CEG 4350 OS Internals and Design

Programming Project Report

For completing the Programming Project for OS Internals and Design, I used both the Socket IPC and Threading IPC to do this. Both IPC’s were created in the Java programming language and the Software Program that was used for programming this assignment was NetBeans IDE 8.2.

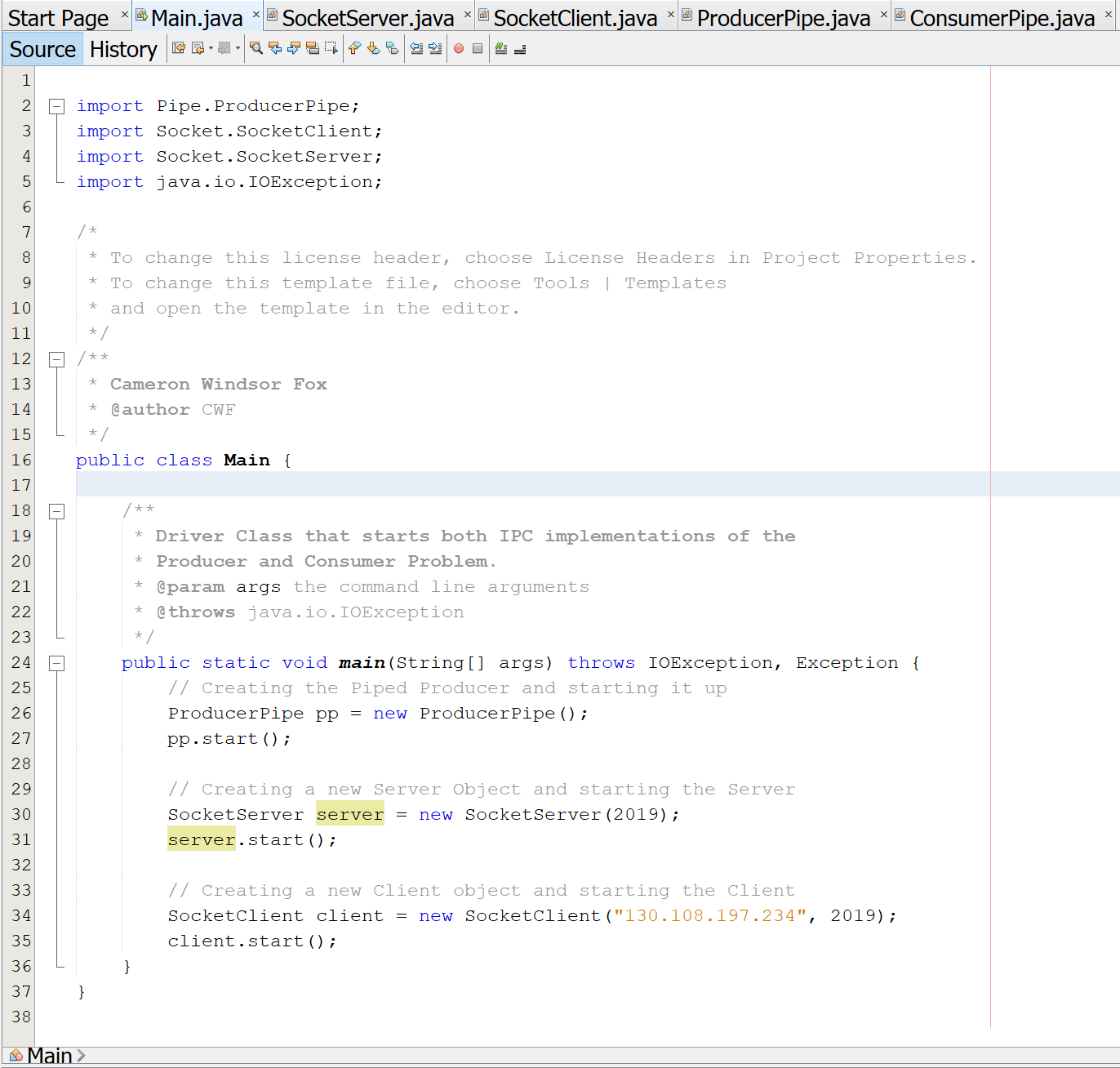
Both implementations are done by generating the 100 integers of data with a producer then sending that data to a file for storing the data and then sending the integer data to the consumer class where they are sent to a separate additional file to checking that data sent and received. This allows us to ensure the al data generated then sent and received was sent in order.

I choose to build this project using the Java language because it one of the languages I am familiar with and I understand the software I use Java with. I went with using Sockets because it was one of the first IPC examples that I am most familiar with and I ws able to recognize them right away. When I finished and worked with piping I was less familiar with how they work and it was a challenge but I was able to figure it out in the end.

Main Class:

The Main class for my project is the Driver class, this will call all the other classes for both the Socket programming and the Piped programming. Both implantations of the Producer and Consumer problem are started and ran from this one class.

Source Code:

Here is the Source Code for the Main Class of the project.

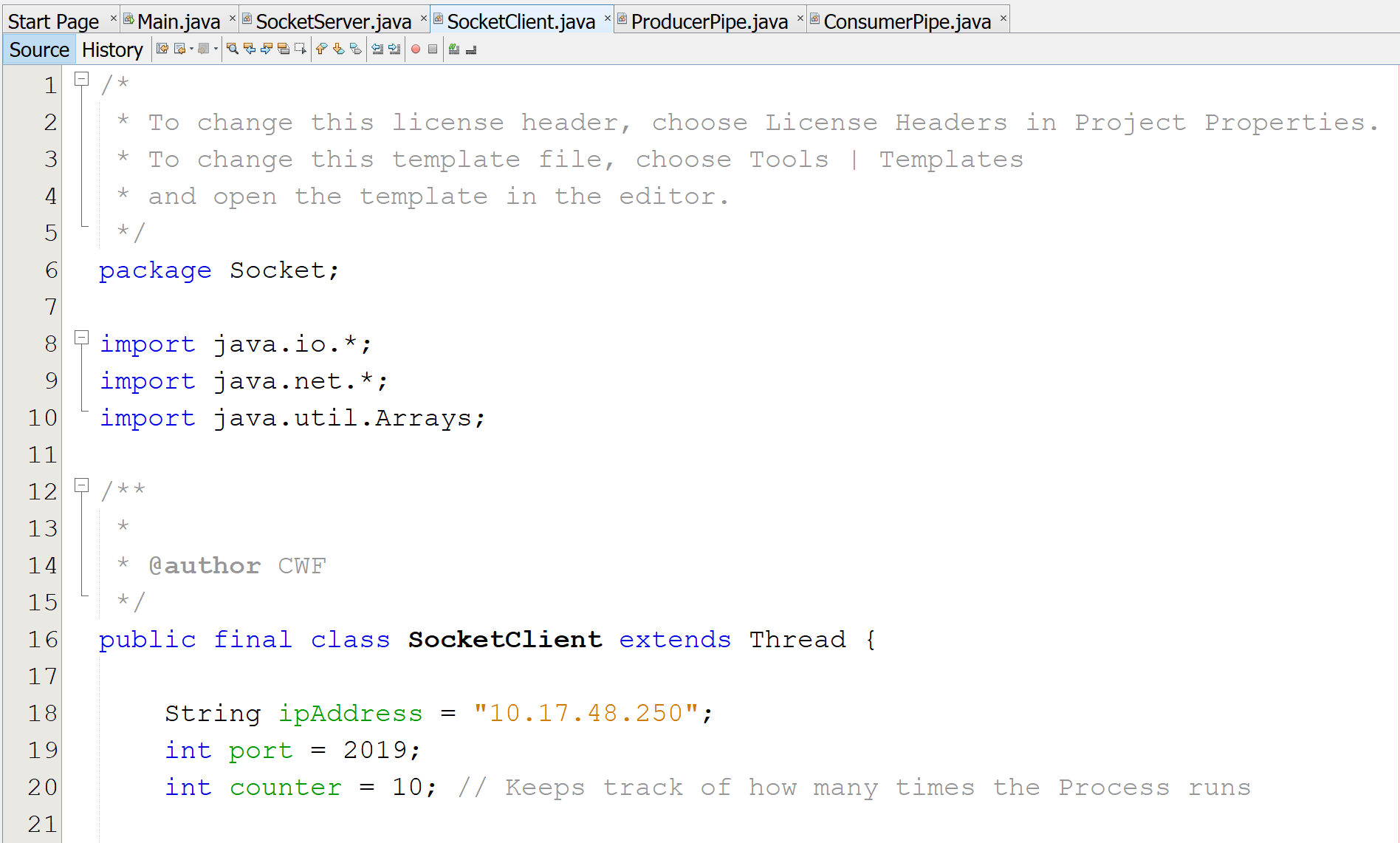
Description of Source Code:

The Main Method here at the start is the Driver class and puts everything else into motion. By starting up the Pipe classes and the Socket classes, everything is controlled and only runs all together if Main is running first.

Socket IPC:

The Socket IPC is done with a Server and a Client class that act as the Producer and the Consumer. The client being the producer creates the 100 data integers, writes them to a file and then sends the data to the Server or Consumer where the data is read and then sent to its own text file to compare each. This way I can see that both Data sets are the same and there are no integers out of order when sent from.

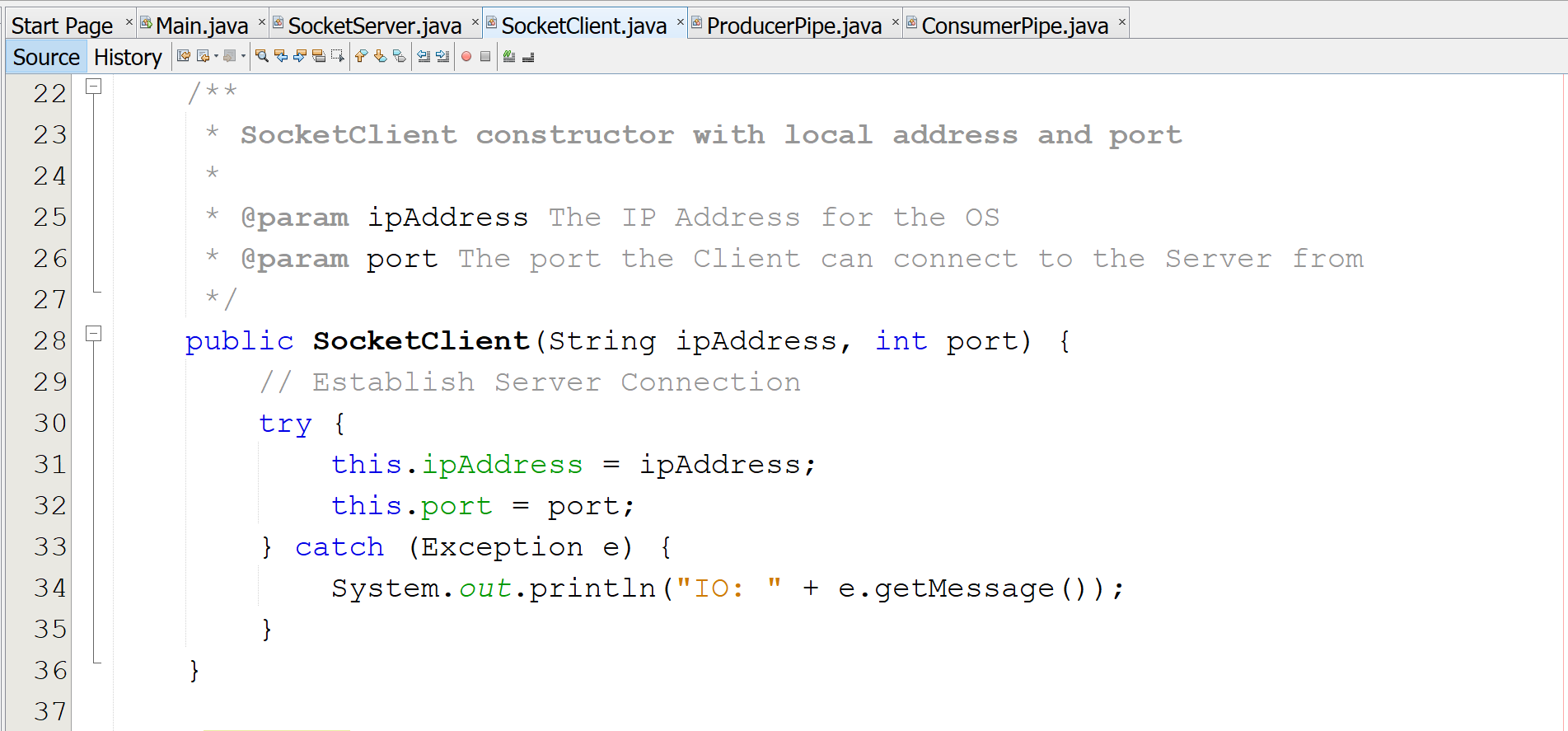
Source Code:

