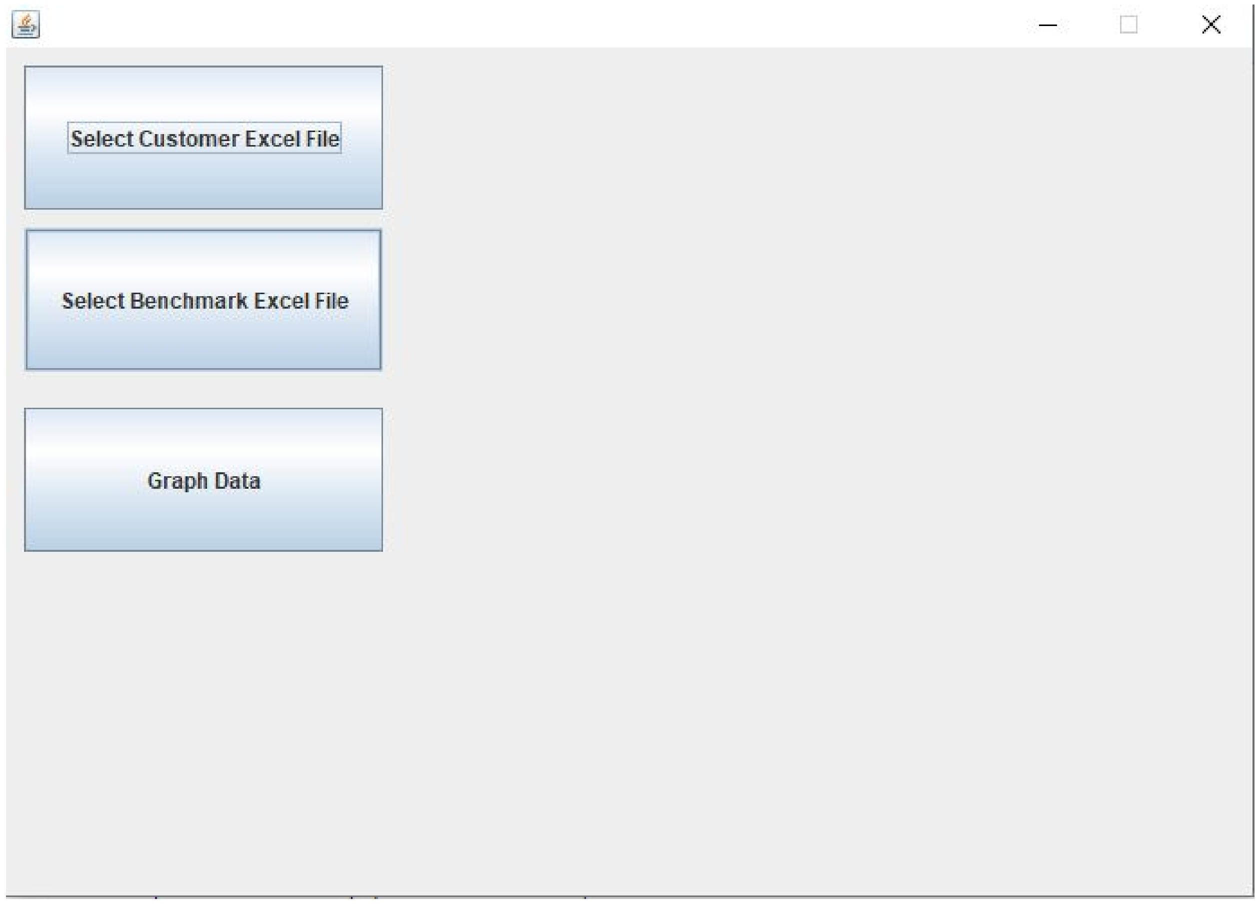
Appendices include:

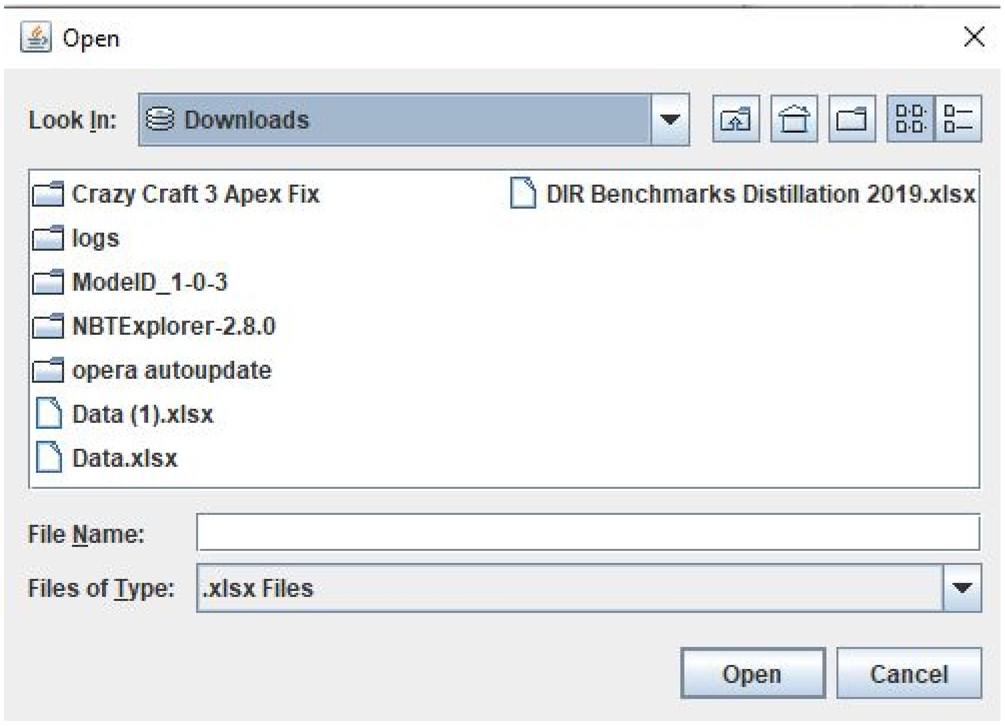
A. Screenshots of user interface.
B. Source code.

User Interface:

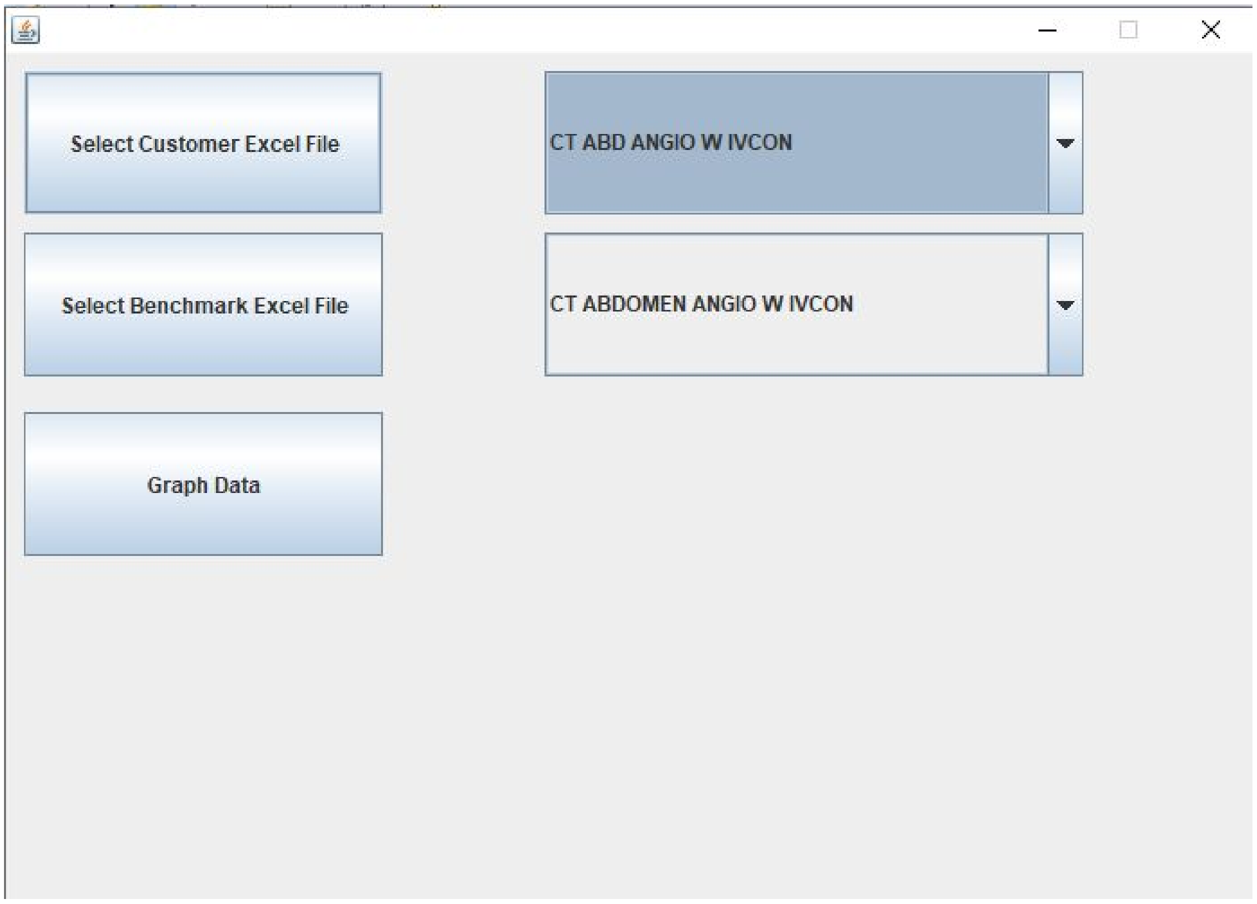
Start screen:



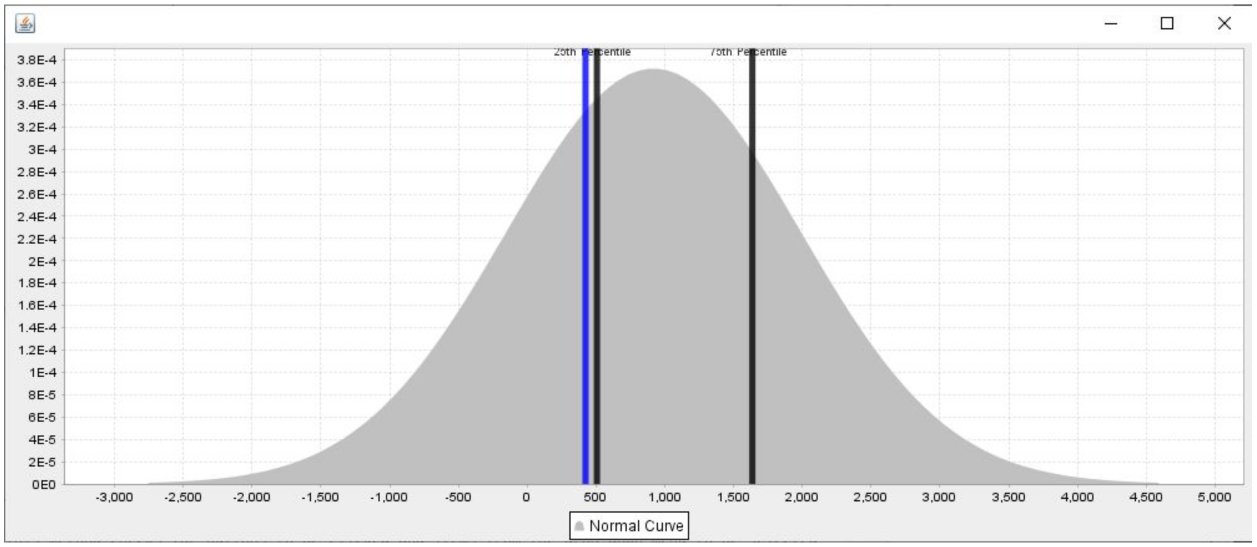
File selection:



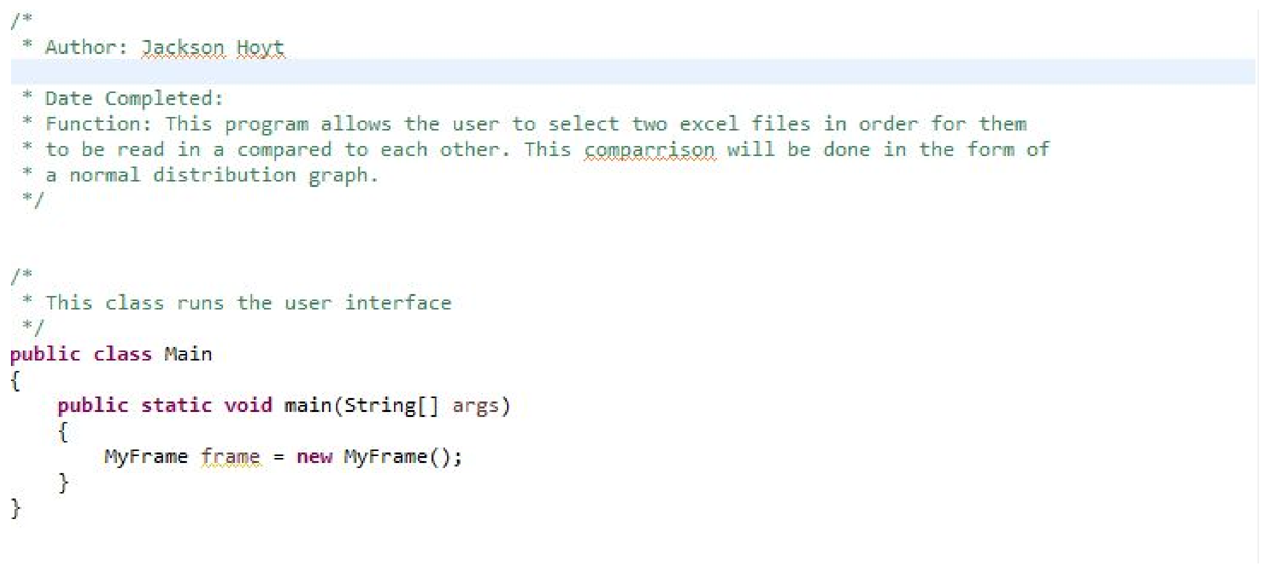
Start screen with drop down menu:



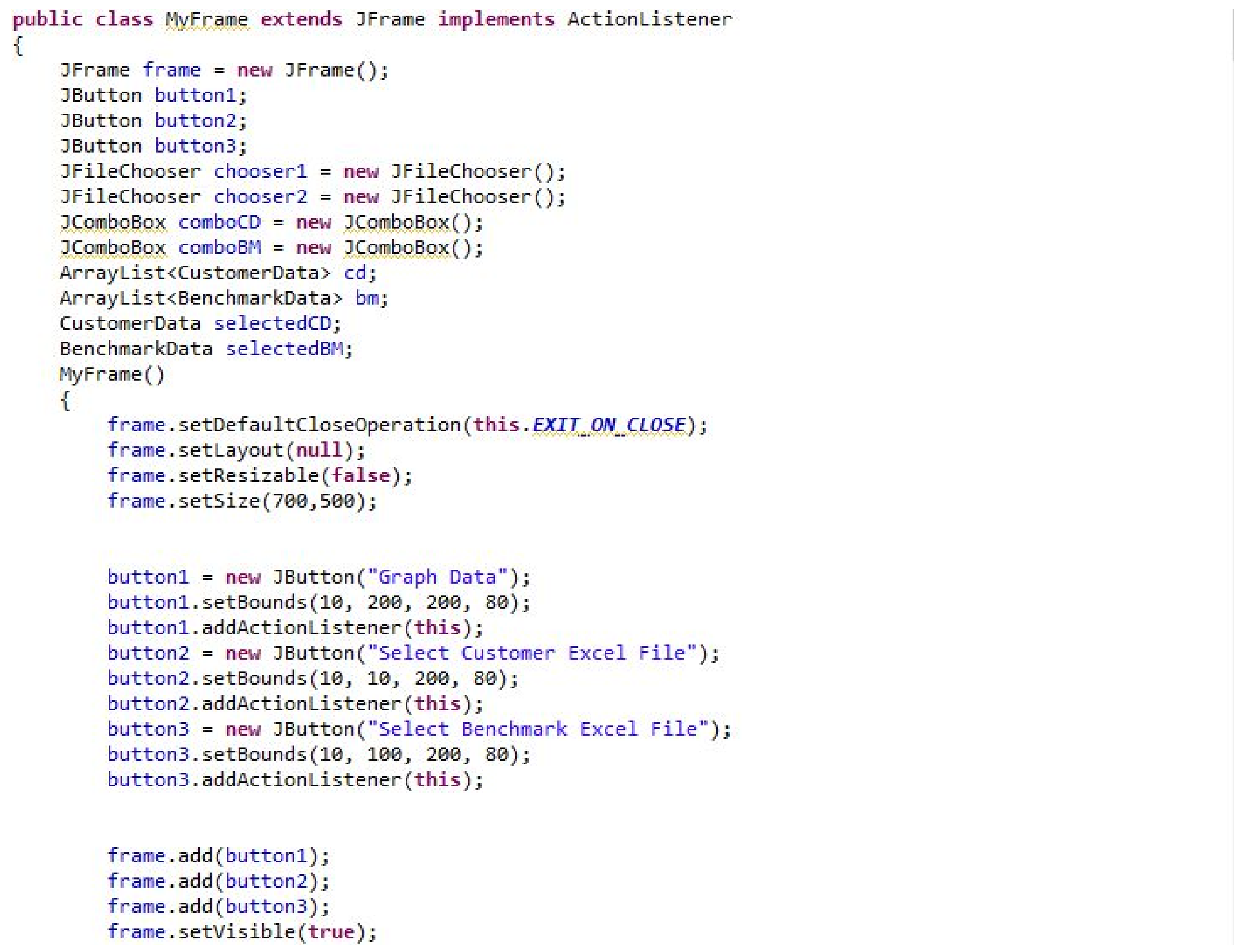
Graph:



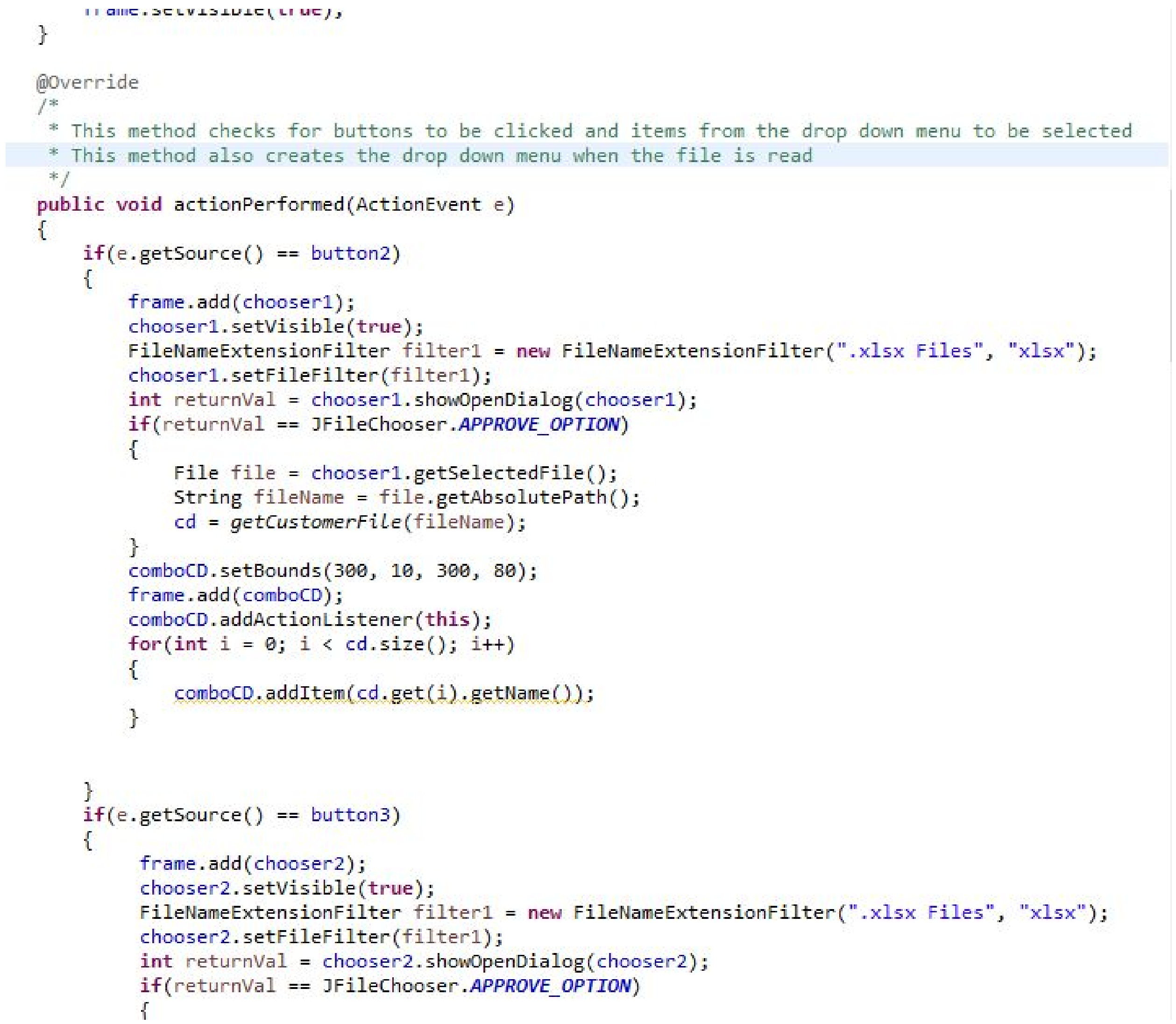
Source Code:



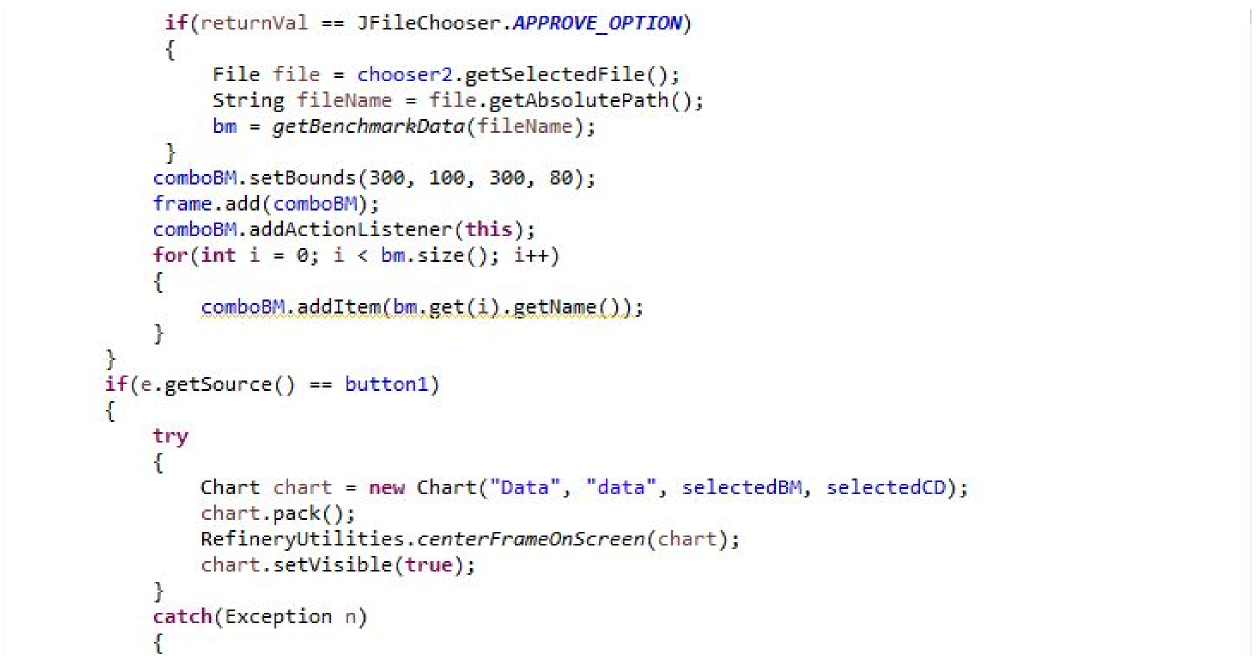
```java
/*
 * Author: Jackson Hoyt

 * Date Completed:
 * Function: This program allows the user to select two excel files in order for them
 * to be read in a compared to each other. This comparrison will be done in the form of
 * a normal distribution graph.
