

## View.java

```
1 package expenses.manager;
2
3 import static expenses.manager.Log.logger;
41
42
43 /**
44 *
45 * @author Avital Chen
46 *
47 * View class
48 * class for the app GUI
49 */
50
51 public class View implements IView {
52     JFrame frame;
53     JTabbedPane tabbedPane;
54     JPanel overviewTabPanel;
55     JPanel overviewButtonPanel;
56     JPanel overviewViewPanel;
57     JPanel inputExpensesPanel;
58     JPanel inputIncomePanel;
59     JTextField expensesAmountInput;
60     JTextField incomeAmountInput;
61     JComboBox expensesTypesCb;
62     JDatePickerImpl datePickerExpenses;
63     JDatePickerImpl datePickerIncome;
64
65     JLabel expensesTitle;
66     JLabel incomeTitle;
67     JLabel typeListLabel;
68     JLabel expensesDateTitle;
69     JLabel expensesAmountTitle;
70     JLabel incomeDateTitle;
71     JLabel incomeAmountTitle;
72     JButton incomeSubmitBtn;
73     JButton expensesSubmitBtn;
74
75     JButton tableBtn;
76     JButton expensesCurrYearBtn;
77     JButton expensesCurrMonthBtn;
78     JButton IncomeVsExpensesBtn;
79     Color[] expensesYearColors;
80     Color[] expensesMonthColors;
81
82     JPanel inputTabPanel;
83     IViewModel viewModel;
84
85
86     /**
```

## View.java

```
87  *
88  *View Constructor
89  */
90  public View() {
91      synchronized(logger){
92          logger.info("new View");
93      }
94      expensesYearColors = new Color[] {new Color(0, 31,
77) ,new Color(0, 51, 128), new Color(0, 82, 204), new Color(26,
117, 255), new Color(202, 163, 255), new Color(179, 209, 255) };
95      expensesMonthColors = new Color[] { new Color(77, 0,
40), new Color(128, 0, 66), new Color(204, 0, 105), new
Color(255, 26, 144), new Color(255, 102, 181), new Color(255,
179, 218) };
96      frame = new JFrame("Expenses Managment");
97      tabbedPanel = new JTabbedPane();
98      overviewTabPanel = new JPanel();
99      overviewButtonPanel = new JPanel();
100     overviewViewPanel = new JPanel();
101     inputExpensesPanel = new JPanel();
102     inputIncomePanel = new JPanel();
103     tableBtn = new JButton("Overall");
104     expensesCurrYearBtn = new JButton("Expenses Current
Year");
105     expensesCurrMonthBtn = new JButton("Expenses Current
Month");
106     IncomeVsExpensesBtn = new JButton("Income vs Expenses");
107
108
109
110     inputTabPanel = new JPanel() ;
111
112     expensesTitle = new JLabel("Expenses Input");
113     incomeTitle = new JLabel("Income Input");
114     typeListLabel = new JLabel("Expenses Type");
115     expensesDateTitle= new JLabel("Expenses Date");
116     expensesAmountTitle =new JLabel(" Expenses Amount");
117     incomeDateTitle= new JLabel("Income Date");
118     incomeAmountTitle = new JLabel("Income Amount");
119
120     Properties p = new Properties();
121     p.put("text.today", "Today");
122     p.put("text.month", "Month");
123     p.put("text.year", "Year");
124     incomeAmountInput = new JTextField("Amount");
125     UtilDateModel modelIncome = new UtilDateModel();
126     JDatePanelImpl datePanelIncome = new
JDatePanelImpl(modelIncome, p);
127     datePickerIncome = new JDatePickerImpl(datePanelIncome,
```

## View.java

```

    new DateLabelFormatter());
128     incomeSubmitBtn = new JButton("Submit");
129
130     expensesAmountInput = new JTextField("Amount");
131     String
    expensesTypes[]={"Bills","Shopping","Mortgage","Groceries","Going
    Out", "Oreder In"};
132     expensesTypesCb=new JComboBox(expensesTypes);
133     UtilDateModel modelExpenses = new UtilDateModel();
134     JDatePanelImpl datePanelExpenses = new
    JDatePanelImpl(modelExpenses, p);
135     datePickerExpenses = new
    JDatePickerImpl(datePanelExpenses, new DateLabelFormatter());
136     expensesSubmitBtn = new JButton("Submit");
137 }
138
139 /**
140  * getTableInfo()
141  * @param infoArray array of doubles with the data that was
    got from the database
142  * this function adds to the view an overview table
143  */
144 @Override
145 public void getTableInfo(ArrayList<Double> infoArray) {
146     synchronized(logger){
147         logger.info("Generate info table ");
148     }
149     clear(overviewViewPanel);
150     JTable infoTable = createTable(infoArray.get(0),
    infoArray.get(1), infoArray.get(2), infoArray.get(3));
151
152     if(infoTable == null) System.out.println("cheking if
    table nulll");
153     JScrollPane sp=new JScrollPane(infoTable);
154
155     overviewViewPanel.add(sp);
156     overviewViewPanel.revalidate();
157     overviewViewPanel.repaint();
158 }
159 /**
160  * getExpensesByTypeCurrMonth()
161  * @param data Vector of [ double expensesAmount, String
    month ]
162  * this function adds to the view a pie chart
163  */
164 @Override
165 public void getExpensesByTypeCurrMonth(Vector data) {
166     synchronized(logger){
167         logger.info("Generate Expenses by type current month

