Report

Chapter 1 – Object-Oriented Principles

1.1 Inheritance

The class **cw3** extends the **Application** class, which is a part of JavaFX. This signifies that **cw3** inherits methods and properties from **Application**. The **NumWords** class, nested within **cw3**, extends **Pane**, indicating it inherits the characteristics of a JavaFX pane. This relationship enables the utilization of Pane functionality while adding custom features for displaying word count diagrams.

1.2 Encapsulation

The **NumWords** class encapsulates the functionalities related to the word count and display. It maintains its internal state and exposes specific functionalities like **wordCount**, **paint**, **setWidth**, and **setHeight**.

1.3 Polymorphism

Polymorphism is exhibited through the method overriding. The **start** method of **Application** is overridden in **cw3**. Similarly, **setWidth** and **setHeight** methods from the **Pane** class are overridden in **NumWords** to provide specific implementations.

1.4 Class and Object

The **cw3** and **NumWords** classes represent blueprints for creating objects. Instances of these classes are used to execute the desired functionalities.

Chapter 2 – Diagram Display Algorithm

The **paint** method in the **NumWords** class is responsible for displaying a Sankey diagram. It involves the following steps:

. **Initial Setup**: A large rectangle is created at left as the total number for the diagram.
. **Rectangle Creation**: Smaller rectangles, representing different data points, are created. Their sizes are proportional to the data values.
. **Bezier Curve Drawing**: For each data point, a Bezier curve is drawn from the large rectangle to the respective small rectangle, indicating a flow or relationship.
. **Data Addition**: Textual data is added alongside the rectangles to provide information about each data point.

Chapter 3 – Display while Resizing Algorithm

The paint() method is designed to be responsive when the size of the window changes. It recalculates the positions and sizes of diagram elements based on the new dimensions, ensuring a visually appealing display even during resizing.

Chapter 4 – Additional Features

. **Dynamic Coloring**: The stroke color of each Bezier curve is dynamically generated, adding a unique visual element to each data point. The big rectangle and small rectangles have different colors which are easy to be distinguished.
. **Text Display**: Along with the graphical representation, textual information is displayed for better understanding. The code includes functionality to display data in the texts. The content of the first line of the input file is not put in the maps which can help the users' distinguish the different types of data.
. **Clearing Old Data**: Before redrawing, old data and shapes are cleared to avoid overlaps.

Chapter 5 – File Handling

The **wordCount** method handles file operations. It reads from a text file (**source.txt**), processes the data, and stores it in a map. This includes: Opening and reading the file line by line. The wordCount() method reads data from the "source.txt" file using a Scanner. It handles file-related exceptions, printing error messages if encountered. Then, it splitting each line to extract relevant information. At last, it stores the extracted data in a Map<String, Integer>, providing an efficient way to manage and access word count information.

Chapter 6 – Exception Handling

Exception handling is implemented in the **wordCount** and **paint** methods to manage IO operations. The try-catch block captures **IOException**, ensuring the program does not crash due to file-related errors. It prints the error message if an exception occurs, aiding in debugging and robustness. By catching and displaying error messages, the code gracefully handles file-related issues, providing feedback to the user about potential problems.