SocketClient Source Code:

Description of Source Code:

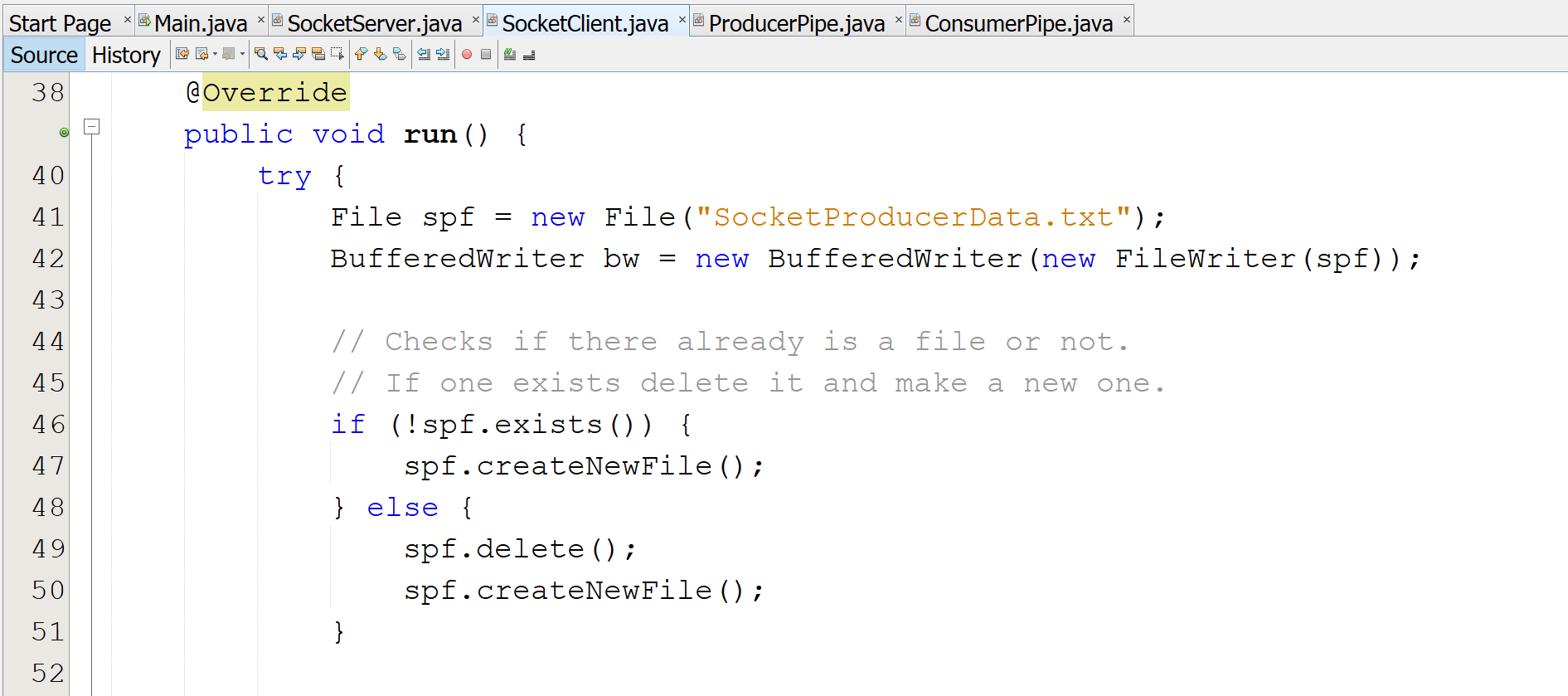
This is the starting portion of the Client Side of the Socket Programming. This is the Producer and staring off I have the IP address of my device and a port number created for connecting to the Server/Consumer when the Server starts up.

I also have a counter variable used to keep track of repeating tests up to ten.



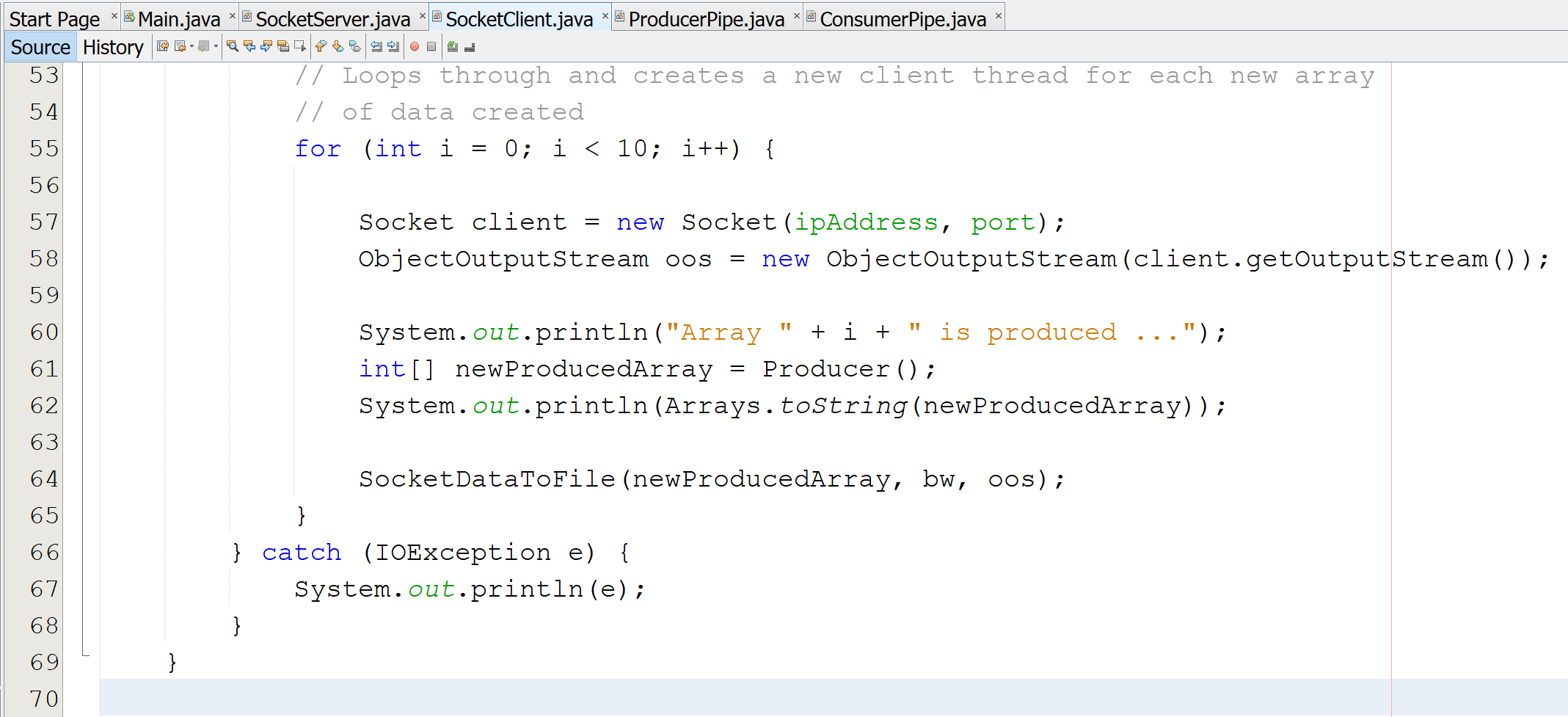
Description of Source Code:

The next section here is the Constructor for the Client Socket. That takes the Global variables created and initializes them.



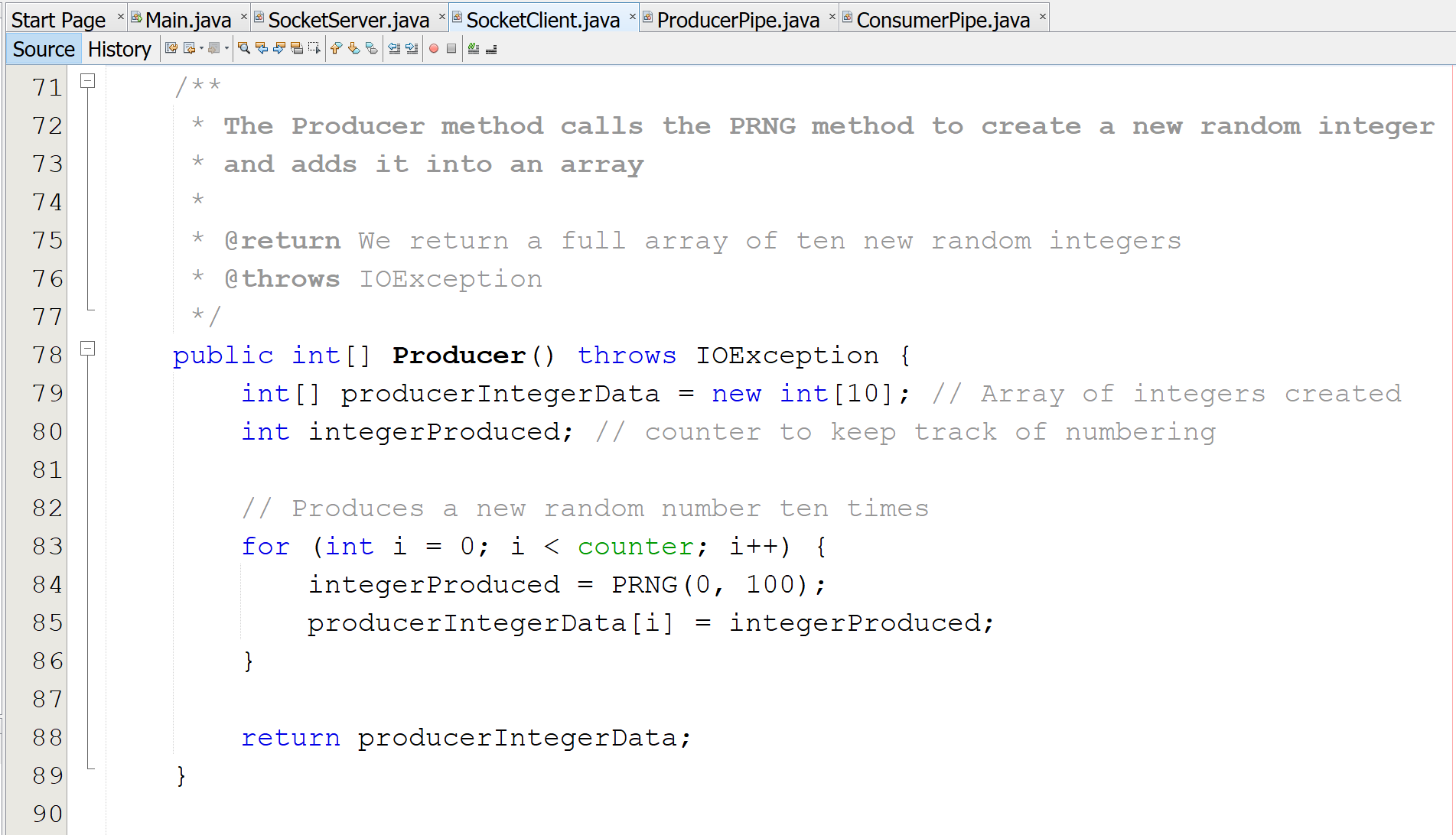
Description of Source Code:

The Run method here is where all the Client Class main meat is located. This is where all the method calls, and functionality are placed. This is only the first part, See below for the other section. This section creates the text file that will store the data sent from the Producer. Along with A BufferedWriter to take the data received from the Producer method and write it out to the Producer Text File.



