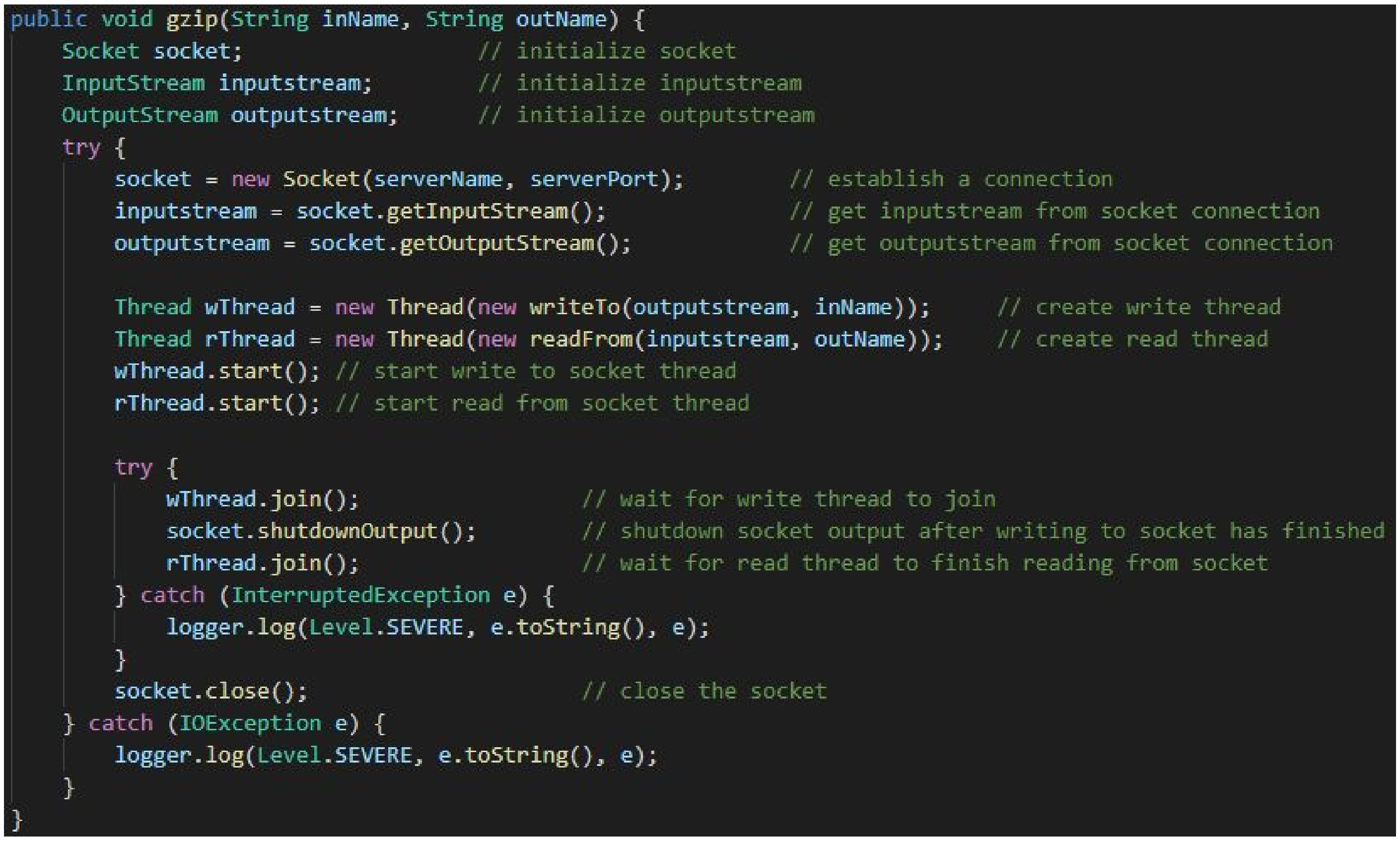
Computer Science 441: Assignment 1

Name: Jack Yang   UCID: 30062393



```java
public void gzip(String inName, String outName) {
    Socket socket;                  // initialize socket
    InputStream inputstream;        // initialize inputstream
    OutputStream outputstream;      // initialize outputstream
    try {
        socket = new Socket(serverName, serverPort);        // establish a connection
        inputstream = socket.getInputStream();              // get inputstream from socket connection
        outputstream = socket.getOutputStream();            // get outputstream from socket connection

        Thread wThread = new Thread(new writeTo(outputstream, inName));     // create write thread
        Thread rThread = new Thread(new readFrom(inputstream, outName));    // create read thread
        wThread.start(); // start write to socket thread
        rThread.start(); // start read from socket thread

        try {
            wThread.join();                 // wait for write thread to join
            socket.shutdownOutput();        // shutdown socket output after writing to socket has finished
            rThread.join();                 // wait for read thread to finish reading from socket
        } catch (InterruptedException e) {
            logger.log(Level.SEVERE, e.toString(), e);
        }
        socket.close();                     // close the socket
    } catch (IOException e) {
        logger.log(Level.SEVERE, e.toString(), e);
    }
}
```

main method signature gzip

The method gzip is the main method that initializes the reading and writing.

First it creates the socket, inputstream, and outputstream and tries to establish a connection with the serverName and serverPort.