Chapter 7 – My Java Code

```
import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.layout.Pane;

import javafx.scene.paint.Color;

import javafx.scene.shape.*;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.io.File;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;
```

```
public class cw3 extends Application {
	public static void main(String[] args) {
		launch(args);
	}
	@Override
	public void start(Stage stage) throws Exception {
		NumWords pane = new NumWords();
		pane.paint();
		//Create a scene object and add the pane object to it
		Scene scene = new Scene(pane, 500, 500);
		//Set the Stage's Scene and display the Stage
		stage.setScene(scene);
		stage.show();
	}
	public class NumWords extends Pane {
		//Read the data in the txt file and store it in the map
		static Map<String, Integer> map = new HashMap<>();
		public void wordCount() {
			int num = 0;
			try {
				File ex = new File("source.txt");
				Scanner scanner = new Scanner(ex);
				while (scanner.hasNextLine()) {
					String line = scanner.nextLine();
					num = num + 1;
					String[] text = line.split(",");
```

```
                    if (text.length == 0) {
                        num--;
                    }
                    if (text.length == 2 && num == 1) {
                        String first = text[0]+text[1];
                    }
                    if (text.length == 2 && num > 1) {
                        map.put(text[0], Integer.valueOf((text[1])));
                    }
                }
                System.out.println(map);
                scanner.close();
            } catch (IOException ioe) {
                System.out.println(ioe.getMessage());
            }
        }
        public void setWidth(double width) {
            super.setWidth(width);
            paint();
        }
        public void setHeight(double height) {
            super.setHeight(height);
            paint();
        }
        //Paint the sankey diagram
        public void paint() {
```

```
wordCount();
//1.Create a big rectangle
double height = 0.8 * getHeight();
Rectangle large = new Rectangle(100, 60, 0.1 * getWidth(), height);
large.setFill(Color.GREEN);
//2.Create small rectangles and calculate the positions
double val, sum = 0, y = 0;
double sety = large.getY();
for (String key : map.keySet()) {
    sum = sum + map.get(key);
}
getChildren().clear();
for (String key : map.keySet()) {
    val = ((map.get(key)) / sum) * height;
    Rectangle rec = new Rectangle(0.8 * getWidth(), y, 0.1 * getWidth(), val);
    rec.setFill(Color.YELLOW);
    rec.setStroke(Color.color(Math.random(), Math.random(), Math.random()));
    y += (val + 20);
    //3.Draw a Bezier curve
    CubicCurve cubic = new CubicCurve();
    cubic.setStartX(large.getX() + large.getWidth());
    cubic.setStartY(rec.getHeight() / 2 + sety);
    sety = sety+ rec.getHeight();
    cubic.setEndX(rec.getX());
    cubic.setEndY(rec.getY() + rec.getHeight() / 2);
    cubic.setControlX1(cubic.getStartX() + (0.4 * getWidth()));
```

```
                    cubic.setControlY1(cubic.getStartY());

                    cubic.setControlX2(cubic.getEndX() - (0.4 * getWidth()));

                    cubic.setControlY2(cubic.getEndY());

                    cubic.setStrokeWidth(rec.getHeight());

                    cubic.setStroke(rec.getStroke());

                    //4.add the data

                    getChildren().addAll(cubic, rec, new Text(cubic.getEndX(), cubic.getEndY(), key+"
"+map.get(key)));

                }

                try{

                    File ex = new File("source.txt");

                    Scanner scanner = new Scanner(ex);

                    String line1=scanner.nextLine();

                    System.out.println(line1);

                    getChildren().addAll(large,new Text(0.1*getWidth(),0.5*getHeight(),line1));

                } catch (IOException ioe) {

                    System.out.println(ioe.getMessage());

                }

            }

        }

}
```

The above code is the part where the data is read and the Sankey diagram is drawn.
Below code is the part where the data in the .txt file is stored. I have prepared five files.