```

## View.java

```

        chart ");
168    }
169    clear(overviewViewPanel);
170    PieChart chart = createPieChart(data, "Expenses by Type
Current Month", expensesMonthColors);
171    XChartPanel<PieChart> chartPanel = new
XChartPanel<>(chart);
172    overviewViewPanel.add(chartPanel);
173    overviewViewPanel.revalidate();
174    overviewViewPanel.repaint();
175    }
176    /**
177     * getExpendsesByTypeCurrYear()
178     * @param data Vector of [ double expensesAmount, String
month ]
179     * this function adds to the view a pie chart
180     * only this year
181     */
182     @Override
183     public void getExpendsesByTypeCurrYear(Vector data) {
184         synchronized(logger){
185             logger.info("Generate Expenses by type current year
chart");
186         }
187         clear(overviewViewPanel);
188         PieChart chart = createPieChart(data, "Expenses by Type
Current Year", expensesMonthColors);
189         XChartPanel<PieChart> chartPanel = new
XChartPanel<>(chart);
190         overviewViewPanel.add(chartPanel);
191         overviewViewPanel.revalidate();
192         overviewViewPanel.repaint();
193     }
194
195     /**
196     * getExpendsesByTypeCurrYear()
197     * @param data Vector of vector  [[ double expensesAmount,
String month ], [ double incomeAmount, String month ]]
198     * this function adds to the view a bar chart
199     * only this year
200     */
201     @Override
202     public void getIncomeAndExpensesByMonth( Vector data) {
203         synchronized(logger){
204             logger.info("Generate Expenses and Income by month
chart");
205         }
206         clear(overviewViewPanel);
207         CategoryChart chart = createBarChart((Vector)data.get(0),

```

## View.java

```
(Vector)data.get(1));
208     XChartPanel<CategoryChart> chartPanel = new
XChartPanel<>(chart);
209     overviewViewPanel.add(chartPanel);
210     overviewViewPanel.revalidate();
211     overviewViewPanel.repaint();
212 }
213
214 @Override
215 public void setViewModel(IViewModel ob) {
216     this.viewmodel = ob;
217 }
218
219 /**
220  * start()
221  * View GUI start function
222  * the view has 2 tabs
223  * 1 for the chart and table - dashboard
224  * 2 for the user input
225  */
226 @Override
227 public void start() {
228     synchronized(logger){
229         logger.info("View Start");
230     }
231     Color bgColor = new Color(211, 234, 245);
232     GridBagLayout GridBagLayoutgrid = new GridBagLayout();
233     GridBagConstraints gbc = new GridBagConstraints();
234     overviewTabPanel.setBackground(bgColor);
235     overviewTabPanel.setLayout(GridBagLayoutgrid);
236     gbc.fill = GridBagConstraints.HORIZONTAL;
237     gbc.gridx = 0;
238     gbc.gridy = 0;
239     overviewTabPanel.add(overviewButtonPanel, gbc);
240     overviewViewPanel.setBackground(Color.WHITE);
241     overviewViewPanel.setPreferredSize(new Dimension(700,
500));
242     gbc.gridx = 0;
243     gbc.gridy = 1;
244     overviewTabPanel.add(overviewViewPanel,gbc);
245     overviewButtonPanel.setBackground(bgColor);
246     overviewButtonPanel.add(tableBtn);
247     overviewButtonPanel.add(IncomeVsExpensesBtn);
248     overviewButtonPanel.add(expensesCurrMonthBtn);
249     overviewButtonPanel.add(expensesCurrYearBtn);
250
251
252
253     inputTabPanel.setLayout(GridBagLayoutgrid);
```

