So with this design we have one program that creates the data and one that will read the data adding on header for each of the values in the table. So to begin we have the datacreate which starts with producing a three run time parameter if under or over the program wont produce the changed file. The three parameters are the source file, destination file, and the records in the data file. After that we got the set of arrays for each of the different values. That will then go through a try catch statement to change the file and then make the new file. Then I use the substring to produce the layout format for each outputs. So in order to get the state id I produced the substring from the indexes 0,2. This goes for the rest of the substrings that were provided in this code. Using the file formater to read each line of the file. Used the I/O buffered reader and file reader in order to read the file and then used file writer and buffered writer to print out the changed text file. Once we have the changed data file we then use the read text to implement the headers to this file of 56 states. I also used the printf to allow for a more concise position on the data. Allowed me to line up the information a lot better than using println and having a bunch of spaces. Took a lot less time than using println to produce the spacing. Since System.out allows you to use printf this made things a lot more simple to produce the rest. Then I used split string method to split them into substrings with whitespace. Then I used the format method to allow me to space out the integers within the set of data in the text file. Since the percent child poverty had decimal points I used the f layout symbol to then produce the layout needed.

```java
/**
 * produces changed a new text file that has the changed values for the population sizes.
 */

package module10;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class DataCreate {
   public static void main(String[] args)
   {
      // allows for the three run time parameters to be read.
      if(args.length != 3)
      {
         System.out.println("File: SmallAreaIncomePovertyEstDatav2.txt 
SmallAreaIncomePovertyEstDataChangev2.txt 13486");
         System.exit(1);
      }
      int[] state = new int[13487];

      int[] population = new int[13487];

      int[] childPopulation = new int[13487];

      int[] childPovertyPopulation = new int[13487];

      int changeState = 0;

      for (int i = 0; i < 13487; i++)
      {
         state[i] = 0;
         population[i] = 0;
         childPopulation[i] = 0;
         childPovertyPopulation[i] = 0;
      }
      try {
         changeState = changeFile(state, population, childPopulation, childPovertyPopulation);
         makeNewFile(args[1], changeState, state, population, childPopulation,
childPovertyPopulation);
         } catch (FileNotFoundException except) {
            System.out.println(except.getMessage());
            } catch (IOException except) {
               System.out.println(except.getMessage());

            }

   }
   /**
    *
    * @param state array for the stateid
    * @param population array for population
    * @param childPopulation array for child population
    * @param childPovertyPopulation array for child poverty population
    * @return numState
    * @throws FileNotFoundException
    */
   private static int changeFile(int[] state, int[] population, int[] childPopulation, int[]
childPovertyPopulation) throws FileNotFoundException

   {
      String fileName = "SmallAreaIncomePovertyEstDatav2.txt";
      BufferedReader br = new BufferedReader(new FileReader(fileName));
      String line;
      String num;
      int stateCode = 0;
      int numState = 0;
      try
      {
         while ((line = br.readLine()) != null)
         {
            // using the Layout text to produce these outputs
            //gets the state code
            num = line.substring(0, 2).trim();
            stateCode = Integer.parseInt(num);
            state[stateCode] = stateCode;

            //gets total for the population
            num = line.substring(82, 90).trim();
            population[stateCode] += Integer.parseInt(num);

            //gets children population
            num = line.substring(91, 99).trim();
            childPopulation[stateCode] += Integer.parseInt(num);

            //gets child poverty population
            num = line.substring(100, 108).trim();
            childPovertyPopulation[stateCode] += Integer.parseInt(num);

            if(stateCode > numState)
            {
               numState = stateCode;

            }

         }
```

```java
            br.close();
            } catch (IOException e) {
               System.out.println("Exception occured." + e.getMessage());

            }
         return numState;

      }
      /**
       *
       * @param fileName string for the new filename
       * @param numState is the value produced by the changeFile
       * @param state number of state
       * @param population size
       * @param childPopulation size
       * @param childPovertyPopulation size
       * @throws IOException
       */
      private static void makeNewFile(String fileName, int numState, int[] state, int[] population,
   int[] childPopulation, int[] childPovertyPopulation) throws IOException
      {
         File f = new File(fileName);
         BufferedWriter bw = new BufferedWriter(new FileWriter(f));
         double percent;
         String line;
         for(int i = 1; i <= numState; i++ )
          {
            if(state[i] == 0) continue; //no data for the state code
            //calculate percentages
            percent = childPovertyPopulation[i] * 100.0 / childPopulation[i];
            line = String.format("%02d %15d %15d %15d %10.2f", state[i], population[i],
   childPopulation[i], childPovertyPopulation[i], percent);
            bw.write(line + "\n");

          }
         bw.close();
         System.out.println("Report generated in file: " + f.getAbsolutePath());

      }

   }

   /**
    * This will read the changed text file that was produced by the
    * other programing when inputing two run time parameters that are the
    * txt file name and number of records.