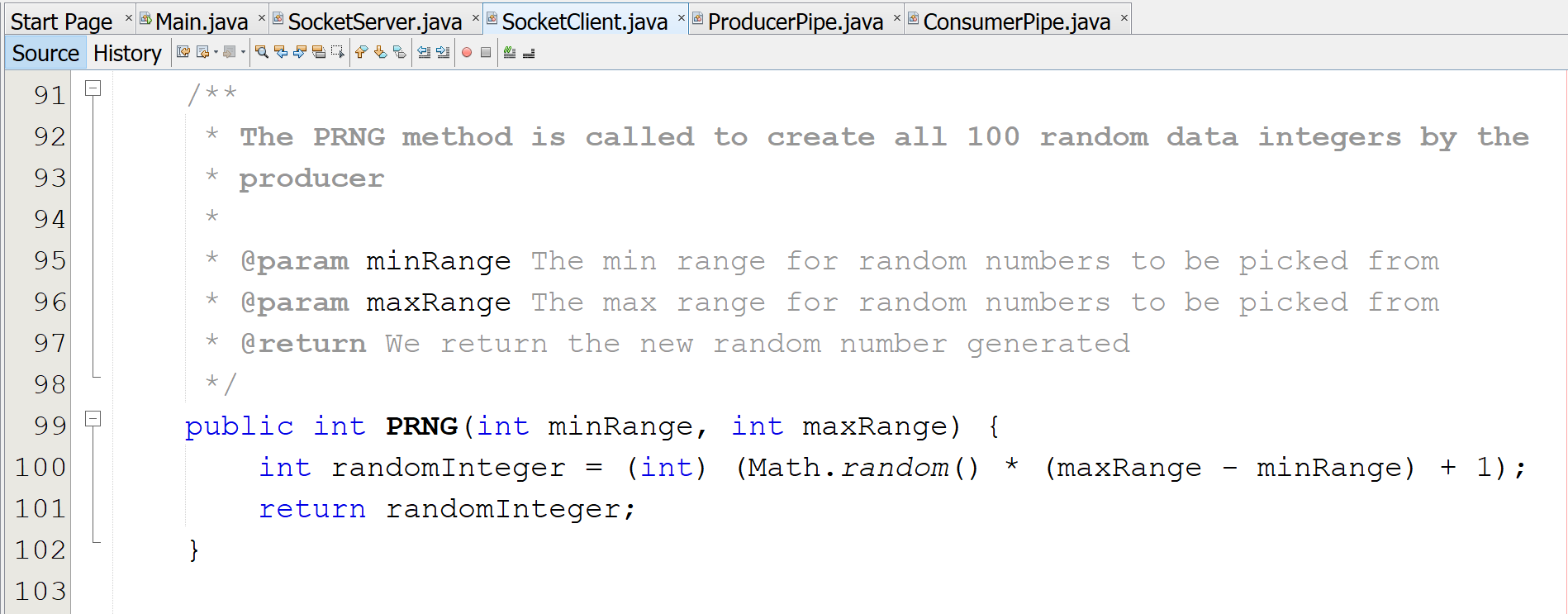
Description of Source Code:

Here is the second part of the Run method. Here I am going through a loop of calling the Producer method and SocketDataToFile method. This is because each call I make to the producer is a new array of ten integers that I am passing to be written out to the Producer Text File and sending the data to the ObjectOutputStream to eb sent to the Consumer or the Server Class.



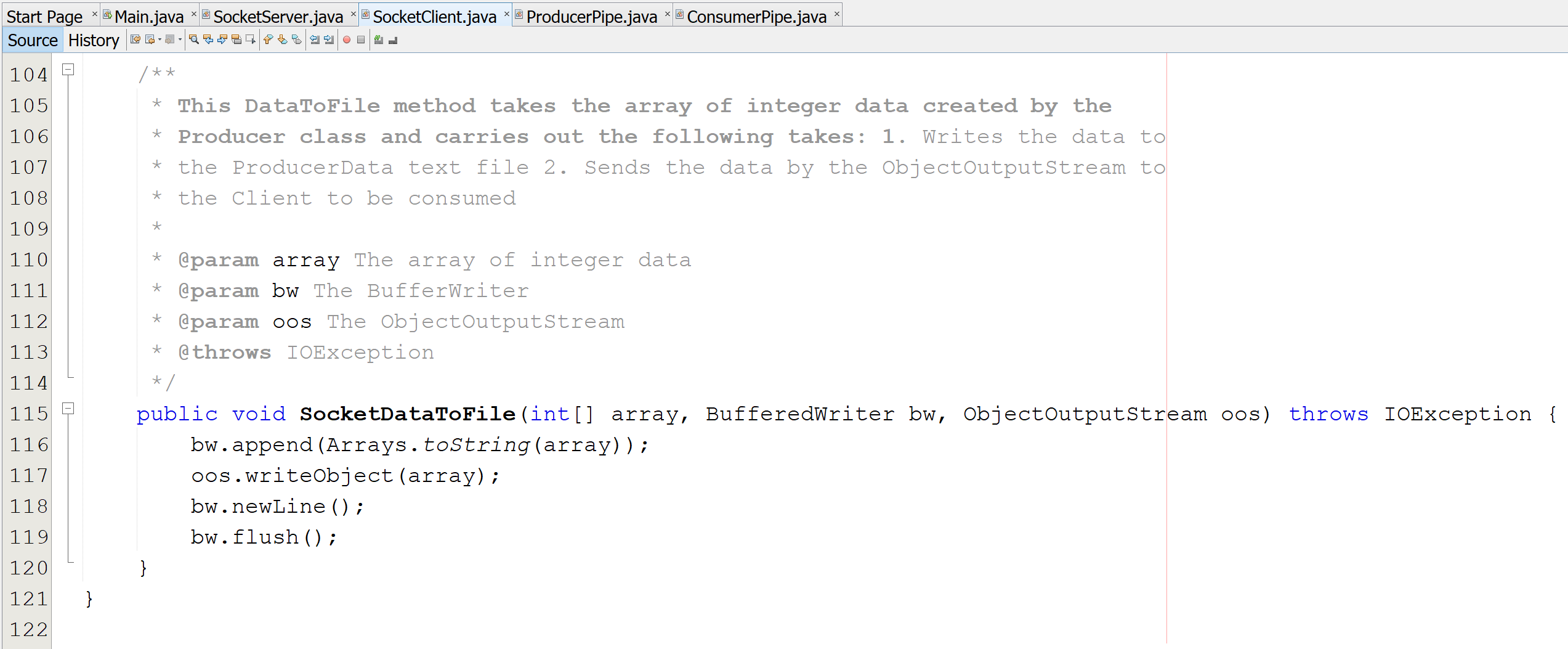
Description of Source Code:

This is the Producer Method mentioned earlier that calls a new method to produce the actual random number and then take the new number generated and add them into a new array with a max size of ten. A new array is made up to a total of ten each full of ten random generated integers.



Description of Source Code:

Here we have the method for actually generating the random integers for the socket producer to add to there arrays and then send to the Consumer.



Description of Source Code:

The final method SocketDataToFile, takes in each new array generated along with a BufferWriter and the ObjectOutputStream in order to write each new array of data to the Producer Text File and then move to a new line when adding the next.

SocketServer Source Code:

Next it is time to review the SocketServer Source code and how it functions.



Description of Source Code:

Similar to the Client we have a few Global variables such as the ServerSocket, Socket, and a counter. The ServerSocket and Socket variables are for the Producer and Consumer connection.



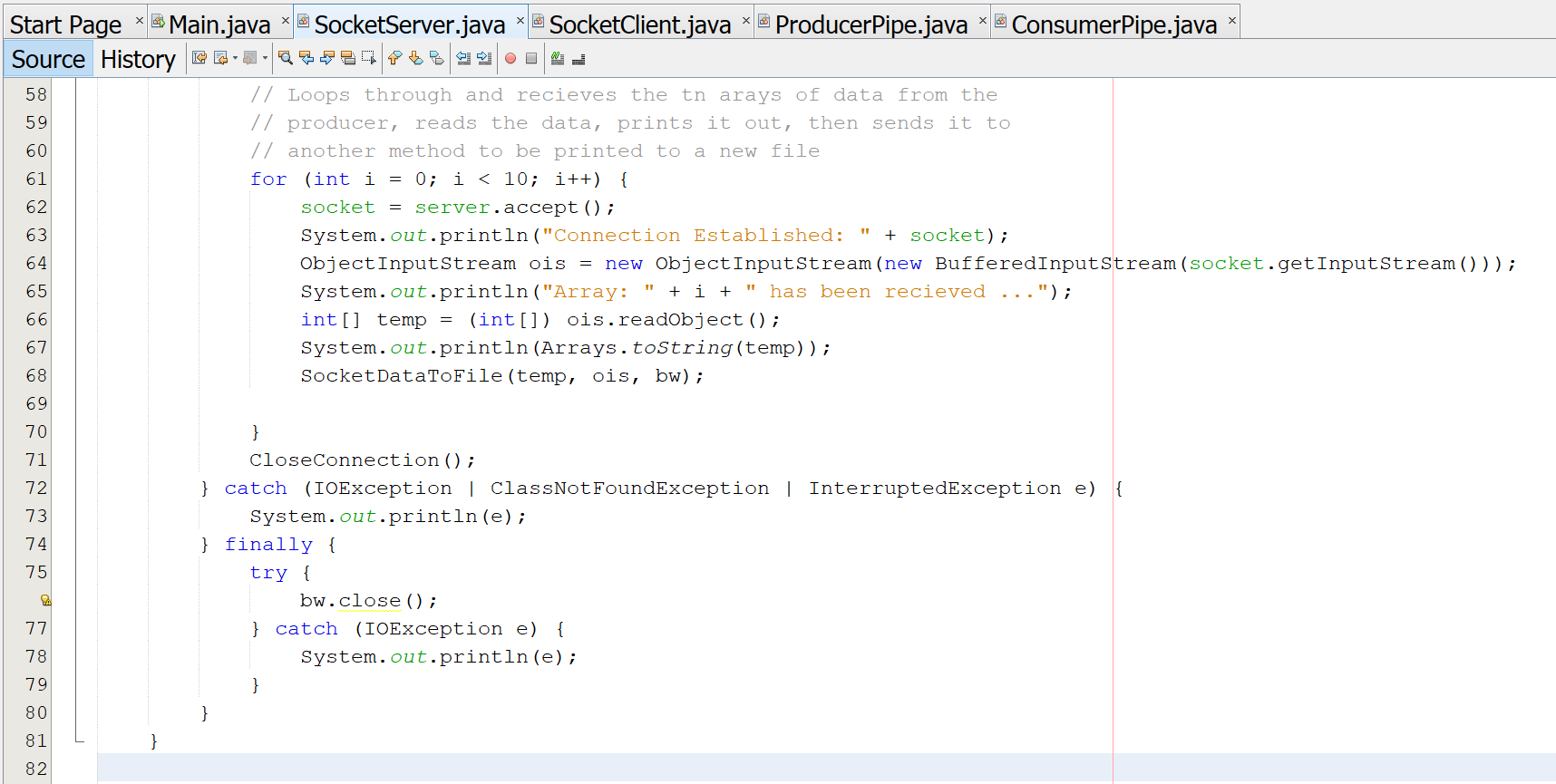
Description of Source Code:

Here we have the constructor for the ServerSocket Class initializing all of the global variables. This is all for the Consumer and Producer connection.