## View.java

```
254     inputTabPanel.setBackground(bgColor);
255     inputExpensesPanel.setLayout(GridBagLayoutgrid);
256     inputExpensesPanel.setPreferredSize(new Dimension(400,
500));
257     inputExpensesPanel.setBackground(Color.WHITE);
258
259     gbc.fill = GridBagConstraints.HORIZONTAL;
260     gbc.insets = new Insets(0,0,0,100);
261     gbc.gridx = 0;
262     gbc.gridy = 0;
263     inputTabPanel.add(inputExpensesPanel, gbc);
264     gbc.fill = GridBagConstraints.HORIZONTAL;
265     gbc.insets = new Insets(0,100,0,0);
266     gbc.gridx = 1;
267     gbc.gridy = 0;
268     inputTabPanel.add(inputIncomePanel,gbc);
269
270     gbc.fill = GridBagConstraints.HORIZONTAL;
271     gbc.insets = new Insets(0,0,0,0);
272     gbc.gridx = 0;
273     gbc.gridy = 0;
274     inputExpensesPanel.add(expensesTitle, gbc);
275     gbc.insets = new Insets(20,0,0,20);
276     gbc.gridx = 0;
277     gbc.gridy = 1;
278     inputExpensesPanel.add(typeListLabel,gbc);
279     gbc.gridx = 0;
280     gbc.gridy = 2;
281     inputExpensesPanel.add(expensesDateTitle, gbc);
282     gbc.gridx = 0;
283     gbc.gridy = 3;
284     inputExpensesPanel.add(expensesAmountTitle, gbc);
285     gbc.insets = new Insets(20,0,0,20);
286     gbc.gridx = 1;
287     gbc.gridy = 1;
288     inputExpensesPanel.add(expensesTypesCb,gbc);
289     gbc.gridx = 1;
290     gbc.gridy = 2;
291     inputExpensesPanel.add(datePickerExpenses, gbc);
292     gbc.gridx = 1;
293     gbc.gridy = 3;
294     inputExpensesPanel.add(expensesAmountInput, gbc);
295     gbc.insets = new Insets(20,0,0,0);
296     gbc.gridx = 0;
297     gbc.gridy = 4;
298     gbc.gridwidth = 2;
299     inputExpensesPanel.add(expensesSubmitBtn,gbc);
300
301
```

## View.java

```
302     inputIncomePanel.setLayout(GridBagLayoutgrid);
303     inputIncomePanel.setPreferredSize(new Dimension(400,
500));
304     inputIncomePanel.setBackground(Color.WHITE);
305     gbc.gridwidth = 0;
306     gbc.insets = new Insets(0,0,0,0);
307     gbc.gridx = 0;
308     gbc.gridy = 0;
309     inputIncomePanel.add(incomeTitle, gbc);
310     gbc.insets = new Insets(20,0,0,0);
311     gbc.gridx = 0;
312     gbc.gridy = 1;
313     inputIncomePanel.add(incomeDateTitle, gbc);
314     gbc.gridx = 0;
315     gbc.gridy = 2;
316     inputIncomePanel.add(incomeAmountTitle, gbc);
317     gbc.insets = new Insets(20,100,0,0);
318     gbc.gridx = 1;
319     gbc.gridy = 1;
320     inputIncomePanel.add(datePickerIncome, gbc);
321     gbc.gridx = 1;
322     gbc.gridy = 2;
323     inputIncomePanel.add(incomeAmountInput, gbc);
324     gbc.insets = new Insets(20,0,0,0);
325     gbc.gridx = 0;
326     gbc.gridy = 3;
327     gbc.gridwidth = 3;
328     inputIncomePanel.add(incomeSubmitBtn, gbc);
329
330     incomeSubmitBtn.addActionListener(new ActionListener(){
331         public void actionPerformed(ActionEvent e){
332             String textFieldValue =
incomeAmountInput.getText();
333             String datePickerValue =
datePickerIncome.getJFormattedTextField().getText();
334             viewmodel.setIncome(datePickerValue,
Double.parseDouble(textFieldValue));
335         }
336     });
337
338     expensesSubmitBtn.addActionListener(new ActionListener(){
339         public void actionPerformed(ActionEvent e){
340             String textFieldValue =
expensesAmountInput.getText();
341             String comboboxValue =
(String)expensesTypesCb.getSelectedItemAt();
342             String datePickerValue =
datePickerExpenses.getJFormattedTextField().getText();
343             viewmodel.setExpenses(comboboxValue,
```

## View.java

```
        datePickerValue, Double.parseDouble(textFieldValue));
344    }
345    });
346
347
348    tableBtn.addActionListener(new ActionListener(){
349        public void actionPerformed(ActionEvent e){
350            viewmodel.getTableInfo();
351        }
352    });
353
354
355    IncomeVsExpensesBtn.addActionListener(new
ActionListener(){
356        public void actionPerformed(ActionEvent e){
357            viewmodel.getInclomeAndExpensesByMonth();
358        }
359    });
360
361
362    expensesCurrMonthBtn.addActionListener(new
ActionListener(){
363        public void actionPerformed(ActionEvent e){
364            viewmodel.getTotalExpensesByTypeCurrMonth();
365        }
366    });
367
368
369    expensesCurrYearBtn.addActionListener(new
ActionListener(){
370        public void actionPerformed(ActionEvent e){
371            viewmodel.getTotalExpensesByTypeCurrYear();
372        }
373    });
374
375
376    tabbedPanel.add(overviewTabPanel, "Overview");
377    tabbedPanel.add(inputTabPanel, "Input");
378
379
380    frame.add(tabbedPanel);
381    frame.setVisible(true);
382    frame.setSize(1000,500);
383
384
385    }
386
387    /**
388    * clear()
```



