Data Visualization System In Java

Introduction of the Project

In this Java project, we will develop Data Visualization System In Java that will create a pie chart for the information entered by the user. In today's data-driven world, the ability to make sense of vast amounts of information is critical. Data visualization has become an essential tool for businesses, organizations, and individuals alike to understand and communicate complex data.

With Java's powerful capabilities, building a data visualization system has never been easier. Java's flexibility, speed, and security make it an ideal language for creating dynamic visualizations that can be customized to meet the specific needs of any project. Whether you're looking to create interactive dashboards, charts, or maps, a data visualization system in Java can help you turn your data into insights and make informed decisions.

We are going to use Swing to implement the GUI of this project along with Jfreechart package to create a good looking pie chart.

Objectives

The objectives for building a Data Visualization System in Java may vary depending on the specific needs and goals of the project. However, some common objectives for building such a system might include the following:

1. To implement the tables that contains the item names & amount & using that information generate a pie chart from the class **JfreeChart**.
2. ***To improve data analysis and decision-making:*** A data visualization system can help users to understand complex data better, identify trends and patterns, and make more informed decisions.

3. ***To enhance data communication:*** A well-designed visualization can help to communicate complex data in a more accessible and understandable way, making it easier to share insights with others.
4. ***To automate data processing and visualization tasks:*** A data visualization system can automate data processing and visualization tasks, reducing the time and effort required to analyze and communicate data.
5. ***To provide real-time data monitoring:*** A data visualization system can be used to monitor data in real-time, providing users with up-to-date information and allowing them to respond quickly to changes.
6. ***To enable customization and flexibility:*** A data visualization system built in Java can be highly customizable and flexible, allowing users to create visualizations that meet their specific needs and requirements.

Requirements

Knowledge of Java programming language: Building a Data Visualization System in Java requires a strong understanding of the Java programming language, including object-oriented programming concepts and data structures.

Familiarity with data visualization concepts: It is essential to have a good understanding of data visualization concepts such as chart types, data mapping, color theory, and visualization best practices.

Knowledge of Java libraries and frameworks: Java has several libraries and frameworks that are specifically designed for data visualization, such as JavaFX, JFreeChart, and Java 3D. JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications.

Ability to design user-friendly interfaces using Swing framework: A Data Visualization System should be easy to use and understand. Therefore, having an understanding of user interface design principles and user experience (UX) best practices is essential.

Understanding of data storage and retrieval: The Data Visualization System in Java will likely need to store and retrieve data from various sources, such as databases, CSV files, or APIs. Therefore, it is essential to have an understanding of how to access, manipulate and store data.

Source Code

Main.java

```
package com.company;

public class Main {

public static void main(String[] args) {

new PieChart();

}

}
```

PieChart.java

```
package com.company;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PiePlot;
import org.jfree.data.general.DefaultPieDataset;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class PieChart {



             private JTextField item;
             private JTextField amount;
             private JButton ADDDATAButton;
             private JPanel mainFrame;
             private JPanel panel2;
             private JButton PIECHARTButton;
             private JPanel piePanel;
             private JButton RESETButton;
             private DefaultPieDataset pieDataset;
             private JFreeChart pieChart;
             private ChartPanel chartPanel;
             DefaultTableModel model;
             JTable table;
             JFrame frame = new JFrame();
             public PieChart() {
                    mainFrame = new JPanel();
                    panel2=new JPanel();
                    piePanel = new JPanel();
                    ADDDATAButton = new JButton("Add Data");
                    PIECHARTButton =new JButton("Pie Chart");
                    RESETButton=new JButton("Reset");

                    item= new JTextField(10);
                    amount =new JTextField(10);

                    mainFrame.setLayout(new BoxLayout(mainFrame,
BoxLayout.Y_AXIS));

                    JPanel inputPanel = new JPanel();
                    inputPanel.add(new JLabel("item"));

                    inputPanel.add(item);
                    inputPanel.add(new JLabel("Amount:"));

                    inputPanel.add(amount);
                    inputPanel.add(ADDDATAButton);
                    inputPanel.add(PIECHARTButton);
                    inputPanel.add(RESETButton);
                    mainFrame.add(inputPanel);
                    mainFrame.add(panel2);
                    mainFrame.add(piePanel);



            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setContentPane(mainFrame);
            frame.pack();
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
            displayTable();

            ADDDATAButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
            String itemName = item.getText();
            String amountData = amount.getText();
            Object[] data = {itemName,amountData};
            model.addRow(data);
            item.setText("");
            amount.setText("");
            }
            });
            PIECHARTButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
            piePanel.removeAll();
            showPie();
            frame.validate();
            }
            });
            RESETButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
            piePanel.removeAll();
            panel2.removeAll();
            displayTable();
            frame.validate();
```

```
              }
              });
              }
              public void displayTable(){
              String[] a = {"ITEMS","AMOUNT"};
              model = new DefaultTableModel(null,a);
              table = new JTable(model);
              panel2.add(new JScrollPane(table));
              }
              public void showPie(){
              pieDataset = new DefaultPieDataset();
              for(int i=0;i<table.getRowCount();i++){
              String name = table.getValueAt(i,0).toString();
              Double amt =
              Double.valueOf(table.getValueAt(i,1).toString());
              pieDataset.setValue(name,amt);
              }
              pieChart = ChartFactory.createPieChart("PIE
CHART",pieDataset,true,true,true);
              pieChart.getPlot();
              chartPanel = new ChartPanel(pieChart);
              piePanel.add(chartPanel);


       }

}
```