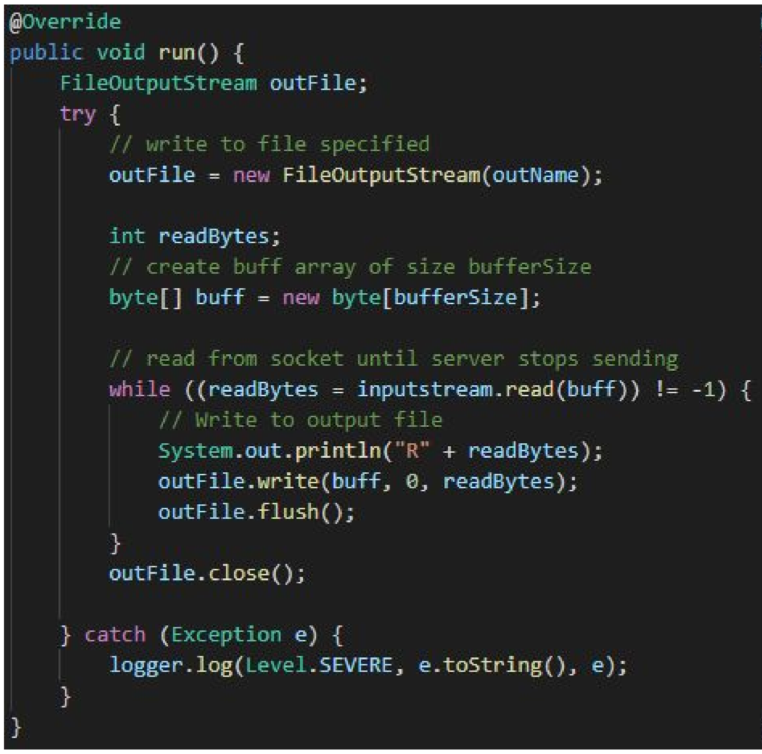
Then it creates a write thread to write to the socket, and creates a read thread to read from the socket.

In the wThread class the run() method runs the code to read from a file until EOF and writes whatever has been read from the file to the socket outputstream, then closes the fileinputstream with inFile.close().

At the same time, in the run() method of rThread, bytes are read from the socket inputstream.

In order for the rThread to finish, the wThread needs to finish and send a shutdownOutput(), therefore back in the gzip() method wThread.join() waits for write to socket to finish first, then signals the server there is nothing to read anymore. After everything has been read from the server and the entire compressed file has been created, fileoutputstream is closed with outFile.close() and rThread finishes

In the end, the socket is closed by socket.close() and everything is cleaned up.



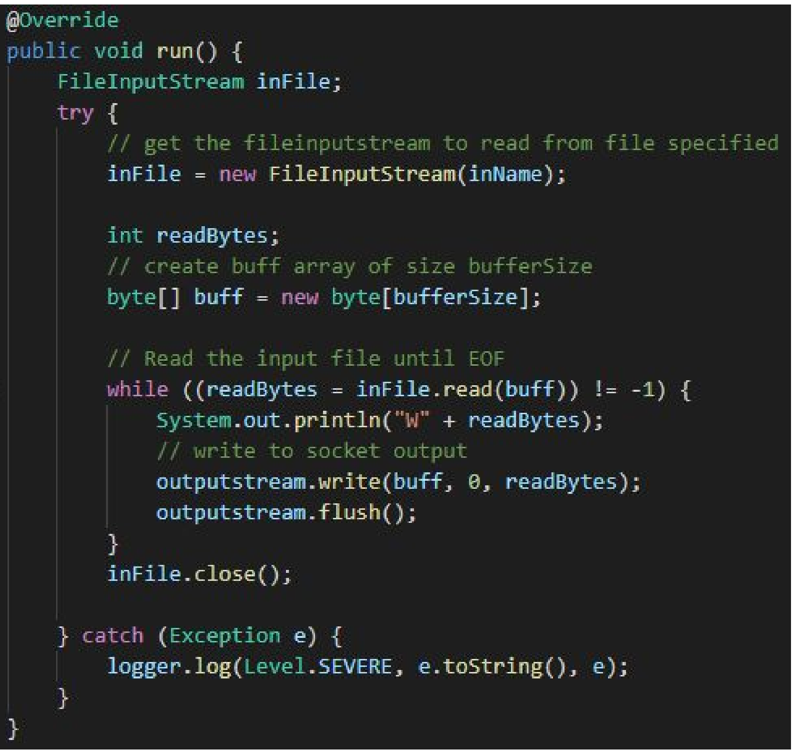
```java
@Override
public void run() {
    FileOutputStream outFile;
    try {
        // write to file specified
        outFile = new FileOutputStream(outName);

        int readBytes;
        // create buff array of size bufferSize
        byte[] buff = new byte[bufferSize];

        // read from socket until server stops sending
        while ((readBytes = inputstream.read(buff)) != -1) {
            // Write to output file
            System.out.println("R" + readBytes);
            outFile.write(buff, 0, readBytes);
            outFile.flush();
        }
        outFile.close();

    } catch (Exception e) {
        logger.log(Level.SEVERE, e.toString(), e);
    }
}
```

run() of wThread



```java
@Override
public void run() {
    FileInputStream inFile;
    try {
        // get the fileinputstream to read from file specified
        inFile = new FileInputStream(inName);

        int readBytes;
        // create buff array of size bufferSize
        byte[] buff = new byte[bufferSize];

        // Read the input file until EOF
        while ((readBytes = inFile.read(buff)) != -1) {
            System.out.println("W" + readBytes);
            // write to socket output
            outputstream.write(buff, 0, readBytes);
            outputstream.flush();
        }
        inFile.close();

    } catch (Exception e) {
        logger.log(Level.SEVERE, e.toString(), e);
    }
}
```

run() of rThread