```java
package expenses.manager;

import static expenses.manager.Log.logger;


/**
*
* @author Avital Chen
*
* View class
* class for the app GUI
*/

public class View implements IView {
    JFrame frame;
    JTabbedPane tabbedPanel;
    JPanel overviewTabPanel;
    JPanel overviewButtonPanel;
    JPanel overviewViewPanel;
    JPanel inputExpensesPanel;
    JPanel inputIncomePanel;
    JTextField expensesAmountInput;
    JTextField incomeAmountInput;
    JComboBox expensesTypesCb;
    JDatePickerImpl datePickerExpenses;
    JDatePickerImpl datePickerIncome;

    JLabel expensesTitle;
    JLabel incomeTitle;
    JLabel typeListLabel;
    JLabel expensesDateTitle;
    JLabel expensesAmountTitle;
    JLabel incomeDateTitle;
    JLabel incomeAmountTitle;
    JButton incomeSubmitBtn;
    JButton expensesSubmitBtn;

    JButton tableBtn;
    JButton expensesCurrYearBtn;
    JButton expensesCurrMonthBtn;
    JButton IncomeVsExpensesBtn;
    Color[] expensesYearColors;
    Color[] expensesMonthColors;

    JPanel inputTabPanel;
    IViewModel viewmodel;


    /**
    *
    *View Constructor
    */
    public View() {
        synchronized(logger){
             logger.info("new View");
         }
         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) };
         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) };
         frame = new JFrame("Expenses Managment");
         tabbedPanel = new JTabbedPane();
         overviewTabPanel = new JPanel();
         overviewButtonPanel = new JPanel();
         overviewViewPanel = new JPanel();
         inputExpensesPanel =  new JPanel();
         inputIncomePanel =  new JPanel();
         tableBtn = new JButton("Overall");
         expensesCurrYearBtn = new JButton("Expenses Current Year");
         expensesCurrMonthBtn = new JButton("Expenses Current Month");
         IncomeVsExpensesBtn = new JButton("Income vs Expenses");



         inputTabPanel = new JPanel() ;

          expensesTitle = new JLabel("Expenses Input");
          incomeTitle = new JLabel("Income Input");
          typeListLabel = new JLabel("Expenses Type");
          expensesDateTitle= new JLabel("Expenses Date");
          expensesAmountTitle =new JLabel(" Expenses Amount");
          incomeDateTitle= new JLabel("Income Date");
          incomeAmountTitle = new JLabel("Income Amount");

         Properties p = new Properties();
            p.put("text.today", "Today");
            p.put("text.month", "Month");
            p.put("text.year", "Year");
         incomeAmountInput = new JTextField("Amount");
         UtilDateModel modelIncome = new UtilDateModel();
         JDatePanelImpl datePanelIncome = new JDatePanelImpl(modelIncome, p);
         datePickerIncome = new JDatePickerImpl(datePanelIncome,
```

```java
new DateLabelFormatter());
        incomeSubmitBtn = new JButton("Submit");

        expensesAmountInput = new JTextField("Amount");
        String expensesTypes[]={"Bills","Shopping","Mortgage","Groceries","Going Out", "Oreder In"};
        expensesTypesCb=new JComboBox(expensesTypes);
        UtilDateModel modelExpenses = new UtilDateModel();
        JDatePanelImpl datePanelExpenses = new JDatePanelImpl(modelExpenses, p);
        datePickerExpenses = new JDatePickerImpl(datePanelExpenses, new DateLabelFormatter());
        expensesSubmitBtn = new JButton("Submit");
    }

    /**
    * getTableInfo()
    * @param infoArray array of doubles with the data that was got from the database
    * this function adds to the view an overview table
    */
    @Override
    public void getTableInfo(ArrayList<Double> infoArray) {
        synchronized(logger){
            logger.info("Generate info table ");
        }
           clear(overviewViewPanel);
           JTable infoTable  = createTable(infoArray.get(0), infoArray.get(1), infoArray.get(2), infoArray.get(3));

           if(infoTable == null) System.out.println("cheking if table nulll");