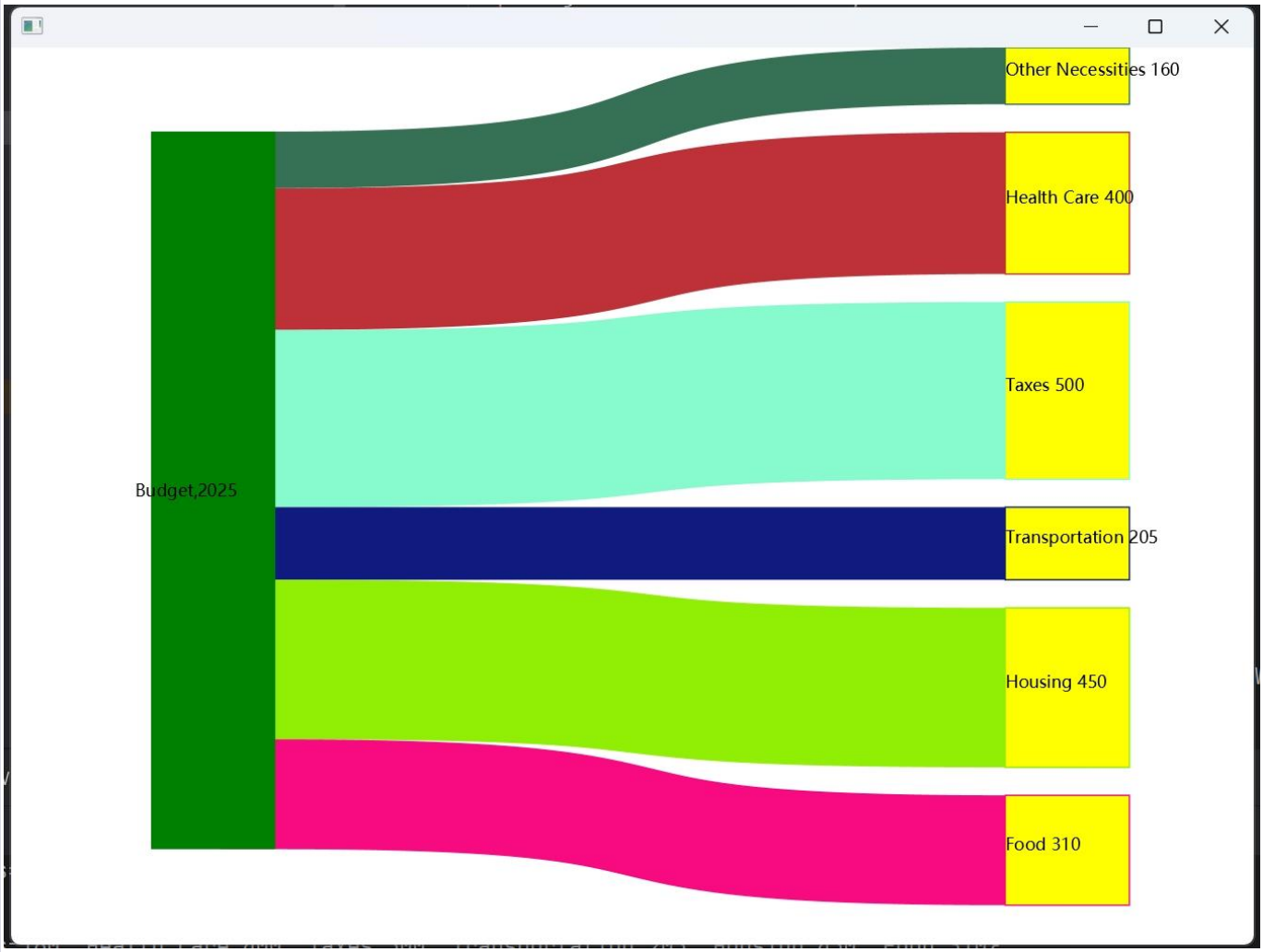
This is the first one:

```
import java.io.BufferedWriter;
```

```
import java.io.File;

import java.io.FileWriter;

import java.io.IOException;


public class DataWriter1 {

	public static void main(String[] args) {

		String[][] Data = {

				{"Budget","2025"},

				{"Taxes","500"},

				{"Housing","450"},

				{"Food","310"},

				{"Transportation","205"},

				{"Health Care","400"},

				{"Other Necessities","160"}

		};


		File file = new File("source.txt");

		try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {

			for (String[] data : Data) {

				bw.write(data[0] + "," + data[1]);

				bw.newLine();

			}

			System.out.println("Data written to source.txt");

		} catch (IOException e) {

			e.printStackTrace();

		}
```

```
    }
}
```

Below is the Sankey diagram of the data which is stored in class DataWriter1:

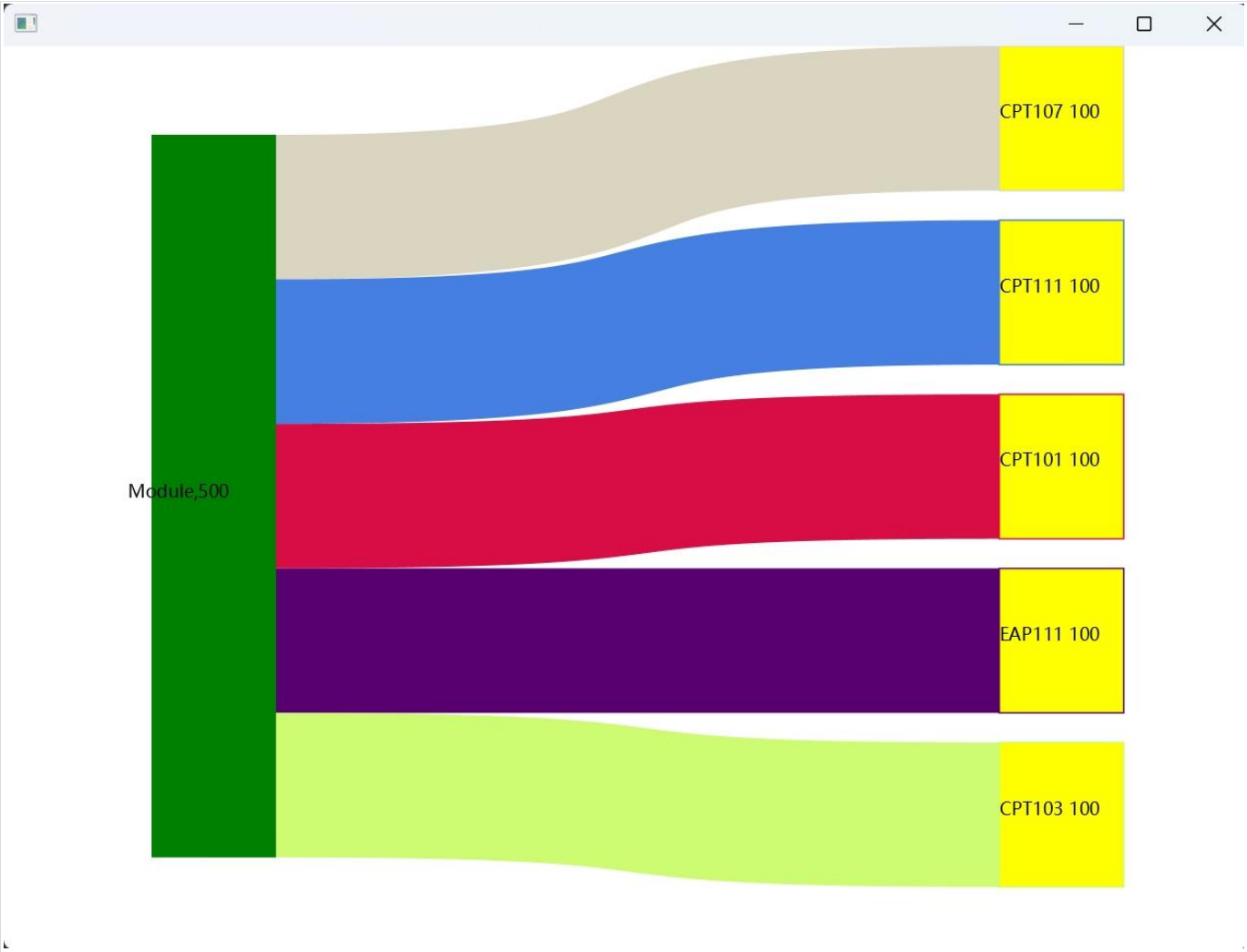


This is the second one:

```
import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;



public class DataWriter2 {

    public static void main(String[] args) {
```

```
        String[][] Data = {
                {"Module","500"},
                {"CPT101","100"},
                {"CPT103","100"},
                {"CPT107","100"},
                {"CPT111","100"},
                {"EAP111","100"}
        };

        File file = new File("source.txt");
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
            for (String[] data : Data) {
                bw.write(data[0] + "," + data[1]);
                bw.newLine();
            }
            System.out.println("Data written to source.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Below is the Sankey diagram of the data which is stored in class DataWriter2:

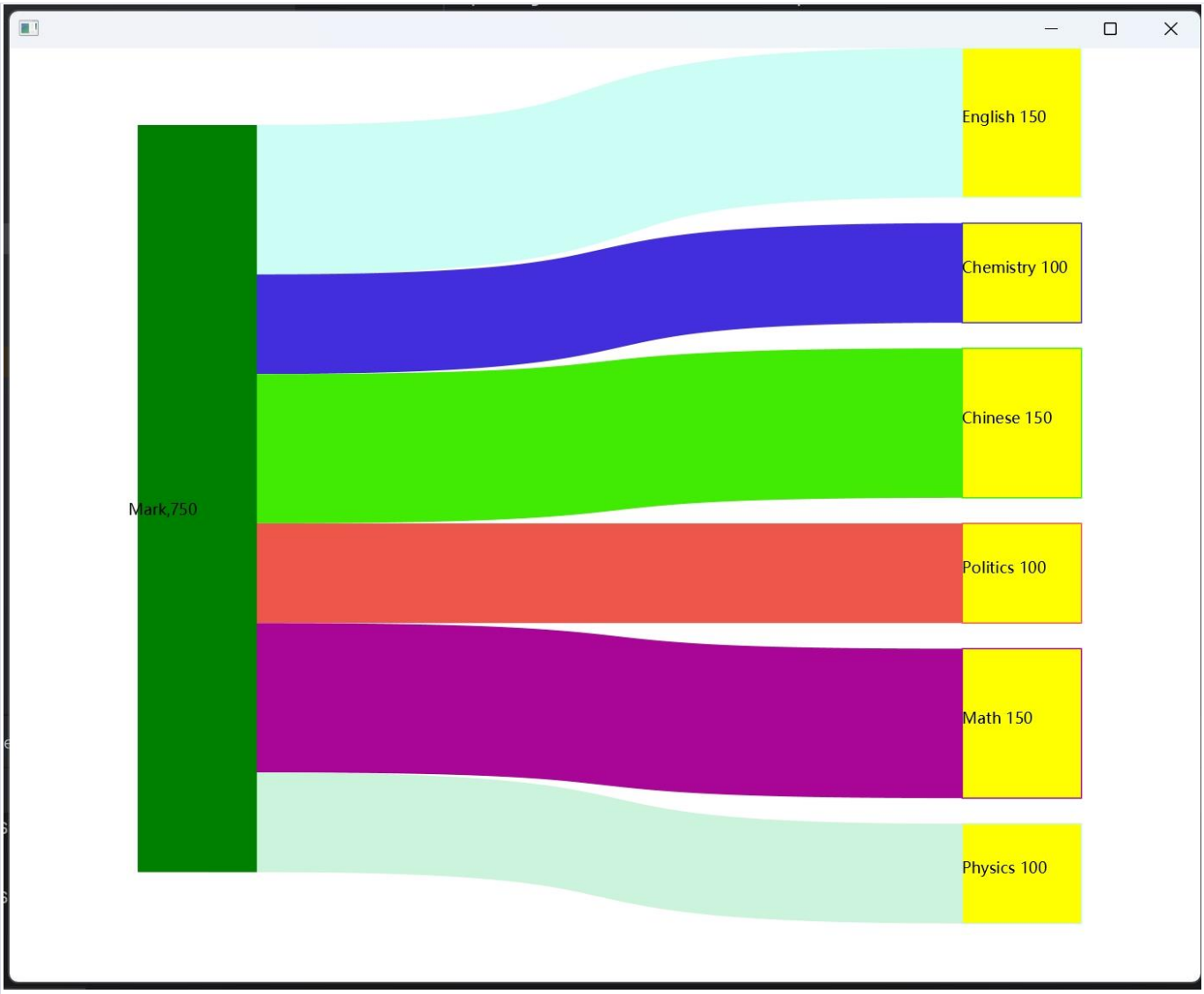


This is the third one:

```
import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;



public class DataWriter3 {

	public static void main(String[] args) {

		String[][] Data = {

				{"Mark","750"},

				{"Chinese","150"},
```

```
                {"Math","150"},
                {"English","150"},
                {"Physics","100"},
                {"Chemistry","100"},
                {"Politics","100"}
        };

        File file = new File("source.txt");
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
            for (String[] data : Data) {
                bw.write(data[0] + "," + data[1]);
                bw.newLine();
            }
            System.out.println("Data written to source.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Below is the Sankey diagram of the data which is stored in class DataWriter3:

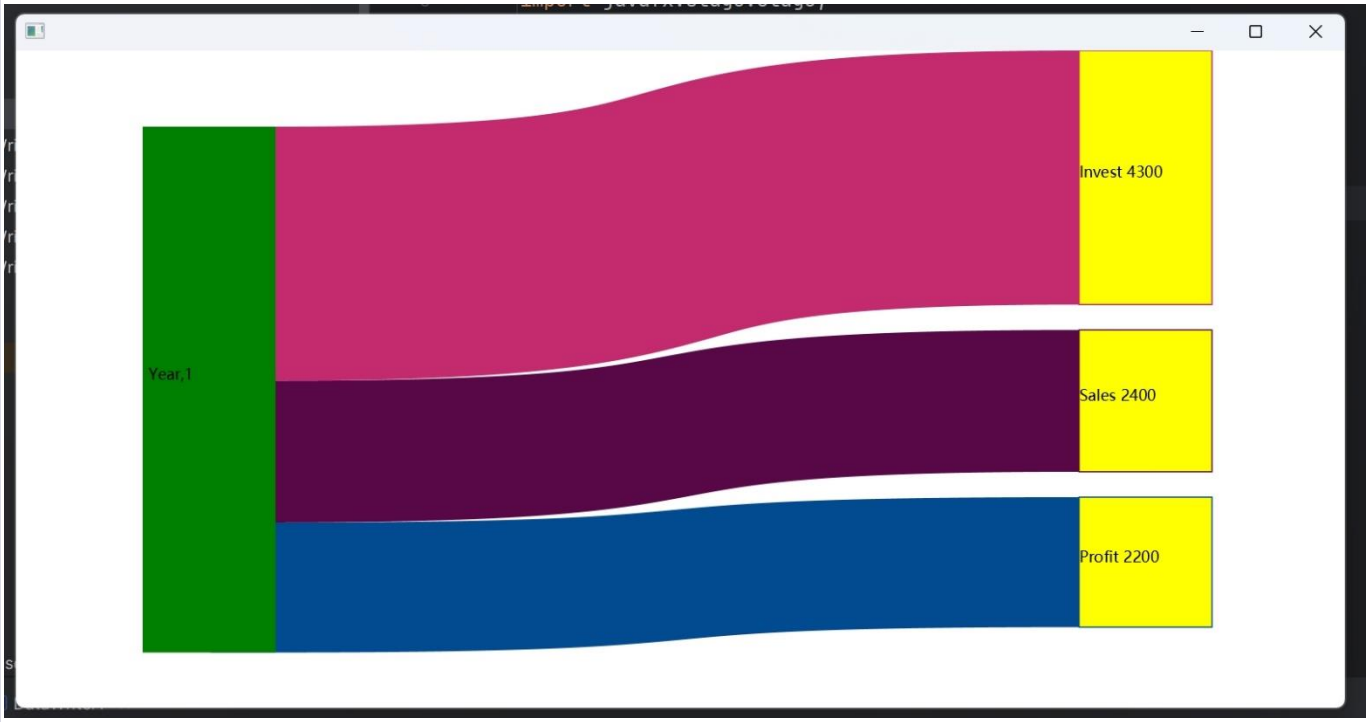


This is the fourth one:

```
import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;


public class DataWriter4 {

	public static void main(String[] args) {

		String[][] Data = {

				{"Year","1"},
```

```
                {"Invest","4300"},
                {"Sales","2400"},
                {"Profit","2200"}
        };

        File file = new File("source.txt");
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
            for (String[] data : Data) {
                bw.write(data[0] + "," + data[1]);
                bw.newLine();
            }
            System.out.println("Data written to source.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Below is the Sankey diagram of the data which is stored in class DataWriter4:

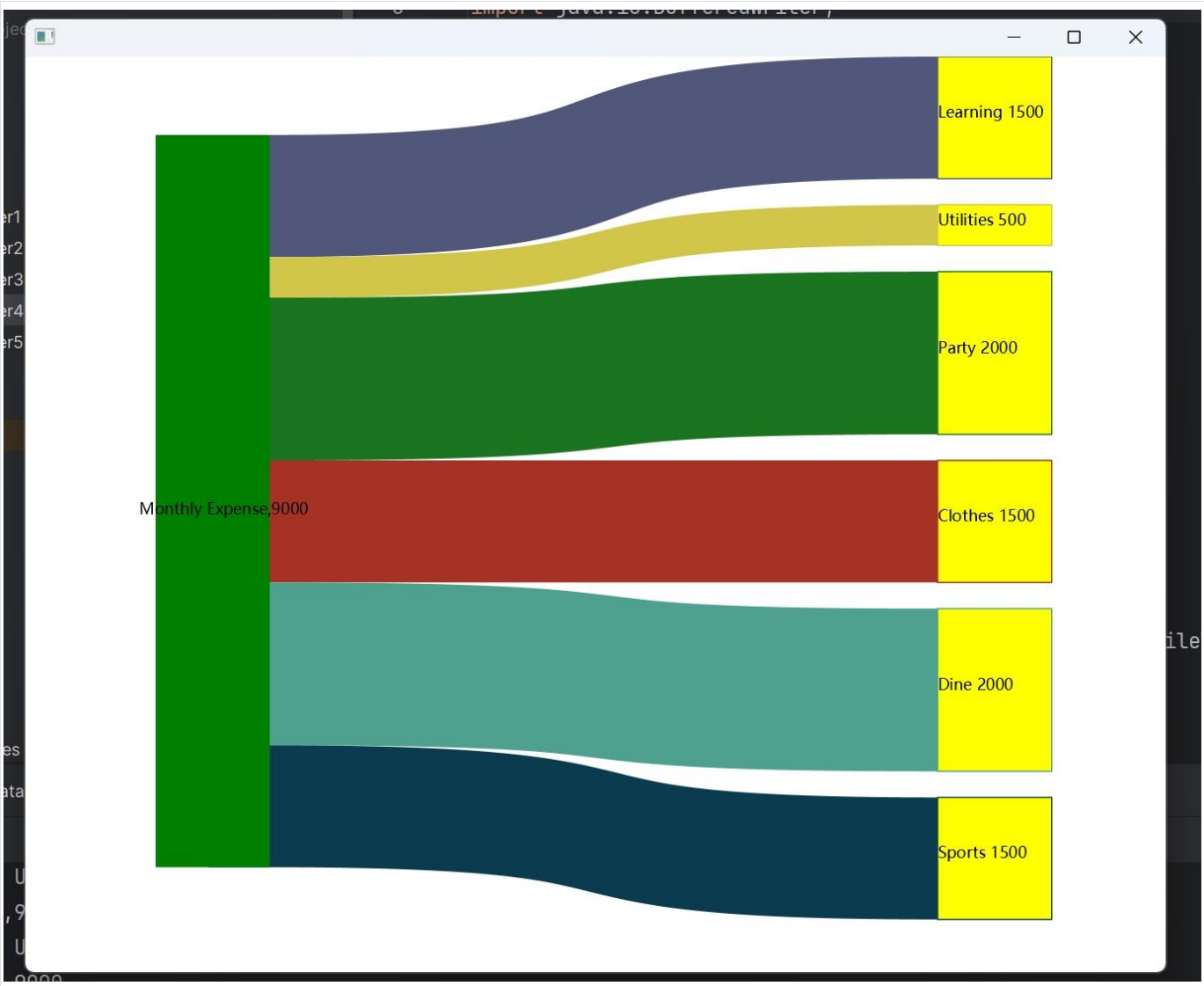


This is the fifth one:

```
import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;



public class DataWriter5 {

	public static void main(String[] args) {

		String[][] Data = {

				{"Monthly Expense","9000"},

				{"Dine","2000"},

				{"Clothes","1500"},

				{"Party","2000"},

				{"Learning","1500"},

				{"Sports","1500"},

				{"Utilities","500"}
```

```
        };


        File file = new File("source.txt");
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
            for (String[] data : Data) {
                bw.write(data[0] + "," + data[1]);
                bw.newLine();
            }
            System.out.println("Data written to source.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Below is the Sankey diagram of the data which is stored in class DataWriter5:



In conclusion, the code demonstrates effective use of object-oriented principles, implements a responsive display algorithm, includes additional features for a richer user experience, and incorporates robust file and exception handling mechanisms.