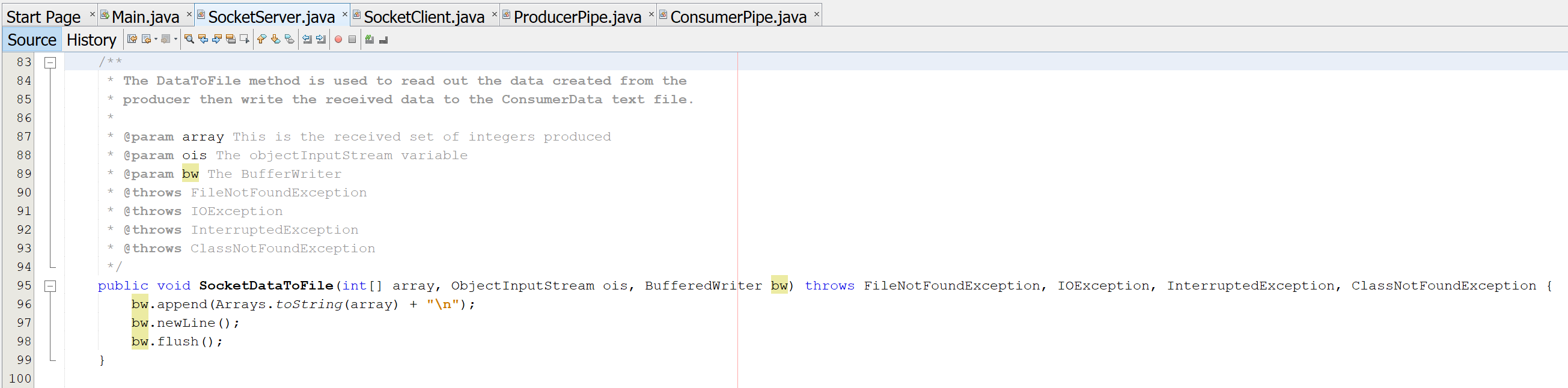
Description of Source Code:

The run method is where all the method calls take place this is only the firsts section and the other parts can be seen before. At the start I am creating the text file used for the Consumer to be filled with the integer data received from the Producer.



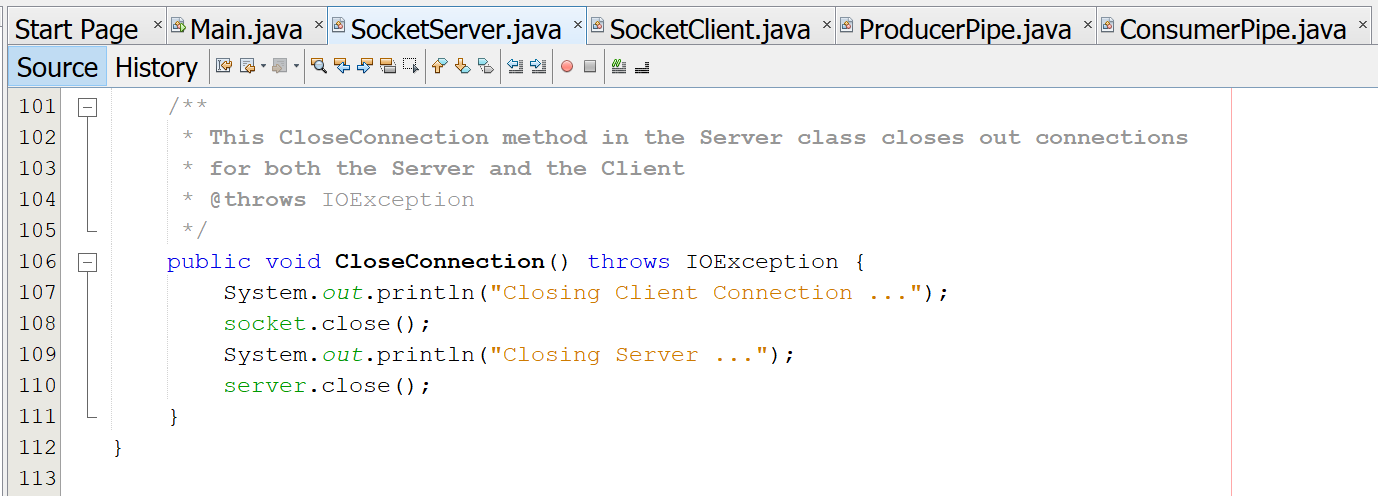
Description of Source Code:

Here we have the second half of the run method for the Consumer where I am reading in the output sent from the producer and displaying the data to the console and then sending the received data to a new method to be written to the Consumer text file.



Description of Source Code:

Here we have the SocketDataToFile method were like the Producer we take the data we received and input it into a new text file.

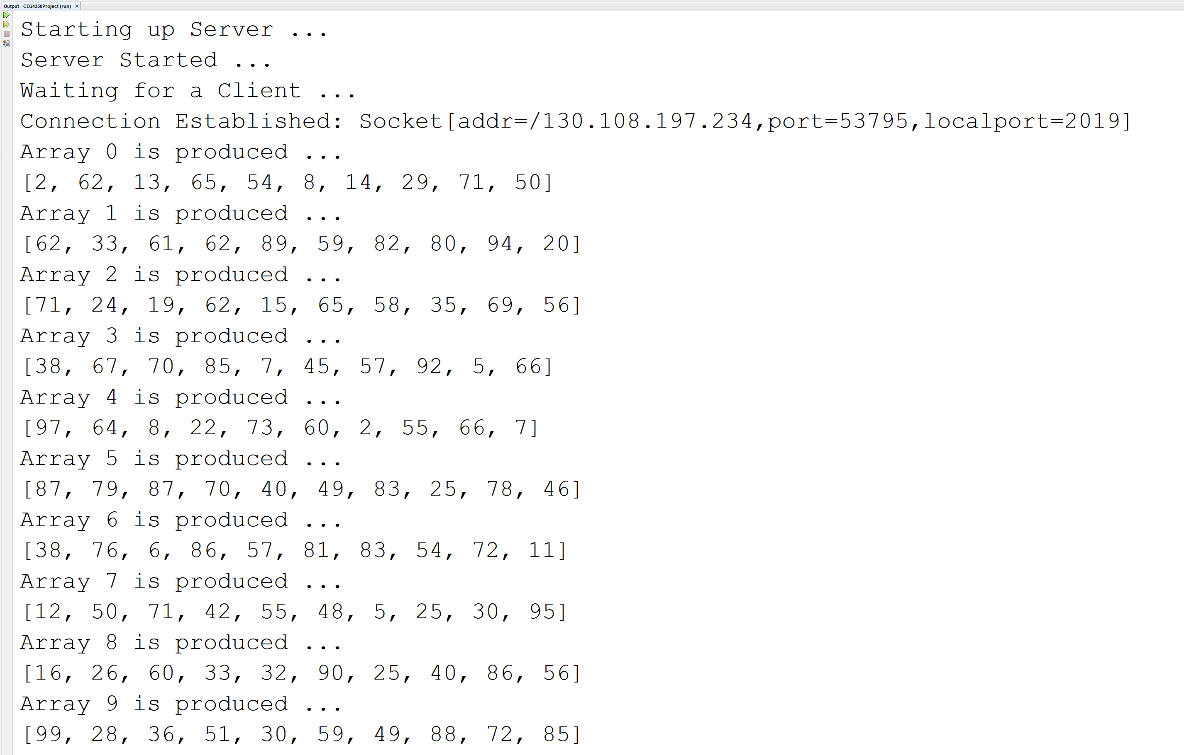


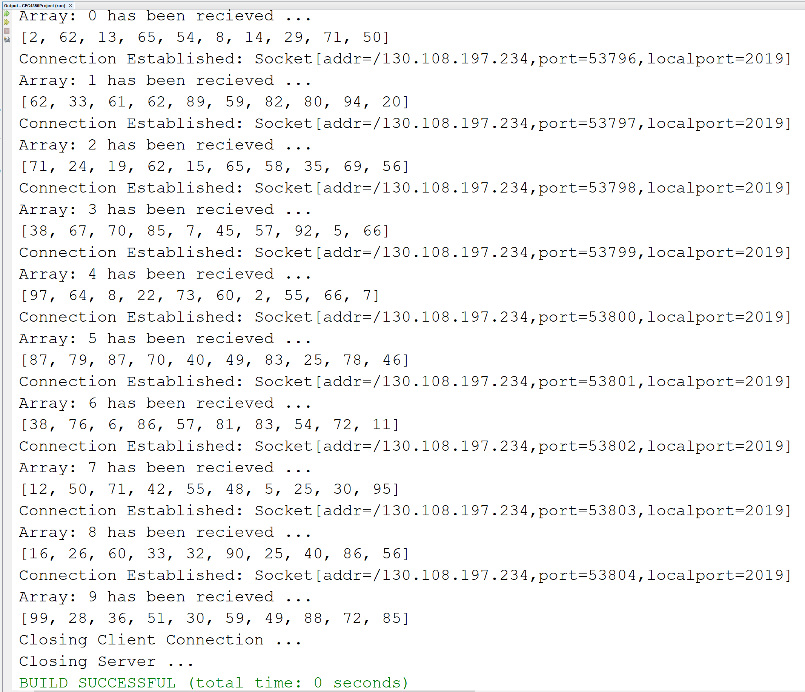
Description of Source Code:

The final method the CloseConnection method is used to close the client and server ending the Socket based Producer and Consumer Implementation. This make sure that the server and client will not continue to run when the processes are already finished.

Results for Socket Programming:

Here is the console output for the Socket implementation of the assignment. With print statements to test and show that the connection between them as well as when the data is produced, what data is produced and then the data sent to the consumer and the displaying the data sent to the consumer to check that there are no problems with the data being transferred from Producer to Consumer.