           JScrollPane sp=new JScrollPane(infoTable);

           overviewViewPanel.add(sp);
           overviewViewPanel.revalidate();
           overviewViewPanel.repaint();
    }
    /**
    * getExpensesByTypeCurrMonth()
    * @param data Vector of [ double expensesAmount, String month ]
    * this function adds to the view a pie chart
    */
    @Override
    public void getExpensesByTypeCurrMonth(Vector data) {
        synchronized(logger){
            logger.info("Generate Expenses by type current month
chart ");
         }
        clear(overviewViewPanel);
        PieChart chart = createPieChart(data, "Expenses by Type Current Month", expensesMonthColors);
        XChartPanel<PieChart> chartPanel = new XChartPanel<>(chart);
        overviewViewPanel.add(chartPanel);
        overviewViewPanel.revalidate();
        overviewViewPanel.repaint();
    }
    /**
    * getExpendsesByTypeCurrYear()
    * @param data Vector of [ double expensesAmount, String month ]
    * this function adds to the view a pie chart
    * only this year
    */
    @Override
    public void getExpendsesByTypeCurrYear(Vector data) {
         synchronized(logger){
             logger.info("Generate Expenses by type current year chart");
         }
        clear(overviewViewPanel);
        PieChart chart = createPieChart(data, "Expenses by Type Current Year", expensesMonthColors);
        XChartPanel<PieChart> chartPanel = new XChartPanel<>(chart);
        overviewViewPanel.add(chartPanel);
        overviewViewPanel.revalidate();
        overviewViewPanel.repaint();
    }

    /**
    * getExpendsesByTypeCurrYear()
    * @param data Vector of vector  [[ double expensesAmount, String month ], [ double incomeAmount, String month ]]
    * this function adds to the view a bar chart
    * only this year
    */
    @Override
    public void getIncomeAndExpensesByMonth( Vector data) {
         synchronized(logger){
             logger.info("Generate Expenses and Income by month chart");
         }
        clear(overviewViewPanel);
        CategoryChart chart = createBarChart((Vector)data.get(0),
(Vector)data.get(1));
        XChartPanel<CategoryChart> chartPanel = new XChartPanel<>(chart);
        overviewViewPanel.add(chartPanel);
        overviewViewPanel.revalidate();
        overviewViewPanel.repaint();
    }

    @Override
    public void setViewModel(IViewModel ob) {
        this.viewmodel = ob;
    }

    /**
    * start()
    * View GUI start function
    * the view has 2 tabs
    * 1 for the chart and table - dashboard
    * 2 for the user input
    */
    @Override
    public void start() {
         synchronized(logger){
             logger.info("View Start");
         }
        Color bgColor = new Color(211, 234, 245);
        GridBagLayout GridBagLayoutgrid = new GridBagLayout();
        GridBagConstraints gbc = new GridBagConstraints();
        overviewTabPanel.setBackground(bgColor);
        overviewTabPanel.setLayout(GridBagLayoutgrid);
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridx = 0;
        gbc.gridy = 0;
        overviewTabPanel.add(overviewButtonPanel, gbc);
        overviewViewPanel.setBackground(Color.WHITE);
        overviewViewPanel.setPreferredSize(new Dimension(700, 500));
         gbc.gridx =0;
         gbc.gridy = 1;
        overviewTabPanel.add(overviewViewPanel,gbc);
        overviewButtonPanel.setBackground(bgColor);
        overviewButtonPanel.add(tableBtn);
        overviewButtonPanel.add(IncomeVsExpensesBtn);
        overviewButtonPanel.add(expensesCurrMonthBtn);
        overviewButtonPanel.add(expensesCurrYearBtn);



        inputTabPanel.setLayout(GridBagLayoutgrid);
        inputTabPanel.setBackground(bgColor);
        inputExpensesPanel.setLayout(GridBagLayoutgrid);
        inputExpensesPanel.setPreferredSize(new Dimension(400,
500));
        inputExpensesPanel.setBackground(Color.WHITE);

        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(0,0,0,100);
        gbc.gridx = 0;