# View.java

```

389     * clear the dashboard panel
390     */
391     public static void clear(JPanel panel)
392     {
393         if (panel.getComponentCount() == 1)
394         {
395             panel.remove(0);
396             panel.updateUI();
397         }
398     }
399
400     public class DateLabelFormatter extends AbstractFormatter {
401
402         private String datePattern = "yyyy-MM-dd";
403         private SimpleDateFormat dateFormatter = new
SimpleDateFormat(datePattern);
404
405         @Override
406         public Object stringToValue(String text) throws
ParseException {
407             return dateFormatter.parseObject(text);
408         }
409
410         @Override
411         public String valueToString(Object value) throws
ParseException {
412             if (value != null) {
413                 Calendar cal = (Calendar) value;
414                 return dateFormatter.format(cal.getTime());
415             }
416
417             return "";
418         }
419     }
420
421     /**
422     * createTable()
423     * Table generator function
424     *
425     */
426     public JTable createTable(double incomeOverall, double
expensesOverall, double incomeCurrYear, double expensesCurrYear )
    {
427         String data[][]={ {"Overall",
String.valueOf(incomeOverall),
String.valueOf(expensesOverall) ,String.valueOf(incomeOverall -
expensesOverall) },
428                             {"Current Year",
String.valueOf(incomeCurrYear), String.valueOf(expensesCurrYear),

```

# View.java

```

String.valueOf(incomeCurrYear - expensesCurrYear) }};
429     String column[]={
    "Income", "Expenses", "Balance" };
430     JTable jt=new JTable(data, column);
431     return jt;
432 }
433 /**
434  * createBarChart()
435  * Bar chart generator function
436  *
437  */
438     public CategoryChart createBarChart(Vector income, Vector
expenses ){
439         Double incomeArray[] = new Double[12];
440         Arrays.fill(incomeArray, 0.0);
441         Double expensesArray[] = new Double[12];
442         Arrays.fill(expensesArray, 0.0);
443         Iterator incomeIterator = income.iterator();
444         while (incomeIterator.hasNext()) {
445             Vector v = (Vector) incomeIterator.next();
446             incomeArray[Integer.valueOf((String)v.get(1))-1]
= (Double) v.get(0);
447         }
448         Iterator expensesIterator = expenses.iterator();
449         while (expensesIterator.hasNext()) {
450             Vector v = (Vector) expensesIterator.next();
451             expensesArray[Integer.valueOf((String)v.get(1))-1] = (Double)
v.get(0);
452         }
453         String months[] =
454         {
455             "Jan" , "Feb" , "Mar" , "Apr" , "May",
456             "Jun", "Jul", "Aug", "Sep", "Oct",
457             "Nov", "Dec"
458         };
459
460         CategoryChart chart = new
CategoryChartBuilder().width(600).height(300).title("Income vs.
Expenses").xAxisTitle("Month").yAxisTitle("Amount").theme(ChartTh
eme.GGPlot2).build();
461         chart.addSeries("Income", new
ArrayList<String>(Arrays.asList(months)), new
ArrayList<Double>(Arrays.asList(incomeArray)));
462         chart.addSeries("Expenses", new
ArrayList<String>(Arrays.asList(months)), new
ArrayList<Double>(Arrays.asList(expensesArray)));
463         chart.getStyler().setSeriesColors(new Color[]{new
Color(46, 184, 46), new Color(255, 102, 102)});

```

View.java

```
464         return chart;
465     }
466
467     /**
468     * createPieChart()
469     * Pie chart generator function
470     *
471     */
472     public PieChart createPieChart(Vector expenses, String title,
    Color[] seriesColor){
473         PieChart chart = new
    PieChartBuilder().width(400).height(300).title(title).build();
474         Iterator expensesIterator = expenses.iterator();
475         while (expensesIterator.hasNext()) {
476             Vector v = (Vector)expensesIterator.next();
477             chart.addSeries((String)v.get(0),
    (Number)Math.round(((Double) v.get(1)).intValue()));
478         }
479         chart.getStyle().setSeriesColors(seriesColor);
480
481         return chart;
482     }
483 }
484
```