 */


/*
 * This class runs the user interface
 */
public class Main
{
    public static void main(String[] args)
    {
        MyFrame frame = new MyFrame();
    }
}
```



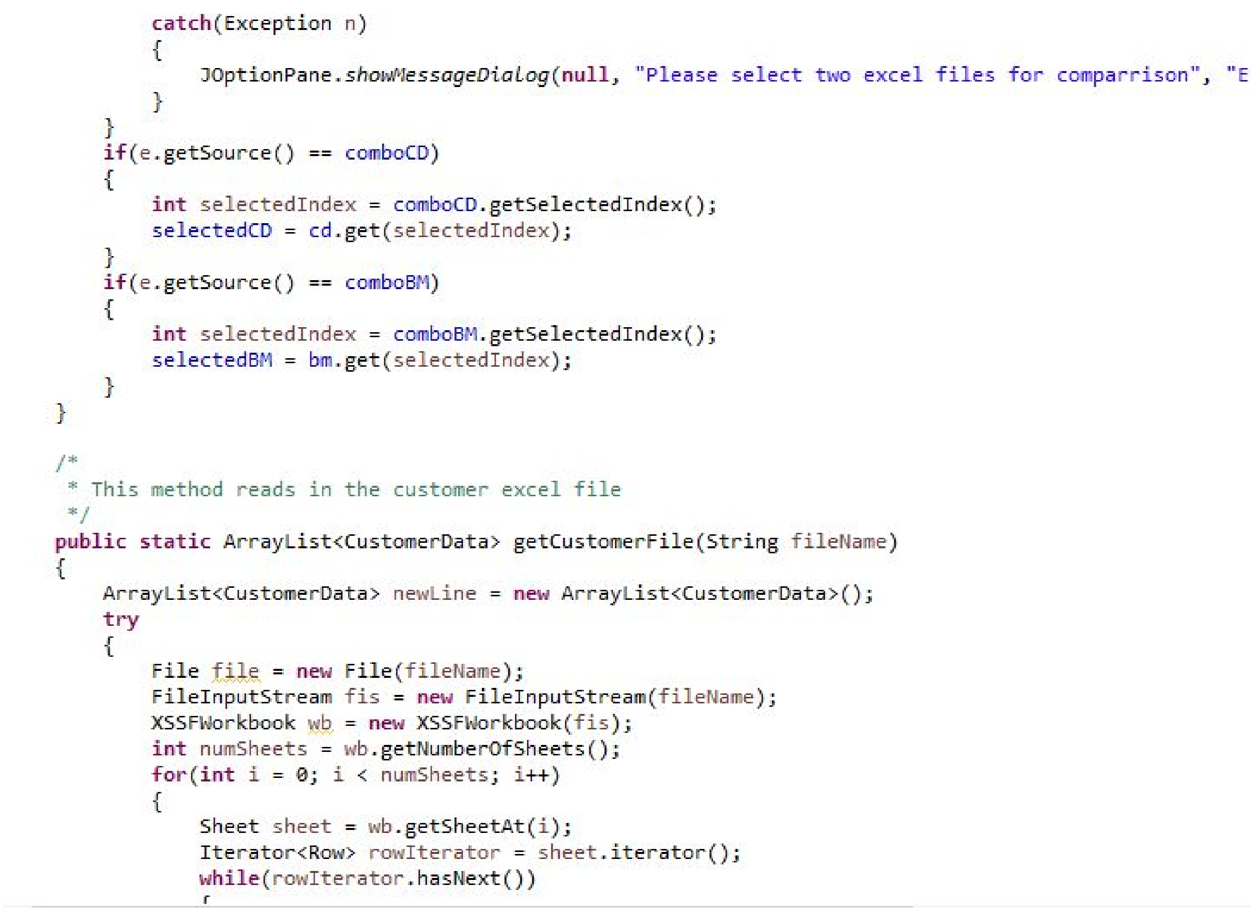
```java
public class MyFrame extends JFrame implements ActionListener
{
    JFrame frame = new JFrame();
    JButton button1;
    JButton button2;
    JButton button3;
    JFileChooser chooser1 = new JFileChooser();
    JFileChooser chooser2 = new JFileChooser();
    JComboBox comboCD = new JComboBox();
    JComboBox comboBM = new JComboBox();
    ArrayList<CustomerData> cd;
    ArrayList<BenchmarkData> bm;
    CustomerData selectedCD;
    BenchmarkData selectedBM;
    MyFrame()
    {
        frame.setDefaultCloseOperation(this.EXIT_ON_CLOSE);
        frame.setLayout(null);
        frame.setResizable(false);
        frame.setSize(700,500);


        button1 = new JButton("Graph Data");
        button1.setBounds(10, 200, 200, 80);
        button1.addActionListener(this);
        button2 = new JButton("Select Customer Excel File");
        button2.setBounds(10, 10, 200, 80);
        button2.addActionListener(this);
        button3 = new JButton("Select Benchmark Excel File");
        button3.setBounds(10, 100, 200, 80);
        button3.addActionListener(this);


        frame.add(button1);
        frame.add(button2);
        frame.add(button3);
        frame.setVisible(true);
```



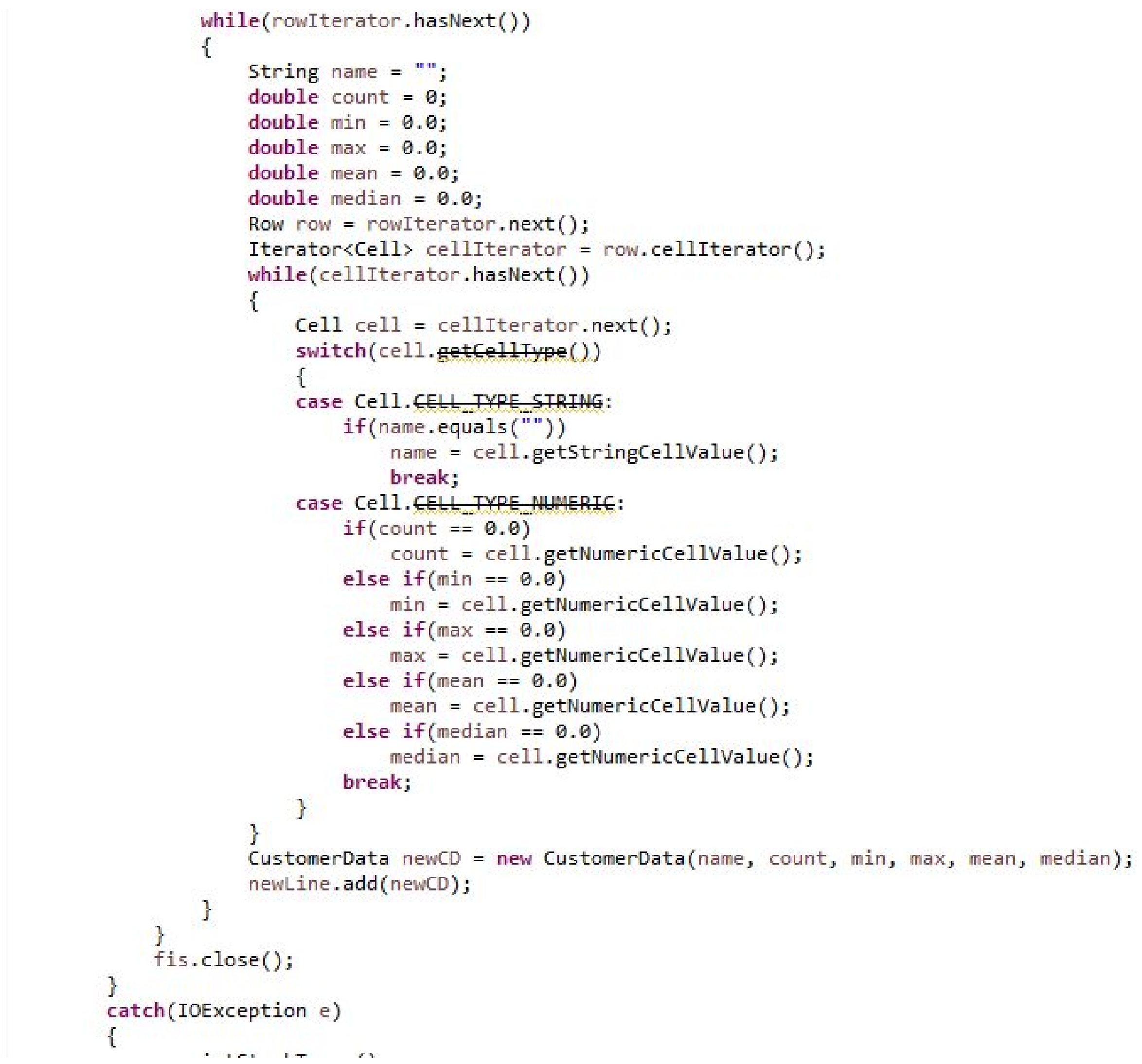
```
        frame.setVisible(true);
    }

    @Override
    /*
     * This method checks for buttons to be clicked and items from the drop down menu to be selected
     * This method also creates the drop down menu when the file is read
     */
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == button2)
        {
            frame.add(chooser1);
            chooser1.setVisible(true);
            FileNameExtensionFilter filter1 = new FileNameExtensionFilter(".xlsx Files", "xlsx");
            chooser1.setFileFilter(filter1);
            int returnVal = chooser1.showOpenDialog(chooser1);
            if(returnVal == JFileChooser.APPROVE_OPTION)
            {
                File file = chooser1.getSelectedFile();
                String fileName = file.getAbsolutePath();
                cd = getCustomerFile(fileName);
            }
            comboCD.setBounds(300, 10, 300, 80);
            frame.add(comboCD);
            comboCD.addActionListener(this);
            for(int i = 0; i < cd.size(); i++)
            {
                comboCD.addItem(cd.get(i).getName());
            }


        }
        if(e.getSource() == button3)
        {
             frame.add(chooser2);
             chooser2.setVisible(true);
             FileNameExtensionFilter filter1 = new FileNameExtensionFilter(".xlsx Files", "xlsx");