        gbc.gridy = 0;
        inputTabPanel.add(inputExpensesPanel, gbc);
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(0,100,0,0);
        gbc.gridx = 1;
        gbc.gridy = 0;
        inputTabPanel.add(inputIncomePanel,gbc);

        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(0,0,0,0);
        gbc.gridx = 0;
        gbc.gridy = 0;
        inputExpensesPanel.add(expensesTitle, gbc);
        gbc.insets = new Insets(20,0,0,20);
        gbc.gridx = 0;
        gbc.gridy = 1;
        inputExpensesPanel.add(typeListLabel,gbc);
        gbc.gridx = 0;
        gbc.gridy = 2;
        inputExpensesPanel.add(expensesDateTitle, gbc);
        gbc.gridx = 0;
        gbc.gridy = 3;
        inputExpensesPanel.add(expensesAmountTitle, gbc);
        gbc.insets = new Insets(20,0,0,20);
        gbc.gridx = 1;
        gbc.gridy = 1;
        inputExpensesPanel.add(expensesTypesCb,gbc);
        gbc.gridx = 1;
        gbc.gridy = 2;
        inputExpensesPanel.add(datePickerExpenses, gbc);
        gbc.gridx = 1;
        gbc.gridy = 3;
        inputExpensesPanel.add(expensesAmountInput, gbc);
        gbc.insets = new Insets(20,0,0,0);
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 2;
        inputExpensesPanel.add(expensesSubmitBtn,gbc);


```

```java
        inputIncomePanel.setLayout(GridBagLayoutgrid);
        inputIncomePanel.setPreferredSize(new Dimension(400, 500));
        inputIncomePanel.setBackground(Color.WHITE);
        gbc.gridwidth = 0;
        gbc.insets = new Insets(0,0,0,0);
        gbc.gridx = 0;
        gbc.gridy = 0;
        inputIncomePanel.add(incomeTitle, gbc);
        gbc.insets = new Insets(20,0,0,0);
        gbc.gridx = 0;
        gbc.gridy = 1;
        inputIncomePanel.add(incomeDateTitle, gbc);
        gbc.gridx = 0;
        gbc.gridy = 2;
        inputIncomePanel.add(incomeAmountTitle, gbc);
        gbc.insets = new Insets(20,100,0,0);
        gbc.gridx = 1;
        gbc.gridy = 1;
        inputIncomePanel.add(datePickerIncome, gbc);
        gbc.gridx = 1;
        gbc.gridy = 2;
        inputIncomePanel.add(incomeAmountInput, gbc);
        gbc.insets = new Insets(20,0,0,0);
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.gridwidth = 3;
        inputIncomePanel.add(incomeSubmitBtn, gbc);

        incomeSubmitBtn.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                     String textFieldValue = incomeAmountInput.getText();
                     String datePickerValue = datePickerIncome.getJFormattedTextField().getText();
                     viewmodel.setIncome(datePickerValue, Double.parseDouble(textFieldValue));
                }
            });

        expensesSubmitBtn.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                 String textFieldValue = expensesAmountInput.getText();
                 String comboboxValue = (String)expensesTypesCb.getSelectedItem();
                 String datePickerValue = datePickerExpenses.getJFormattedTextField().getText();
                 viewmodel.setExpenses(comboboxValue,
datePickerValue, Double.parseDouble(textFieldValue));
            }
        });


        tableBtn.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                viewmodel.getTableInfo();

            }
        });

        IncomeVsExpensesBtn.addActionListener(new
ActionListener(){
            public void actionPerformed(ActionEvent e){
                viewmodel.getInclomeAndExpensesByMonth();

            }
        });

        expensesCurrMonthBtn.addActionListener(new
ActionListener(){
            public void actionPerformed(ActionEvent e){
                viewmodel.getTotalExpensesByTypeCurrMonth();

            }
        });

        expensesCurrYearBtn.addActionListener(new
ActionListener(){
            public void actionPerformed(ActionEvent e){
                viewmodel.getTotalExpensesByTypeCurrYear();

            }
        });

        tabbedPanel.add(overviewTabPanel, "Overview");