    */
   package module10;

   import java.io.BufferedReader;
   import java.io.File;
   import java.io.FileReader;

   public class ReadText {
      public static void main(String[] args) {
          if(args.length != 2)
          {
              System.out.println("SmallAreaIncomePovertyEstDataChangev2.txt 56");
              System.exit(1);
          }
```

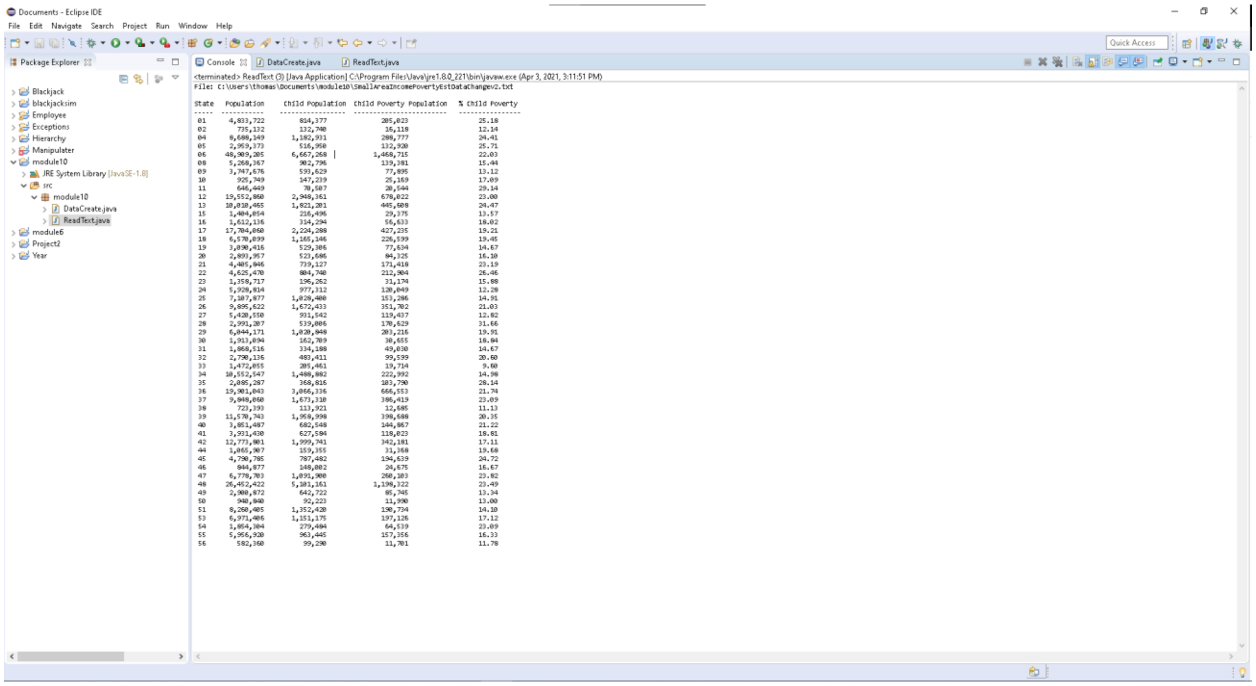
```
        BufferedReader newFile;
        File changedFile = new File(args[0]);
        System.out.println("File: " + changedFile.getAbsolutePath() + "\n");
        int state;
        int population;
        int childPopulations;
        int childPovertyPopulations;
        double percentChildPoverty;
        String line;
        try
        {
           newFile = new BufferedReader(new FileReader(changedFile));
           System.out.printf("State %12s %20s %25s %17s\n", "Population", "Child Population",
"Child Poverty Population", "% Child Poverty");
           System.out.printf("----- %12s %20s %25s %16s\n",
                    "-----------", "-----------------", "------------------------", "  ------------
----");
           while((line = newFile.readLine()) != null)
           {
              String[] value = line.split("( )+");
              state = Integer.parseInt(value[0]);
              population = Integer.parseInt(value[1]);
              childPopulations = Integer.parseInt(value[2]);
              childPovertyPopulations = Integer.parseInt(value[3]);
              percentChildPoverty = Double.parseDouble(value[4]);
              line = String.format(" %02d", state);
              line += String.format(" %,13d", population);
              line += String.format(" %,15d", childPopulations);
              line += String.format(" %,20d", childPovertyPopulations);
              line += String.format(" %22.2f", percentChildPoverty);
              System.out.println(line);

           }

           newFile.close();
           }catch (Exception except){
              System.out.println(except.getMessage());
        }


    }

}
```



```
<terminated> ReadText (3) [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Apr 3, 2021, 3:11:51 PM)
File: C:\Users\thomas\Documents\module10\SmallAreaIncomePovertyEstDataChangev2.txt

State   Population     Child Population  Child Poverty Population   % Child Poverty
-----  ------------   -----------------  ------------------------  ----------------
 01      4,833,722          814,377              205,023                 25.18
 02        735,132          132,740               16,118                 12.14
 04      8,600,149        1,182,931              288,777                 24.41
 05      2,959,373          516,950              132,920                 25.71
 06     48,909,205        6,667,268            1,468,715                 22.03
 08      5,268,367          902,796              139,381                 15.44
 09      3,747,676          593,629               77,895                 13.12
 10        925,749          147,239               25,169                 17.09
 11        646,449           70,507               20,544                 29.14
 12     19,552,860        2,948,361              678,022                 23.00
