Assignment 1, CPCS 441, Winter 2021, Lachlan Moore

In order to achieve the implementation of step 4 and 5 effeinctly for larger data sets I used an extra thread. The main program runs parallel with this thread so that data may be written and read by the client at the same time. In my implementation I used the thread to write to the server, while the main program read from the server.

Step 4. Implementation:

I created a method to send a file it takes in the name of the file and then tries to create the file with that given name. If that fails it catches an error and prints it to the stack. Then I open a fileInputStream to read data from the file. I create a buffer set to the size given in the constructor and use a while loop to read the file with the given buffer size. As long as there is data still to read it should return something other than -1 so the while loop will continue. In that loop I write on dataOutputStream to the server with all the data read in that chunk of the buffer as well as print out the amount of bits written to the server (flush data each time from dataOutputStream for good practice). Below is a picture of said while loop:

```
while ((count = fileInputStream.read(buffer)) != -1) {
    dataOutputStream.write(buffer, 0, count);
    dataOutputStream.flush();
    // print details for assignment
    System.out.println("W " + count);
}
```

Finally after the loop is completed I clean up streams by closing the opened fileInputStream, and outside of the method I created I call shutdownOutput on the socket in order to signal to the server I am done writing to it.

Step 5. Implementation:

Similarly to step 4 I created a separate method to handle this section, it takes in the name of the file to create. I start by creating a fileOutputStream as there is no file to open but to create with data from the server. I create a buffer set to the size given in the constructor and use a while loop to read the file with the given buffer size. As long as there is data still to read it should return something other than -1 so the while loop will continue. If there is, we write the data in the buffer to the created file (fileOutputStream), print out the amount written to the file. Below is a picture of said while loop:

```
while ((count = dataInputStream.read(buffer)) != -1) {
    fileOutputStream.write(buffer, 0, count);
    fileOutputStream.flush();
    // print details for the assingment
    System.out.println("R " + count);
}
// clean up
```

Finally after the while loop we clean up by closing the newly created file (fileOutputStream).