        tabbedPanel.add(inputTabPanel, "Input");


        frame.add(tabbedPanel);
        frame.setVisible(true);
        frame.setSize(1000,500);


    }

    /**
     * clear()
```

```java
     * clear the dashboard panel
     */
    public static void clear(JPanel panel)
    {
        if (panel.getComponentCount() == 1)
        {
            panel.remove(0);
            panel.updateUI();
        }
    }

     public class DateLabelFormatter extends AbstractFormatter {

            private String datePattern = "yyyy-MM-dd";
            private SimpleDateFormat dateFormatter = new SimpleDateFormat(datePattern);

            @Override
            public Object stringToValue(String text) throws ParseException {
                return dateFormatter.parseObject(text);
            }

            @Override
            public String valueToString(Object value) throws ParseException {
                if (value != null) {
                    Calendar cal = (Calendar) value;
                    return dateFormatter.format(cal.getTime());
                }

                return "";
            }

        }
        /**
         * createTable()
         * Table generator function
         *
         */
     public JTable createTable(double incomeOverall, double expensesOverall, double incomeCurrYear, double expensesCurrYear )
{
         String data[][]={ {"Overall", String.valueOf(incomeOverall), String.valueOf(expensesOverall) ,String.valueOf(incomeOverall - expensesOverall) },
                                {"Current Year", String.valueOf(incomeCurrYear), String.valueOf(expensesCurrYear),
String.valueOf(incomeCurrYear - expensesCurrYear) }};
       String column[]={" ","Income","Expenses","Balance" };
       JTable jt=new JTable(data, column);
      return jt;
    }
  /**
  * createBarChart()
  * Bar chart generator function
  *
  */
  public CategoryChart createBarChart(Vector income, Vector expenses ){
      Double incomeArray[] = new Double[12];
      Arrays.fill(incomeArray, 0.0);
      Double expensesArray[] = new Double[12];
      Arrays.fill(expensesArray, 0.0);
      Iterator incomeIterator = income.iterator();
      while (incomeIterator.hasNext()) {
        Vector v = (Vector) incomeIterator.next();
        incomeArray[Integer.valueOf((String)v.get(1))-1] = (Double) v.get(0);
      }
      Iterator expensesIterator = expenses.iterator();
      while (expensesIterator.hasNext()) {
        Vector v = (Vector) expensesIterator.next();
expensesArray[Integer.valueOf((String)v.get(1))-1] = (Double) v.get(0);
      }
      String months[] =
        {
          "Jan" , "Feb" , "Mar" , "Apr", "May",
          "Jun", "Jul", "Aug", "Sep", "Oct",
          "Nov", "Dec"
        };

      CategoryChart chart = new CategoryChartBuilder().width(600).height(300).title("Income vs. Expenses").xAxisTitle("Month").yAxisTitle("Amount").theme(ChartTheme.GGPlot2).build();
      chart.addSeries("Income", new ArrayList<String>(Arrays.asList(months)), new ArrayList<Double>(Arrays.asList(incomeArray)));
      chart.addSeries("Expenses", new ArrayList<String>(Arrays.asList(months)), new ArrayList<Double>(Arrays.asList(expensesArray)));
      chart.getStyler().setSeriesColors(new Color[]{new Color(46, 184, 46), new Color(255, 102, 102)});
```

```java
            return chart;
        }

     /**
    * createPieChart()
    * Pie chart generator function
    *
    */
    public PieChart createPieChart(Vector expenses, String title, Color[] seriesColor){
            PieChart chart = new PieChartBuilder().width(400).height(300).title(title).build();
            Iterator expensesIterator = expenses.iterator();
            while (expensesIterator.hasNext()) {
                Vector v = (Vector)expensesIterator.next();
                chart.addSeries((String)v.get(0), (Number)Math.round(((Double) v.get(1)).intValue()));
             }
            chart.getStyler().setSeriesColors(seriesColor);

            return chart;
        }
}

```