

TellCalcThread.java

```

1 package tellscopeV4;
2
3 //import libraries
4 import java.net.*;
5
6
7 public class TellCalcThread implements Runnable {
8
9
10 //default constructor
11 public TellCalcThread(Socket socket)
12 {
13     this.s = socket;
14 }
15
16 //socket for incoming data from the client
17 private Socket s;
18
19 //array to store incoming calculation values
20 double[] calcValues = new double[3];
21
22 //counter for input array - used as index to the array
23 int i = 0;
24
25 //private attributes to pass to TellsCalculation object
26 private double focalRatio; //store focal ratio
27 private double lensDiameter; //store lens diameter
28 private double eyePieceFocalLength; //store eyepiece focal length
29
30 private Scanner in;
31
32 private boolean type;
33
34 /*
35 //variables to store calculated results
36 private static double focalLength;
37 private static double tubeLength;
38 private static double distToSecond;
39 private static double secondarySizeMinor;
40 private static double secondarySizeMajor;
41 private static double minMagnitude;
42 private static double minResolution;
43 private static double maxVisibleMagnification;
44 private static double minVisibleMagnification;
45 private static double eyePieceMagnification;
46 */
47
48 public void run() {
49
50     String client = s.getInetAddress().toString();
51     TellServerGui04.consoleView.append("\nConnected to " + client);
52     TellServerGui04.consoleView.append("\nRunning Calculations");
53     System.out.println("Connected to " + client);
54
55     //try creating a new scanner
56     try
57     {
58         //create new scanner object and set it to get input stream from socket
59         in = new Scanner(s.getInputStream());
60
61
62         // loop to test if the socket input reads "calculate"
63         // if it does not - add values to array for processing later
64         // if it does - sort values, perform calculations and store to TellServer calcResults array
65         // ready for sending to client
66         while(true)
67         {
68             //store next line in "input" string
69             String input = in.nextLine();
70             //check for equality with string "calculate"
71             if(input.equalsIgnoreCase("calculate"))
72             {
73                 //add input values to correct attributes for calculations
74                 focalRatio = calcValues[0];
75                 lensDiameter = calcValues[1];
76                 eyePieceFocalLength = calcValues[2];
77
78                 /*
79                 // write inputs to input text fields */
80
81                 TellServerGui04.setInput(String.format("%.2f", lensDiameter),String.format(
82                     "%.2f", focalRatio), String.format("%.2f", eyePieceFocalLength));
83
84
85                 /* moved below for test
86                 //create new TellsCalculations object and pass user input attributes
87                 TelescopeReflect testCalcs = new TelescopeReflect(focalRatio, lensDiameter, eyePieceFocalLength);
88                 */
89
90
91
92
93
94
95
96

```

TellCalcThread.java

```

97
98      /* check the telescope type (reflect == true, refract == false) */
99      if(type == true)
100      {
101          //create new TellsCalculations object and pass user input attributes
102          TelescopeReflect reflecting = new TelescopeReflect(focalRatio, lensDiameter,
eyePieceFocallLength);
103
104          //call calculation methods to get values
105          //convert to string with 2 decimal places
106          //store in TellServer.calcResults array
107          TellServerGui04.calcResults[0] = String.format("%.2f", reflecting.calcFocallLength());
108          TellServerGui04.calcResults[1] = String.format("%.2f", reflecting.calcTubeLength());
109          TellServerGui04.calcResults[2] = String.format("%.2f", reflecting.calcDistToSecond());
110          TellServerGui04.calcResults[3] = String.format("%.2f",
reflecting.calcSecondarySizeMinor());
111          TellServerGui04.calcResults[4] = String.format("%.2f",
reflecting.calcSecondarySizeMajor());
112          TellServerGui04.calcResults[5] = String.format("%.2f", reflecting.calcMinMagnitude());
113          TellServerGui04.calcResults[6] = String.format("%.2f", reflecting.calcMinResolution());
114          TellServerGui04.calcResults[7] = String.format("%.2f", reflecting.calcMaxVisibleMag());
115          TellServerGui04.calcResults[8] = String.format("%.2f", reflecting.calcMinVisibleMag());
116          TellServerGui04.calcResults[9] = String.format("%.2f",
reflecting.calcEyepieceMagnification());
117
118
119          //once array is populated - add string "result" to inform client to display the results
120          TellServerGui04.calcResults[10] = "result";
121
122          /* place results in server results area */
123          TellServerGui04.setResults();
124
125          //once calculations are done and stored - create a new thread to pass results to client
126          Thread sendResults = new Thread(new SendResultsThread(s));
127          sendResults.start();
128
129          i = 0;
130          //DO I NEED A BREAK HERE???????
131          continue;
132      }
133
134      else if(type == false)
135      {
136          //create new TellsCalculations object and pass user input attributes
137          TelescopeRefract refracting = new TelescopeRefract(focalRatio, lensDiameter,
eyePieceFocallLength);
138
139          //call calculation methods to get values
140          //convert to string with 2 decimal places
141          //store in TellServer.calcResults array
142          TellServerGui04.calcResults[0] = String.format("%.2f",
refracting.calcFocallLength());
143          TellServerGui04.calcResults[1] = String.format("%.2f",
refracting.calcTubeLength());
144          TellServerGui04.calcResults[2] = "Not Required";
145          TellServerGui04.calcResults[3] = "Not Required";
146          TellServerGui04.calcResults[4] = "Not Required";
147          TellServerGui04.calcResults[5] = String.format("%.2f",
refracting.calcMinMagnitude());
148          TellServerGui04.calcResults[6] = String.format("%.2f",
refracting.calcMinResolution());
149          TellServerGui04.calcResults[7] = String.format("%.2f",
refracting.calcMaxVisibleMag());
150          TellServerGui04.calcResults[8] = String.format("%.2f",
refracting.calcMinVisibleMag());
151          TellServerGui04.calcResults[9] = String.format("%.2f",
refracting.calcEyepieceMagnification());
152
153
154          //once array is populated - add string "result" to inform client to display the
results
155          TellServerGui04.calcResults[10] = "result";
156
157          //TellServer.printResults();
158          //TellServer.sendResultsToClient();
159
160          /* place results in server results area */
161          TellServerGui04.setResults();
162
163          //once calculations are done and stored - create a new thread to pass results to
client
164          Thread sendResults = new Thread(new SendResultsThread(s));
165          sendResults.start();
166
167          i = 0;
168          //DO I NEED A BREAK HERE???????
169          continue;
170      }
171
172      }
173
174
175      else if(input.equalsIgnoreCase("reflect"))
176      {

```

```

177         type = true;
178     }
179
180     else if(input.equalsIgnoreCase("refract"))
181     {
182         type = false;
183     }
184
185
186
187     //if input != "calculate"
188     else
189     {
190
191
192         //try parse the input to a double
193         try{
194
195             //store in calcValues array for sorting later
196             calcValues[i] = Double.parseDouble(input);
197             //increment counter for array index
198             i++;
199         }
200         //catch exceptions
201         catch (NumberFormatException nfe)
202         {
203             //if exception caught - print "Failed" and exception to console
204             System.out.println("Failed: " + nfe.toString());
205             //exit
206             System.exit(1);
207         } //end nfe exception catch
208
209     } //end else
210
211 } //end while loop
212
213 } //end top try
214 catch (Exception e)
215 {
216     //if exception caught - do nothing!
217 }
218
219 in.close();
220
221
222 } //end run
223
224 } //end class
225

```