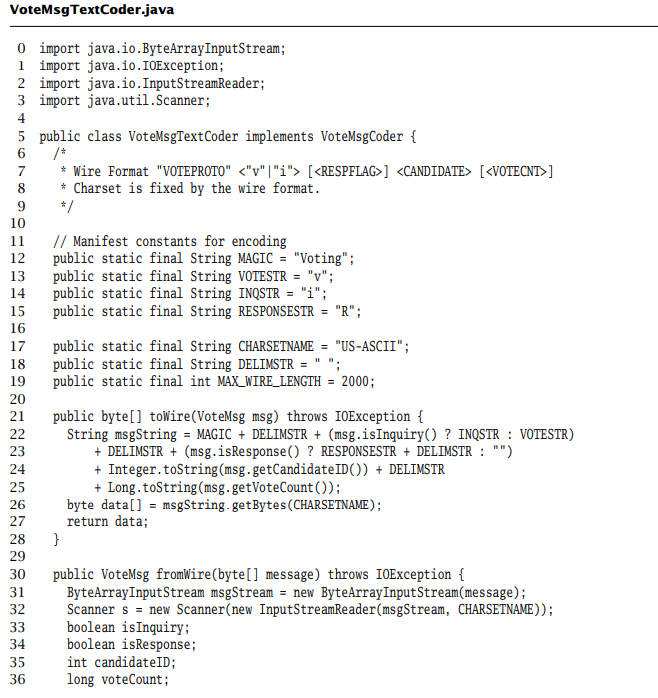


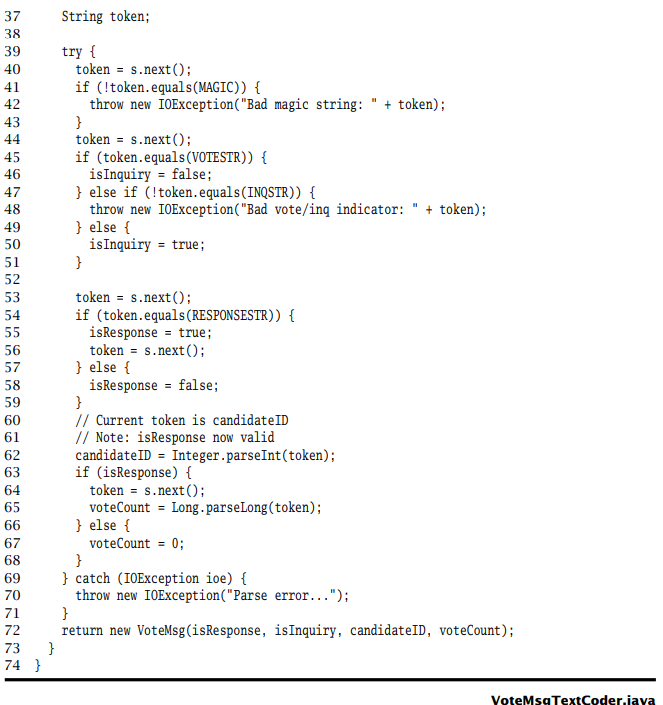




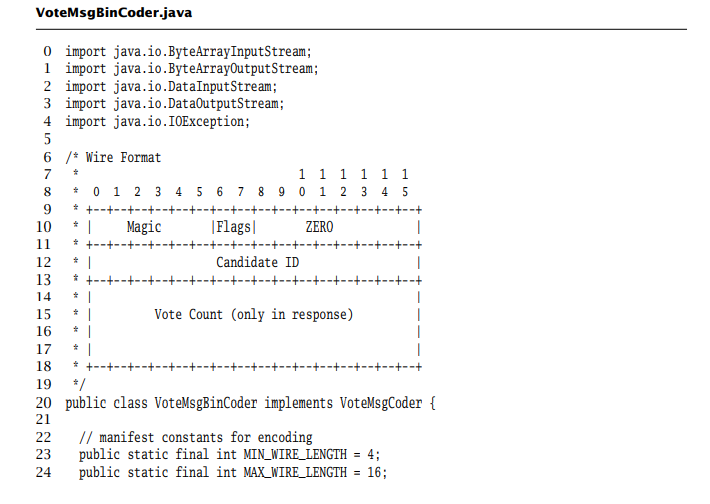


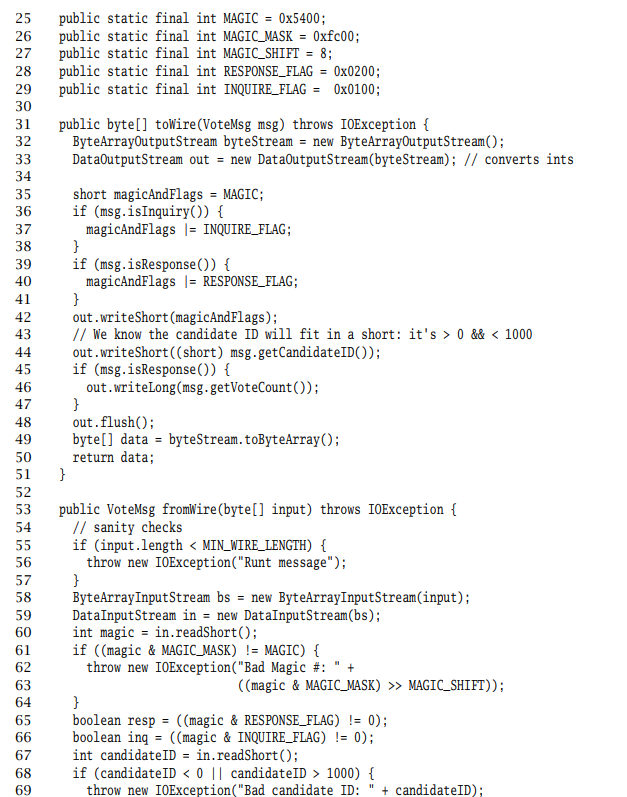
Now the requirements are the encode and decode of the protocol:

3.5.1 Text-Based Representaion  
  
The first version will be a protocol specifies that the text will be encoded using the ASCII charset.   
  
This version begins with what we called a “magic string‘’ – a sequence of characters that allows receipent to quickly regconize the message