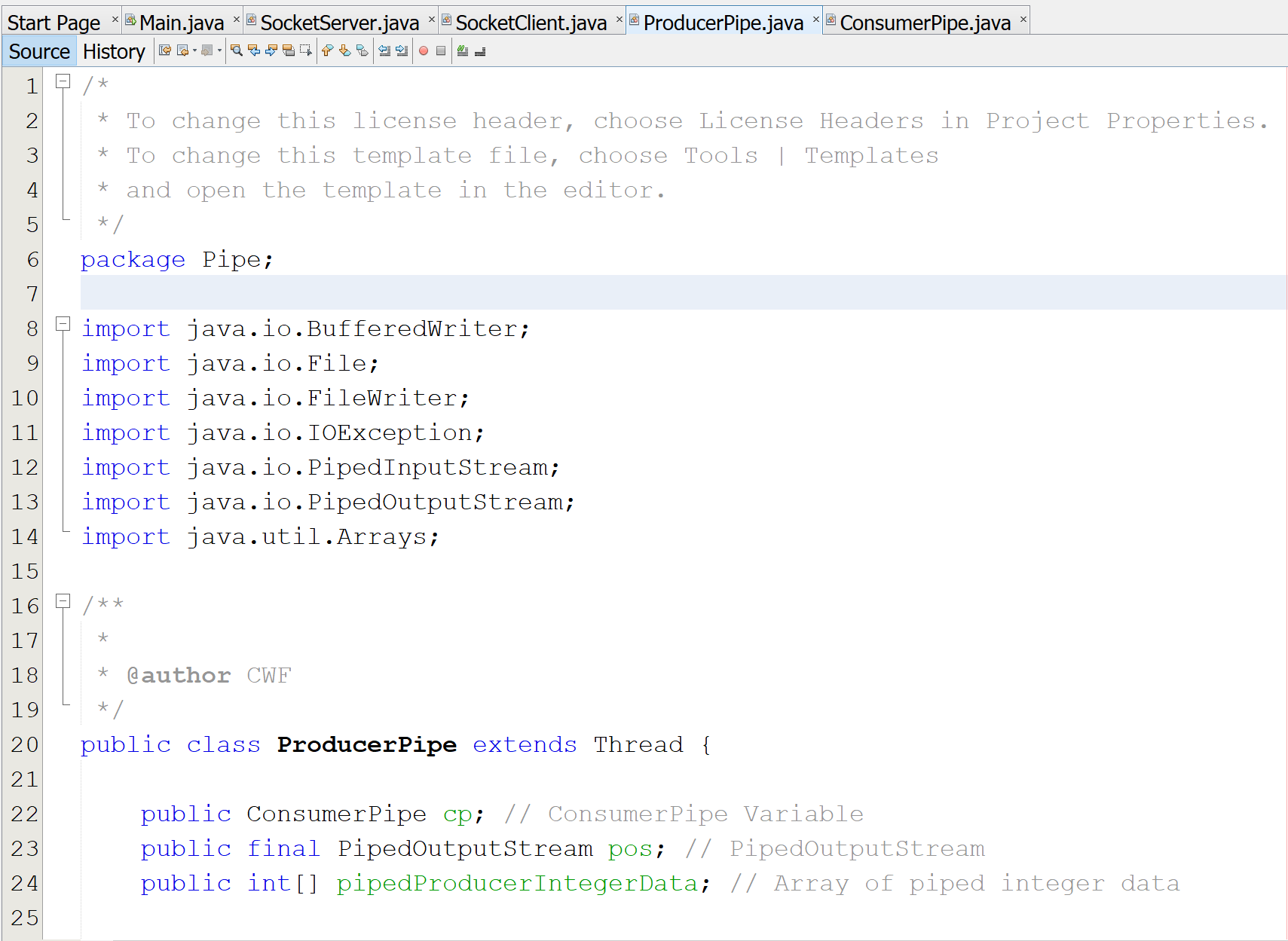


Pipe IPC:

Next I have the Implementation of a piped version of the Producer and Consumer Process for this programming assignment. This is done with no named pipes mainly dealing with the PipedInoutStream and PipedOutputStream. So this section of code and descriptions are going to be much shorter than the descriptions for the Socket based implementation.

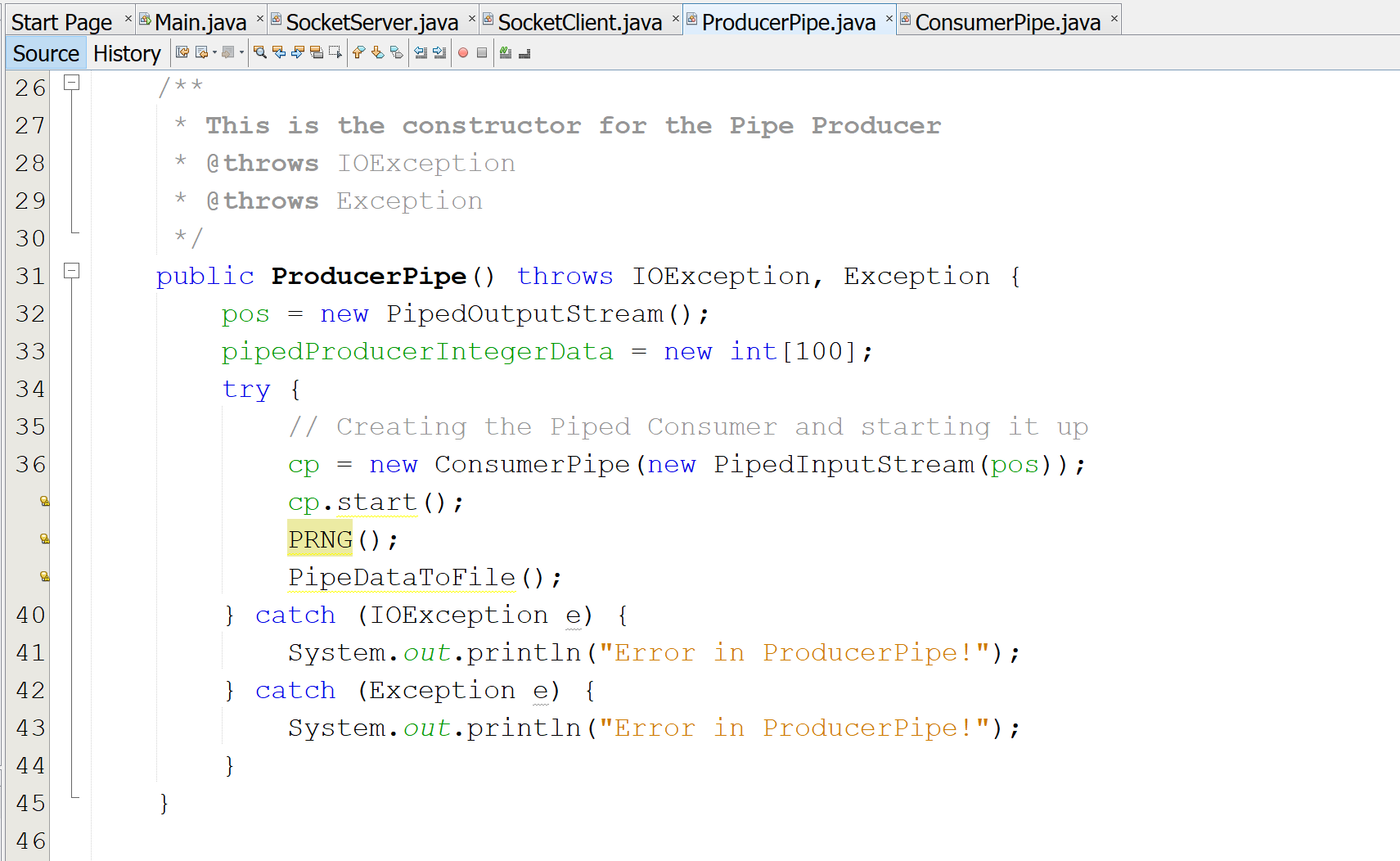
Source Code:

ProducerPipe Source Code:



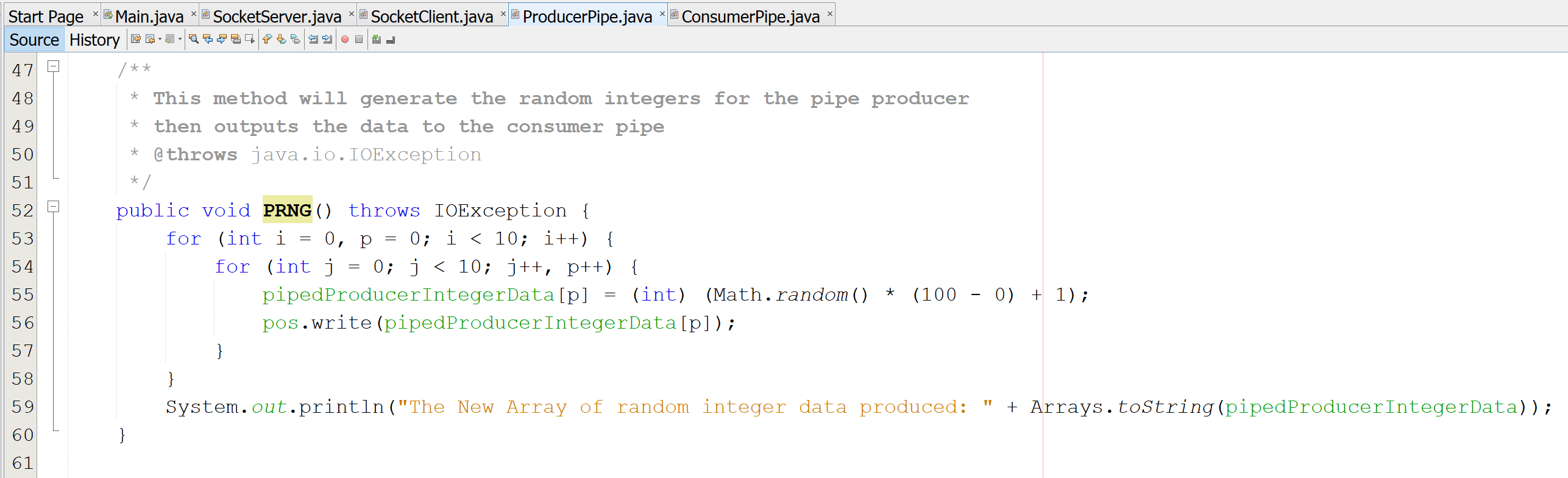
Description of Source Code:

Here I have the start of the Pipe Producer section of code. Here I am creating my global variables of the Array to hold the integers generated and the PipedOutputStream to send the data to the Piped Consumer. As well as a reference variable to the Piped Consumer Class.