             chooser2.setFileFilter(filter1);
             int returnVal = chooser2.showOpenDialog(chooser2);
             if(returnVal == JFileChooser.APPROVE_OPTION)
             {
```



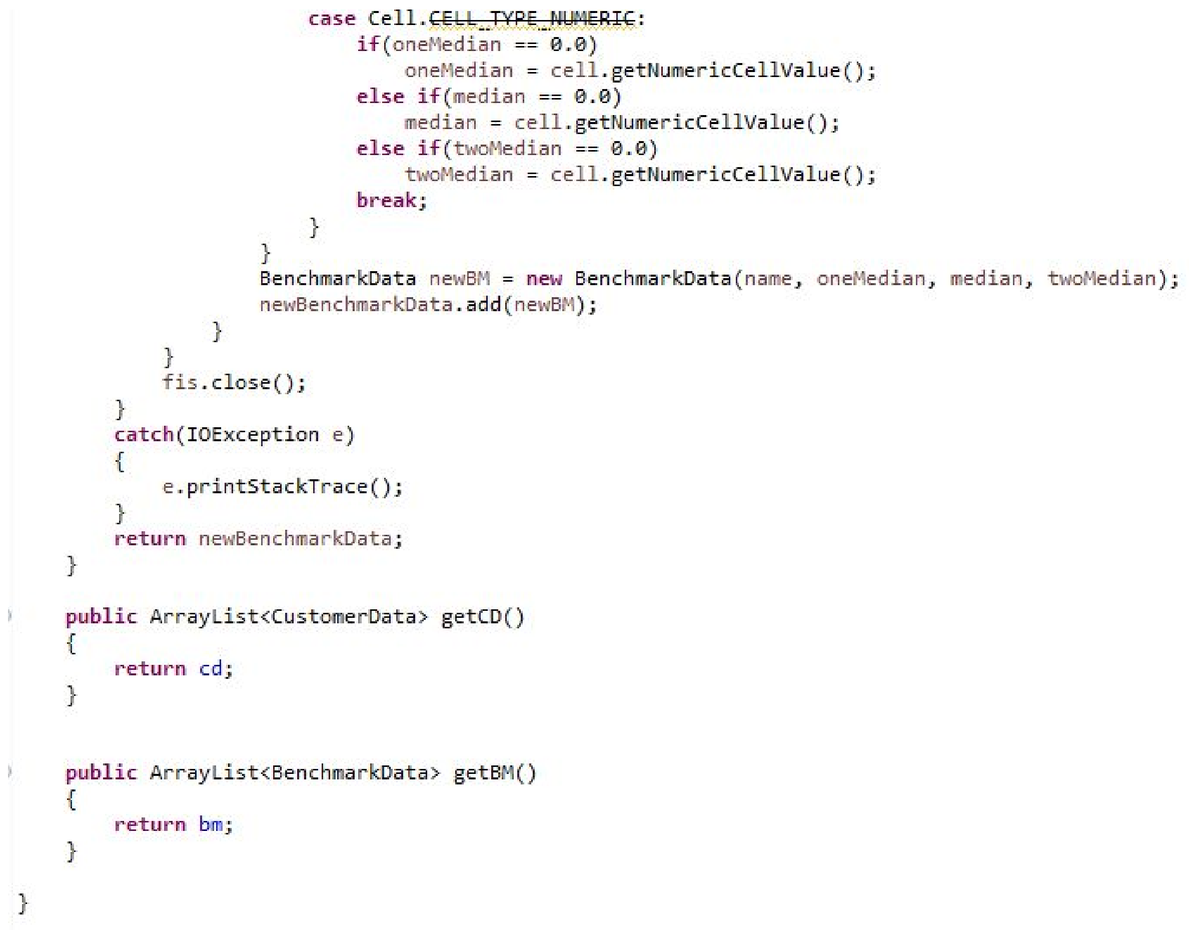
```
             if(returnVal == JFileChooser.APPROVE_OPTION)
             {
                 File file = chooser2.getSelectedFile();
                 String fileName = file.getAbsolutePath();
                 bm = getBenchmarkData(fileName);
             }
            comboBM.setBounds(300, 100, 300, 80);
            frame.add(comboBM);
            comboBM.addActionListener(this);
            for(int i = 0; i < bm.size(); i++)
            {
                comboBM.addItem(bm.get(i).getName());
            }
        }
        if(e.getSource() == button1)
        {
            try
            {
                Chart chart = new Chart("Data", "data", selectedBM, selectedCD);
                chart.pack();
                RefineryUtilities.centerFrameOnScreen(chart);
                chart.setVisible(true);
            }
            catch(Exception n)
            {
```



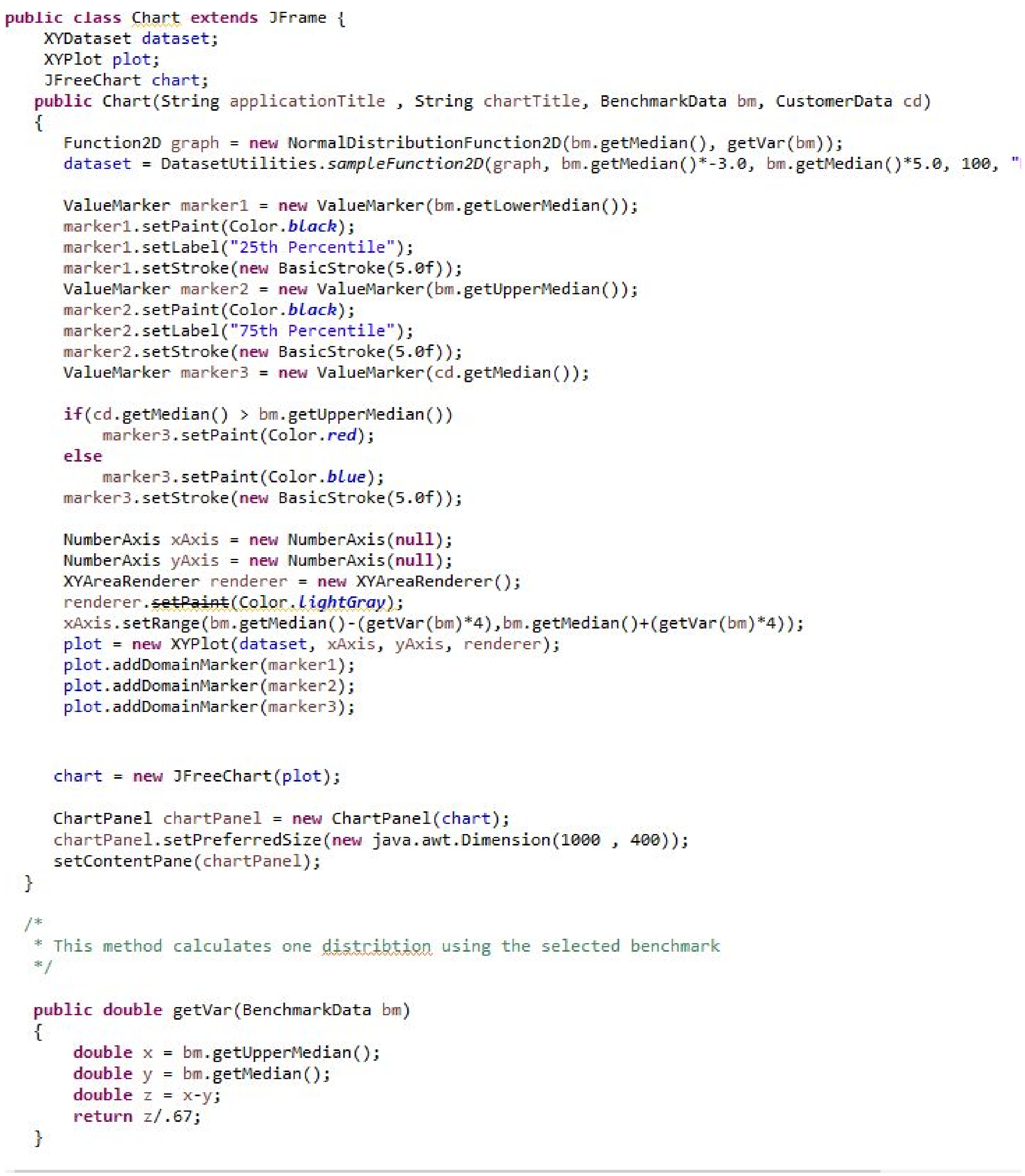
```
            catch(Exception n)
            {
                JOptionPane.showMessageDialog(null, "Please select two excel files for comparrison", "E
            }
        }
        if(e.getSource() == comboCD)
        {
            int selectedIndex = comboCD.getSelectedIndex();
            selectedCD = cd.get(selectedIndex);
        }
        if(e.getSource() == comboBM)
        {
            int selectedIndex = comboBM.getSelectedIndex();
            selectedBM = bm.get(selectedIndex);
        }
    }

    /*
     * This method reads in the customer excel file
     */
    public static ArrayList<CustomerData> getCustomerFile(String fileName)
    {
        ArrayList<CustomerData> newLine = new ArrayList<CustomerData>();
        try
        {
            File file = new File(fileName);
            FileInputStream fis = new FileInputStream(fileName);
            XSSFWorkbook wb = new XSSFWorkbook(fis);
            int numSheets = wb.getNumberOfSheets();
            for(int i = 0; i < numSheets; i++)
            {
                Sheet sheet = wb.getSheetAt(i);
                Iterator<Row> rowIterator = sheet.iterator();
                while(rowIterator.hasNext())
```



```java
        while(rowIterator.hasNext())
        {
            String name = "";
            double count = 0;
            double min = 0.0;
            double max = 0.0;
            double mean = 0.0;
            double median = 0.0;
            Row row = rowIterator.next();
            Iterator<Cell> cellIterator = row.cellIterator();
            while(cellIterator.hasNext())