 13     10,010,465        1,821,201              445,608                 24.47
 15      1,404,054          216,496               29,375                 13.57
 16      1,612,136          314,294               56,633                 18.02
 17     17,704,060        2,224,288              427,235                 19.21
 18      6,570,099        1,165,146              226,599                 19.45
 19      3,090,416          529,306               77,634                 14.67
 20      2,893,957          523,606               84,325                 16.10
 21      4,405,846          739,127              171,418                 23.19
 22      4,625,470          804,740              212,904                 26.46
 23      1,358,717          196,262               31,174                 15.88
 24      5,928,814          977,312              120,049                 12.28
 25      7,107,877        1,028,400              153,206                 14.91
 26      9,895,622        1,672,433              351,702                 21.03
 27      5,420,550          931,542              119,437                 12.82
 28      2,991,207          539,006              170,629                 31.66
 29      6,044,171        1,020,048              203,216                 19.91
 30      1,913,094          162,709               30,655                 18.84
 31      1,868,516          334,188               49,030                 14.67
 32      2,790,136          483,411               99,599                 20.60
 33      1,472,055          205,461               19,714                  9.60
 34     10,552,547        1,488,882              222,992                 14.98
 35      2,085,287          368,816              103,790                 28.14
 36     19,901,043        3,066,336              666,553                 21.74
 37      9,848,060        1,673,310              386,419                 23.09
 38        723,393          113,921               12,605                 11.13
 39     11,570,743        1,958,998              398,688                 20.35
 40      3,851,487          682,548              144,867                 21.22
 41      3,931,430          627,584              118,023                 18.81
 42     12,773,801        1,999,741              342,181                 17.11
 44      1,065,907          159,355               31,360                 19.68
 45      4,790,785          787,482              194,639                 24.72
 46        844,877          148,882               24,675                 16.67
 47      6,778,703        1,091,900              260,103                 23.82
 48     26,452,422        5,101,161            1,198,322                 23.49
 49      2,900,872          642,722               85,745                 13.34
 50        940,840           92,223               11,990                 13.00
 51      8,260,405        1,352,420              190,734                 14.10
 53      6,971,406        1,151,175              197,126                 17.12
 54      1,854,304          279,404               64,539                 23.09
 55      5,956,920          963,445              157,356                 16.33
 56        582,360           99,290               11,701                 11.78
```