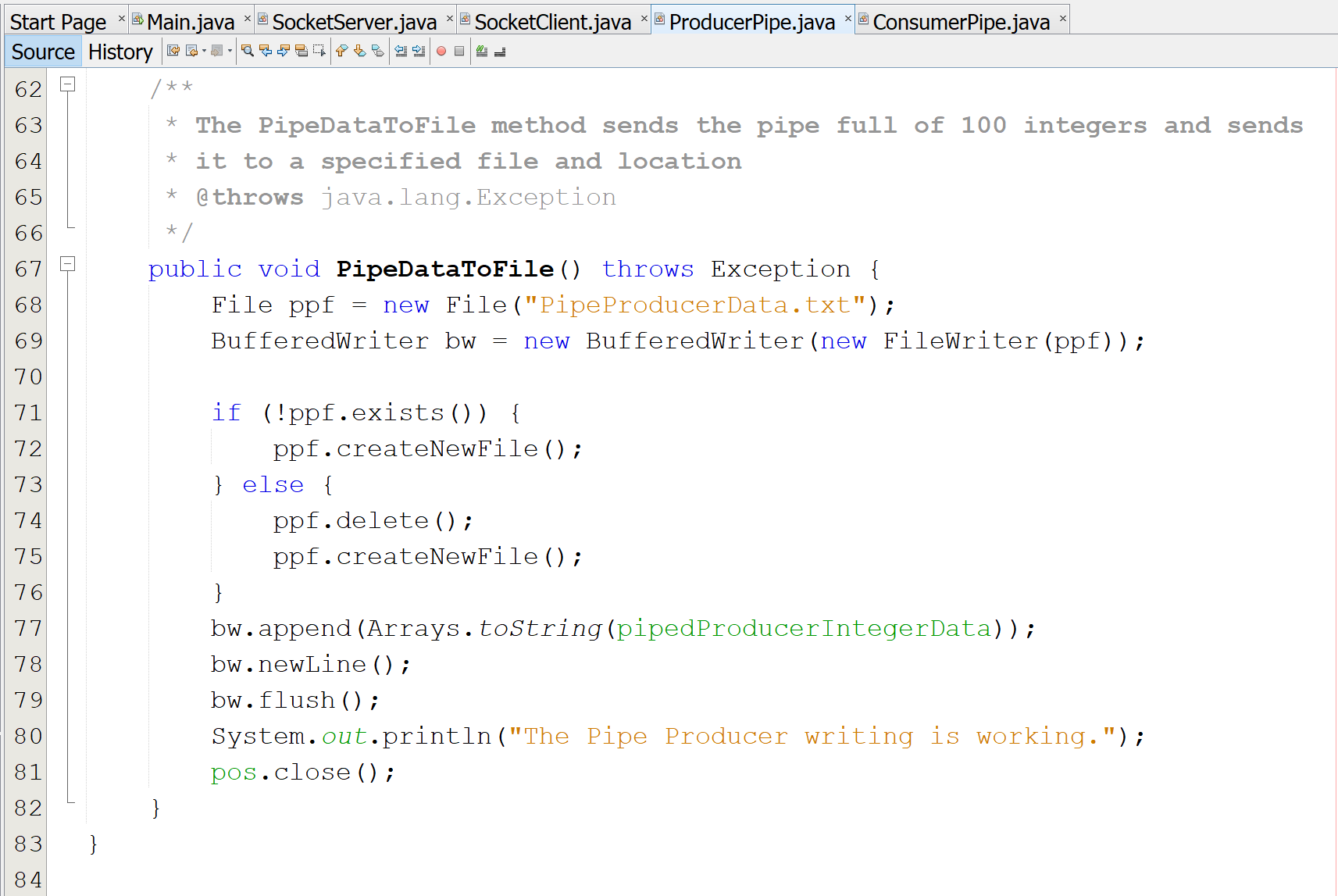
Description of Source Code:

Here I have the Constructor for the ProducerPipe Class. Initializing my variables and setting the array to a new array with a limit of one hundred integers. Not only this but here I am also calling my other methods that are used in the ProducerPipe class.



Description of Source Code:

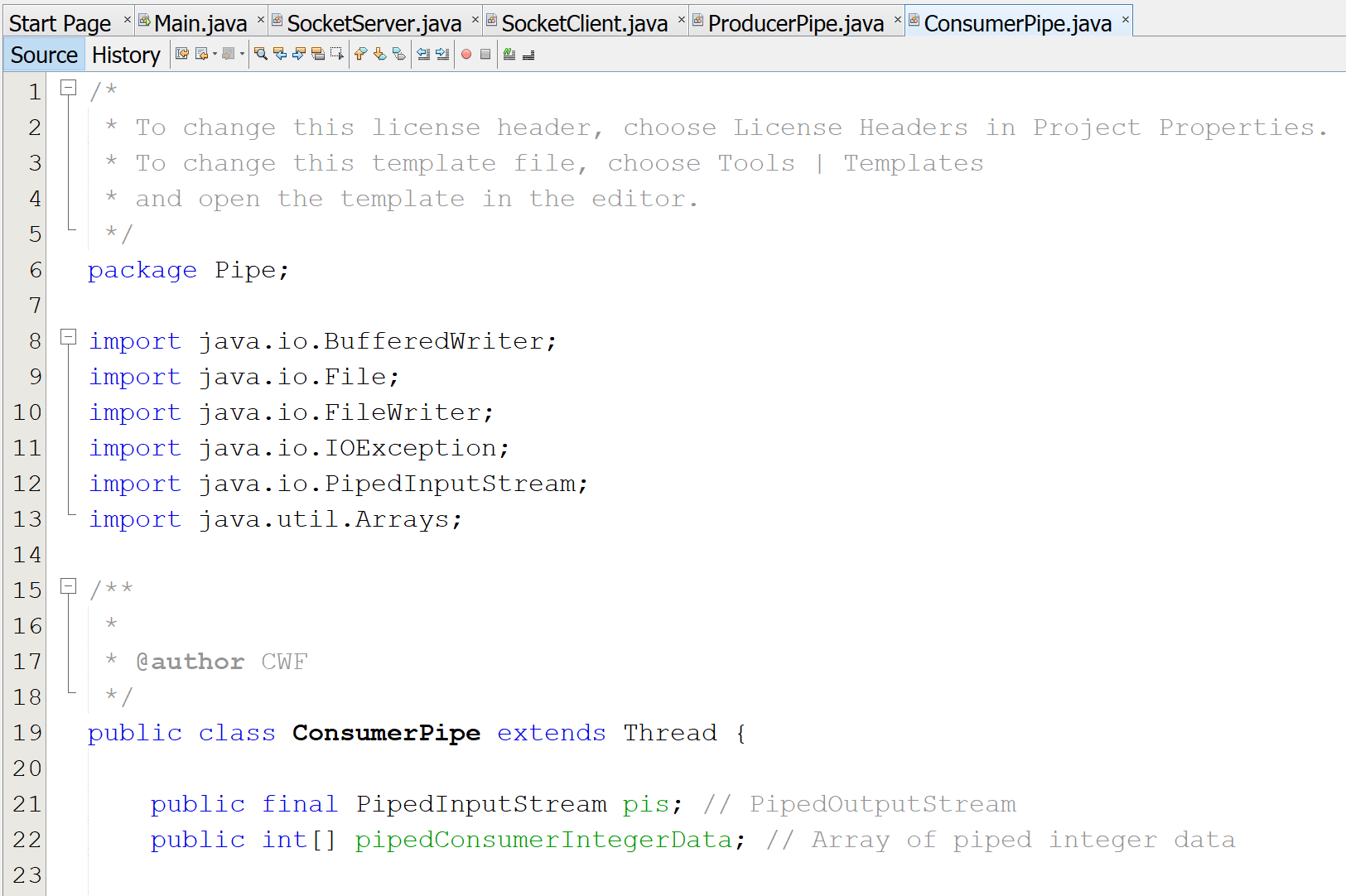
Here is the PRNG method used to generate all the random numbers within the range of zero to one hundred. Each integer generated is then interested into the array and the inner for loop is made to show that as each repeat of the main loop repeats, we are generating another ten new numbers. With ten new integers created repeating ten times we end with a total of one hundred randomly generated integers. As each new item is generated we are sending it to the ConsumerPipe as well. Then at the end of the loops the full list of one hundred integers is displayed to the console.



Description of Source Code:

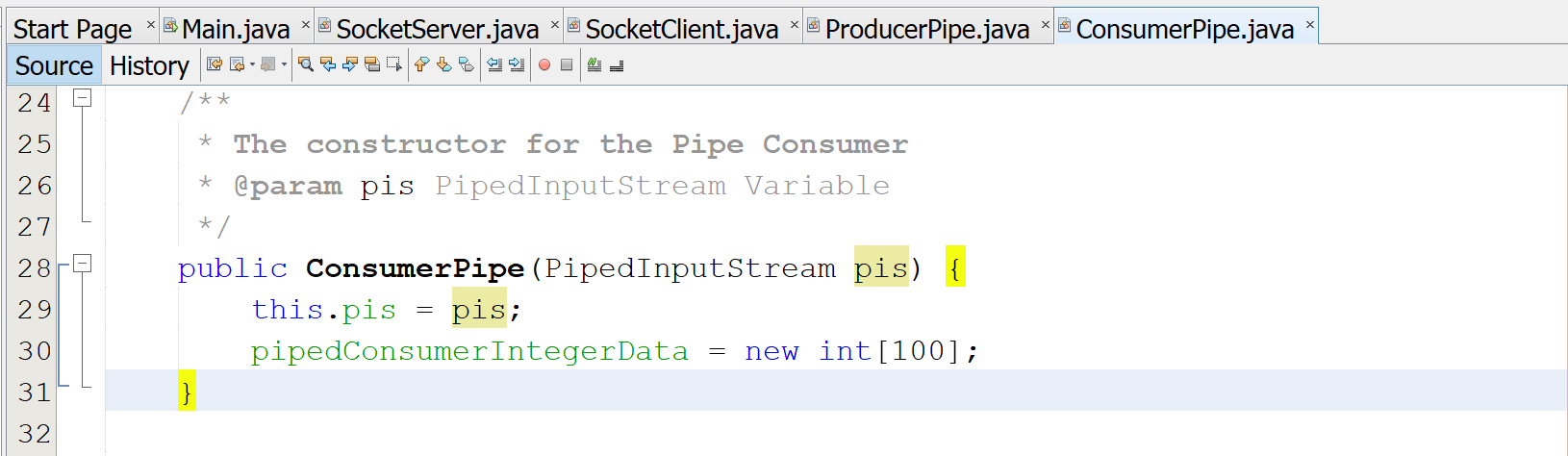
Here we have the PipeDataToFile method that is taking the integer data generated and writing it out to the Pipe Producer text file.

ConsumerPipe Source Code:



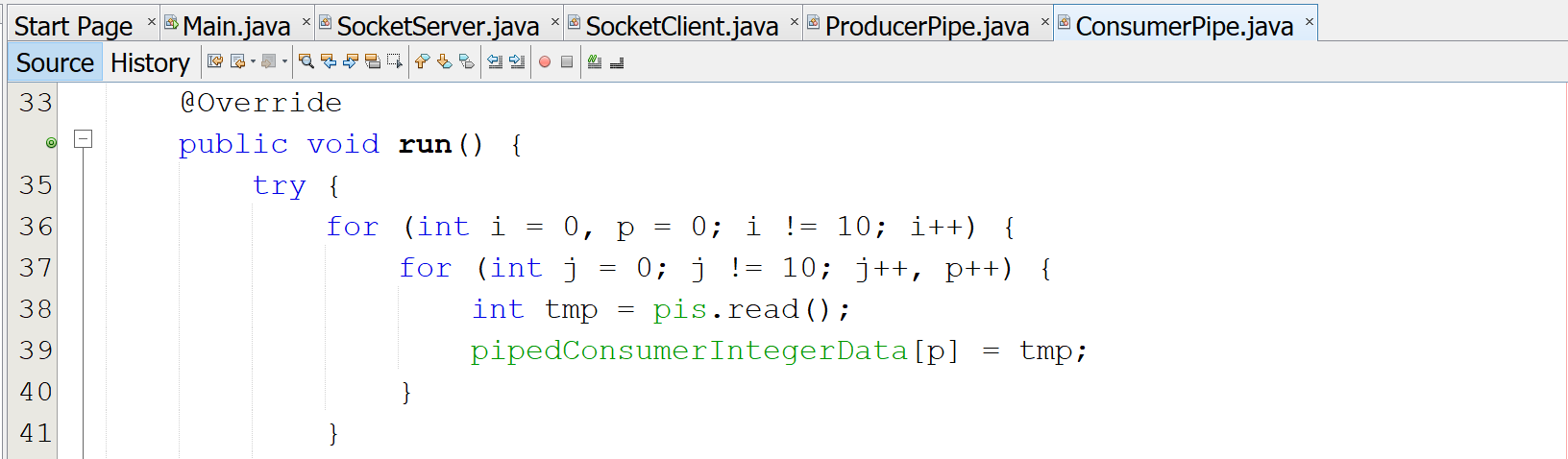
Description of Source Code:

Next is the ConsumerPipe Class, here I am creating the global variables for the class, setting up my PipedInputStream and the array of piped integers that will be filled from the data received from the ProducerPipe Class.