            {
                Cell cell = cellIterator.next();
                switch(cell.~~getCellType~~())
                {
                case Cell.~~CELL_TYPE_STRING~~:
                    if(name.equals(""))
                        name = cell.getStringCellValue();
                        break;
                case Cell.~~CELL_TYPE_NUMERIC~~:
                    if(count == 0.0)
                        count = cell.getNumericCellValue();
                    else if(min == 0.0)
                        min = cell.getNumericCellValue();
                    else if(max == 0.0)
                        max = cell.getNumericCellValue();
                    else if(mean == 0.0)
                        mean = cell.getNumericCellValue();
                    else if(median == 0.0)
                        median = cell.getNumericCellValue();
                    break;
                }
            }
            CustomerData newCD = new CustomerData(name, count, min, max, mean, median);
            newLine.add(newCD);
        }
    }
    fis.close();
}
catch(IOException e)
{
        catch(IOException e)
        {
            e.printStackTrace();
        }
        return newLine;
    }


    /*
     * This method reads in the benchmark excel file
     */
    public static ArrayList<BenchmarkData> getBenchmarkData(String fileName)
    {
        ArrayList<BenchmarkData> newBenchmarkData = new ArrayList<BenchmarkData>();
        try
        {
            File file = new File(fileName);
            FileInputStream fis = new FileInputStream(file);
            XSSFWorkbook wb = new XSSFWorkbook(fis);
            int numSheets = wb.getNumberOfSheets();
            for(int i = 0; i < numSheets; i++)
            {
                Sheet sheet = wb.getSheetAt(i);
                Iterator<Row> rowIterator = sheet.iterator();
                while(rowIterator.hasNext())
                {
                    String name = "";
                    double oneMedian = 0.0;
                    double median = 0.0;
                    double twoMedian = 0.0;
                    Row row = rowIterator.next();
                    Iterator<Cell> cellIterator = row.cellIterator();
                    while(cellIterator.hasNext())
                    {
                        Cell cell = cellIterator.next();
                        switch(cell.~~getCellType~~())
                        {
                        case Cell.~~CELL_TYPE_STRING~~:
                            if(name.equals(""))
                                name = cell.getStringCellValue();
                            break;
```



```
                        case Cell.~~CELL_TYPE_NUMERIC~~:
                            if(oneMedian == 0.0)