Explanation of the Code

We basically had 2 major tasks. One involves creating the GUI, and the other is retrieval of information from the table and pass it to the jfreechart object to generate a pie chart. *Let us look at the GUI first:*

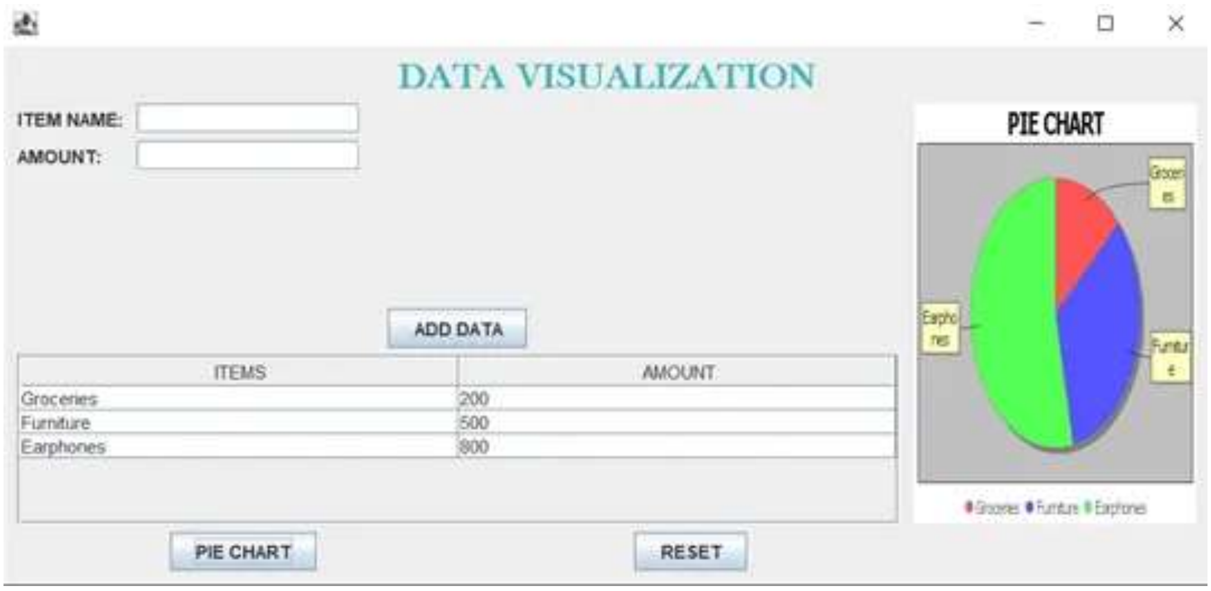
1. The input screen consists of 2 textfields, a Jtable , a Jpanel for pie chart.
2. Our user interface consists of 3 buttons ***"Add Data"***, ***"Pie Chart"*** & ***"Reset"*** respectively.

Moving to generating pie chart, we will apply the following:

1. We firstly create an object of DefaultPieDataset class.
2. Then we enter the entity name & its value using setValue method using the object you made.
3. Use method createPieChart of ChartFactory class and pass your object.
4. Finally, add the piechart to the Jpanel.

Output

Main Interface



ITEMS	AMOUNT
Groceries	200
Furniture	500
Earphones	800

We have successfully built a Data Visualization System In Java that helps to create a pie chart with Swing used for GUI. This is a very efficient way to visualize data & to make decisions based on the Pie chart.