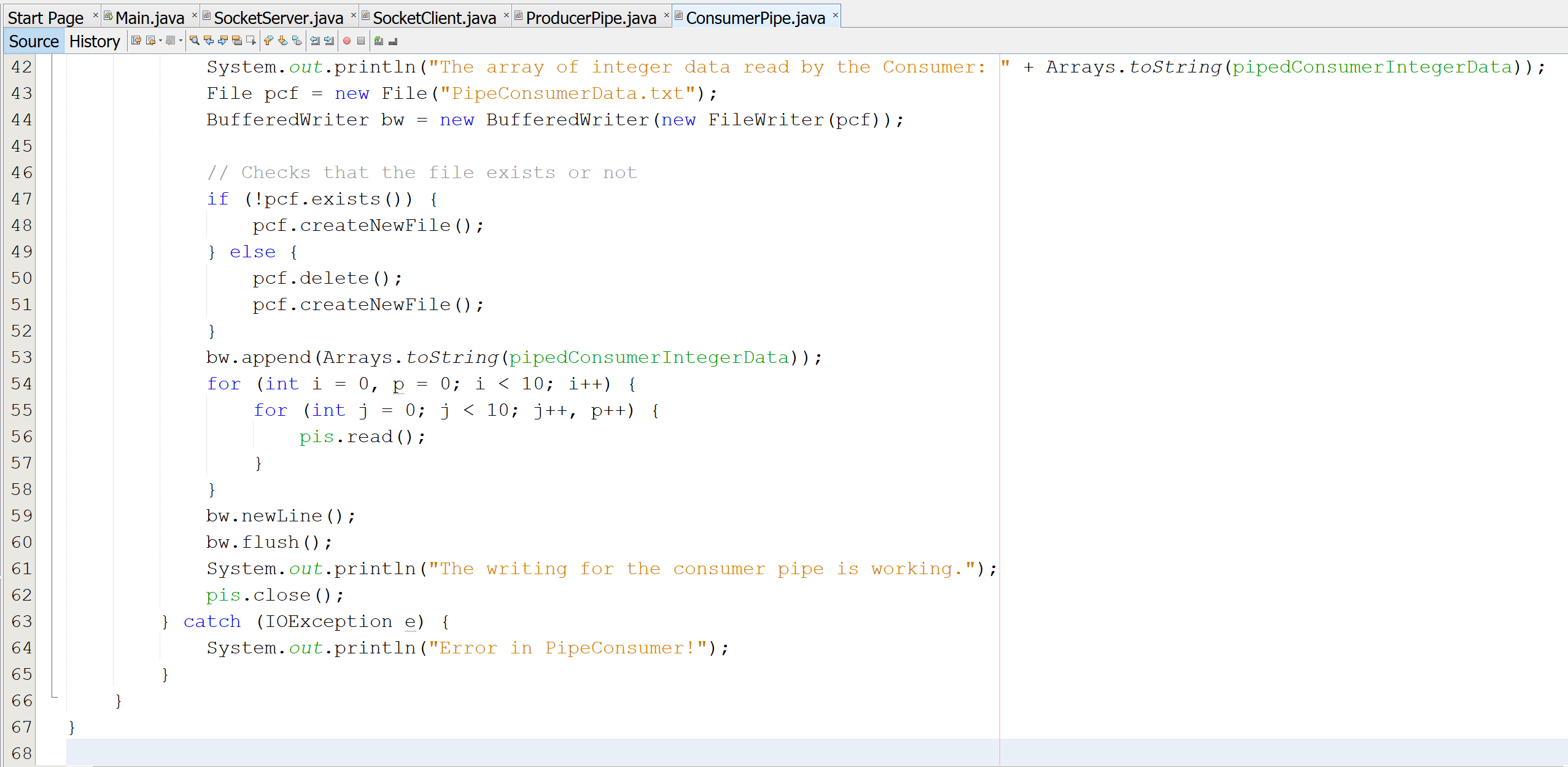
Description of Source Code:

Here I have my constructor for the ConsumerPIpe setting the PipedInputStream and array to there initialized state.



Description of Source Code:

Here is the start of the run method for the class where I am going through each integer that was generated from the ProducerPipe and adding each new element to the Array for the ConsumerPipe Text file.



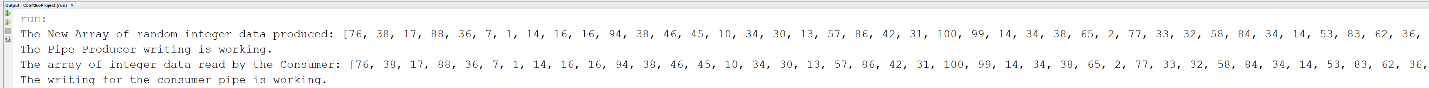
Description of Source Code:

The ending part of the run method for this class deals with creating the text file for this class specifically and then printing out the data to the console. Then the data is written to the new ConsumerPipe text File. Finally the PipedInputStream is closed ending th connection and a sing that the process is complete.

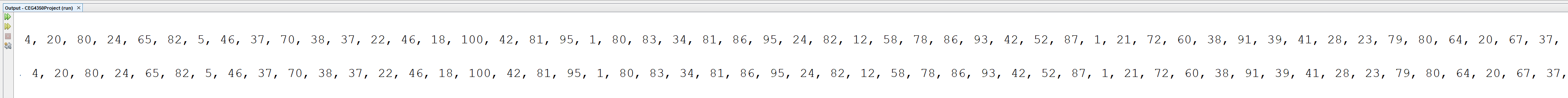
Results for the Pipe Programming:

This is the console output for the Piped section of the program. This display to the console will show the Producer created data that is sent and received by the Consumer. Here I have some print statements to display the data to show and test that all data generated from the Producer and then that data send to the Consumer. With each display here I am able to check each of the array of integers allowing the check of any mismatches from the data transfer.

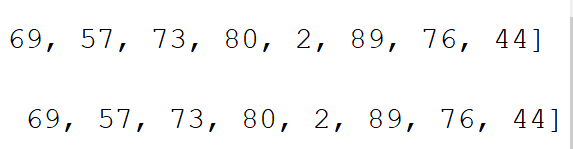
Part 1 Output



Part 2 Output



Part 3 Output



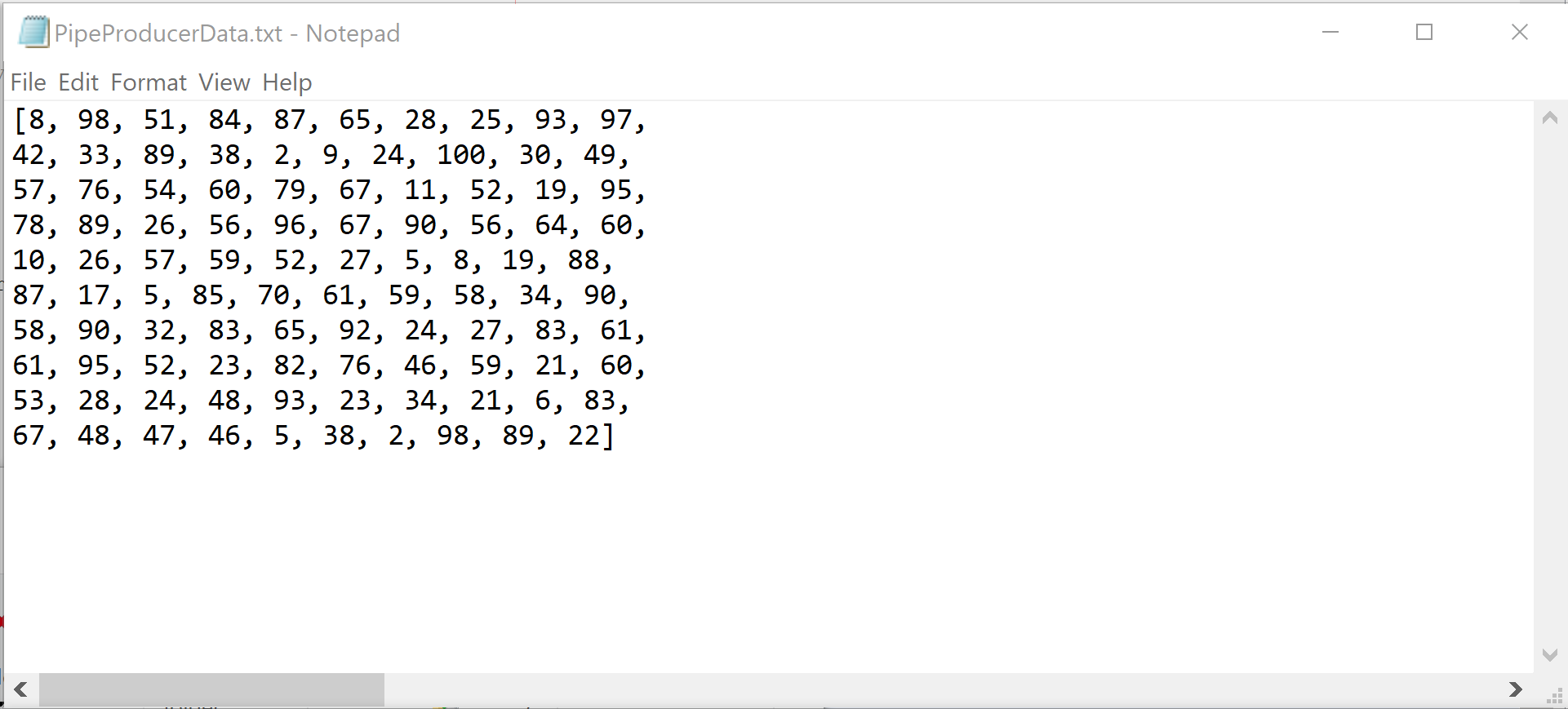
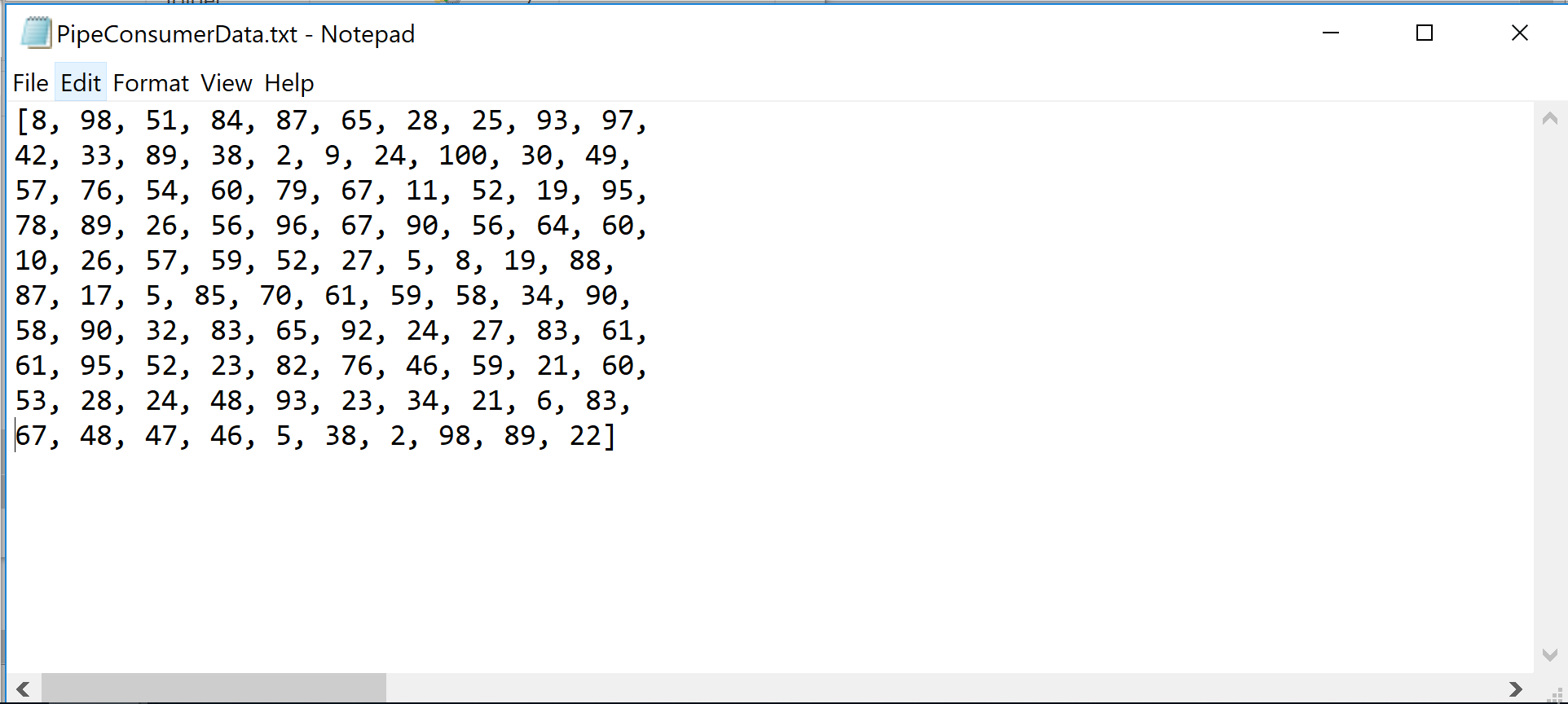
Put them together 1,2,3 and we have all one hundred integers I split them into three parts because I couldn’t get a screenshot of everything together at once and still have it readable.