as a Voting protocol message, or ran dom garbage that arrives on the network.

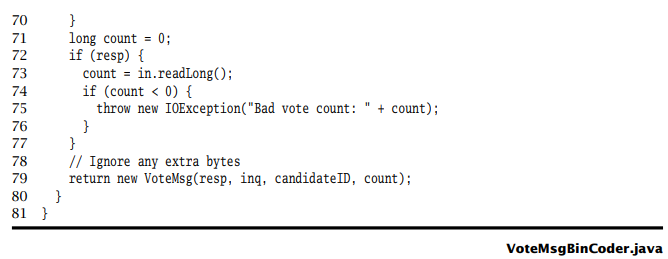


There is an important point in implementing protocol in a network: never assume anything about input in the network. Therefore, your program must be ready for any input.  
  
3.5.2 Binary Presentation  
  
Here’s another different way to encode the Voting Protocol   
  
The binary format uses fixed-size message.

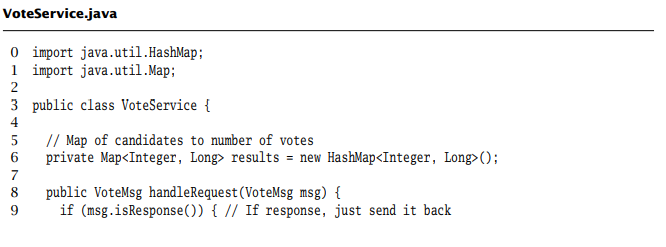










3.5.3 Sending and Receiving