                                oneMedian = cell.getNumericCellValue();
                            else if(median == 0.0)
                                median = cell.getNumericCellValue();
                            else if(twoMedian == 0.0)
                                twoMedian = cell.getNumericCellValue();
                            break;
                        }
                    }
                    BenchmarkData newBM = new BenchmarkData(name, oneMedian, median, twoMedian);
                    newBenchmarkData.add(newBM);
                }
            }
            fis.close();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
        return newBenchmarkData;
    }

    public ArrayList<CustomerData> getCD()
    {
        return cd;
    }


    public ArrayList<BenchmarkData> getBM()
    {
        return bm;
    }

}
```



```
public class Chart extends JFrame {
    XYDataset dataset;
    XYPlot plot;
    JFreeChart chart;
   public Chart(String applicationTitle , String chartTitle, BenchmarkData bm, CustomerData cd)
   {
      Function2D graph = new NormalDistributionFunction2D(bm.getMedian(), getVar(bm));
      dataset = DatasetUtilities.sampleFunction2D(graph, bm.getMedian()*-3.0, bm.getMedian()*5.0, 100, "

      ValueMarker marker1 = new ValueMarker(bm.getLowerMedian());
      marker1.setPaint(Color.black);
      marker1.setLabel("25th Percentile");
      marker1.setStroke(new BasicStroke(5.0f));
      ValueMarker marker2 = new ValueMarker(bm.getUpperMedian());
      marker2.setPaint(Color.black);
      marker2.setLabel("75th Percentile");
      marker2.setStroke(new BasicStroke(5.0f));
      ValueMarker marker3 = new ValueMarker(cd.getMedian());

      if(cd.getMedian() > bm.getUpperMedian())
          marker3.setPaint(Color.red);
      else
          marker3.setPaint(Color.blue);
      marker3.setStroke(new BasicStroke(5.0f));

      NumberAxis xAxis = new NumberAxis(null);
      NumberAxis yAxis = new NumberAxis(null);
      XYAreaRenderer renderer = new XYAreaRenderer();
      renderer.~~setPaint~~(Color.lightGray);
      xAxis.setRange(bm.getMedian()-(getVar(bm)*4),bm.getMedian()+(getVar(bm)*4));
      plot = new XYPlot(dataset, xAxis, yAxis, renderer);
      plot.addDomainMarker(marker1);
      plot.addDomainMarker(marker2);
      plot.addDomainMarker(marker3);


     chart = new JFreeChart(plot);

     ChartPanel chartPanel = new ChartPanel(chart);
     chartPanel.setPreferredSize(new java.awt.Dimension(1000 , 400));
     setContentPane(chartPanel);
  }

  /*
   * This method calculates one distribtion using the selected benchmark
   */

   public double getVar(BenchmarkData bm)
   {
       double x = bm.getUpperMedian();
       double y = bm.getMedian();
       double z = x-y;
       return z/.67;
   }
```