Text File Images of the Data Transferred:

The text files shown here are the results of my latest test with my project the numbers are different before the output results on the previous images are from an earlier test but the results match the test files every time I run the program. The numbers in the Piped Output were stretching across the screen so I needed to split the length into rows of ten manually so it would all fit on screen at the same time instead of sending multiple images at once.

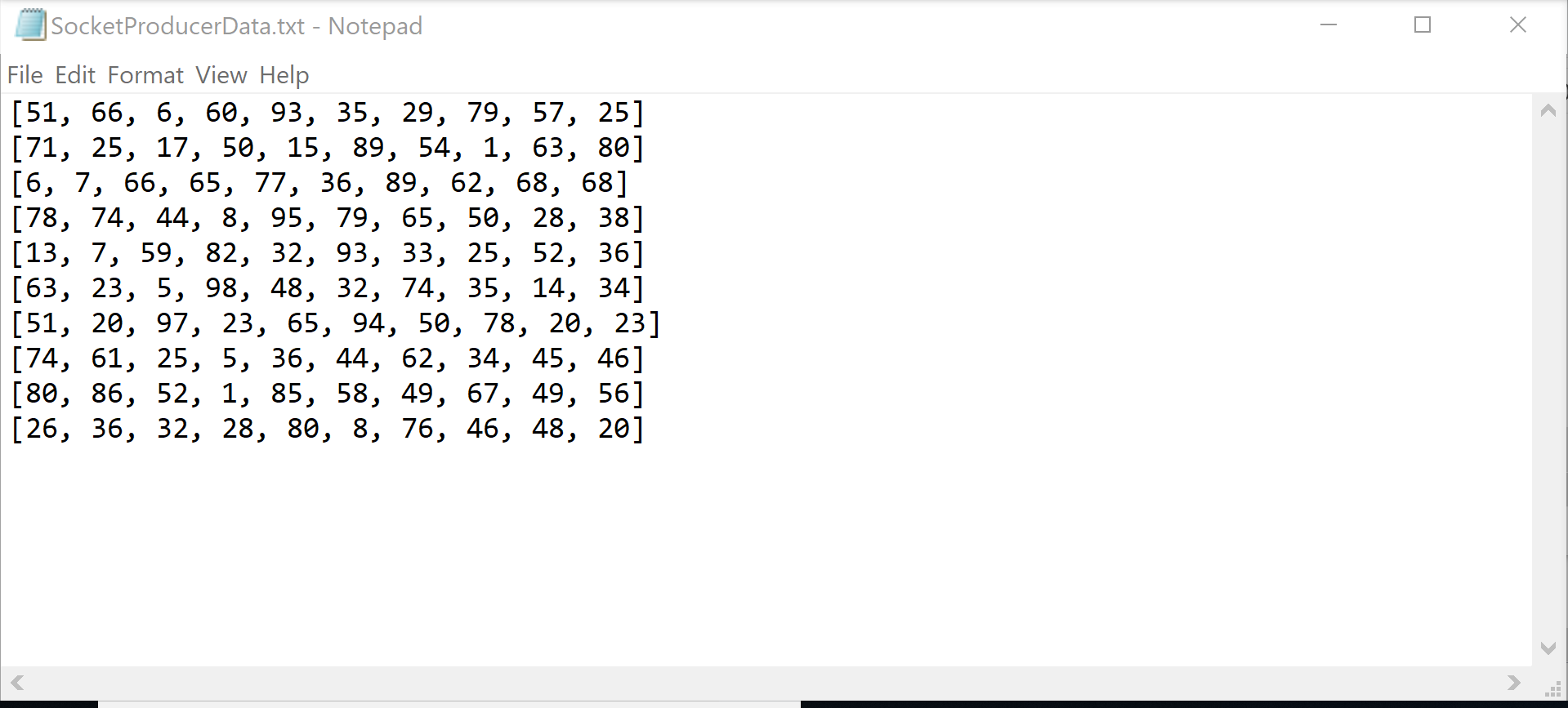
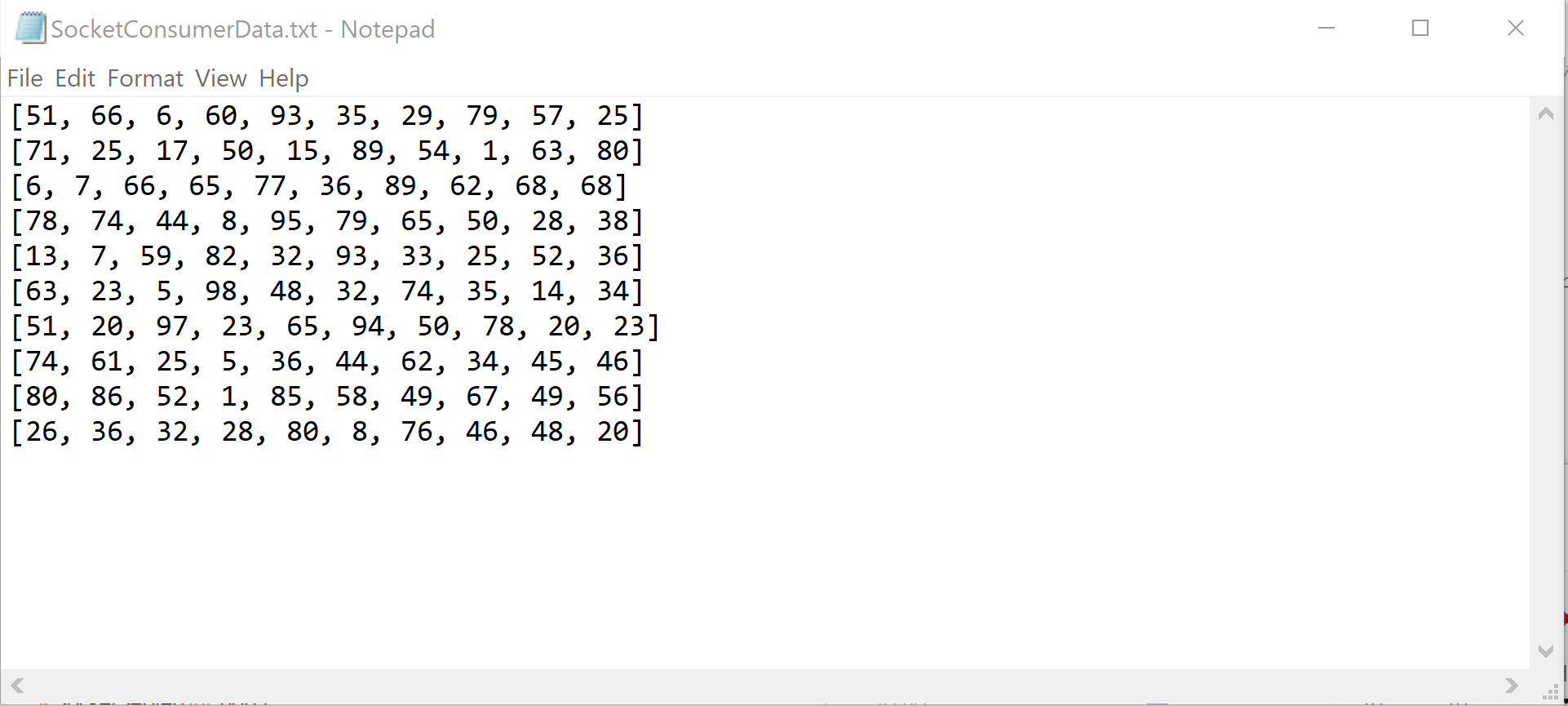
Pipe Files:

Here are the output files for the piped data for both the Piped Producer and Piped Consumer. This data display for the images here are modified to all display in once image shot each. The total is 100 randomly generated integers within the range of 0 to 100.



Socket Files:

Here is the data files for the Socket Producer and Socket Consumer. The data is displayed here in arrays of ten with ten arrays with a total of 100 randomly produced integers. All the integers for each array is randomly generated within the range of 0 to 100.



Overall Discussion:

With the start and finish of this Programming Project I had to jump through a lot of hurtles to get everything finished and turned in. I wasn’t very knowledgeable with how Piping worked or how to implements the additional usage with Sockets to send the data from the Producer to the Consumer. It was all a long struggle but when I completed the project and was able to get all my functions in a working order, I was very proud and excited to see the results of my hard work and all I had learned along the way to complete the assignment. It was a good challenge and one that I think would be good to be repeated in future teachings of this class.

I took a lot of time with this assignment and made meetings with the professor to check on my progress as well as sending emails for questions when I was stuck. I have made sure to comment all the different functions and the variables used as well as explain what each method does and if it returns anything, I tell what is returned and its use.