

# **BUDGET CALCULATOR**

## **A PROJECT REPORT**

*Submitted by*

**SHEKKINAH HEPHZIBAH M (2303811724322104)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 – JAVA PROGRAMMING**

*in*

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by  
AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

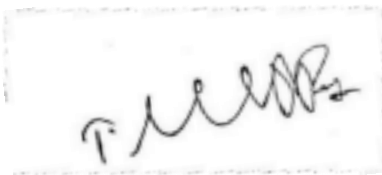
**DECEMBER, 2024**

# **K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

## **BONAFIDE CERTIFICATE**

Certified that this project report on “**BUDGET CALCULATOR**” is the bonafide work of **SHEKKINAH HEPHZIBAH (2303811724322104)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature

**Dr. T. AVUDAIAPPAN M.E.,Ph.D.,**

**HEAD OF THE DEPARTMENT,**

Department of Artificial Intelligence,  
K. Ramakrishnan College of Engineering,  
Samayapuram, Trichy -621 112.



Signature

**Mrs. S. GEETHA M.E.,**

**SUPERVISOR,**

Department of Artificial Intelligence,  
K. Ramakrishnan College of Engineering,  
Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 03.12.24



**INTERNAL EXAMINER**

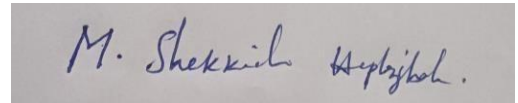


**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**BUDGET CALCULATOR**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.

**Signature**

A rectangular box containing a handwritten signature in blue ink. The signature reads "M. Shekkinah Hephzibah.".

**SHEKKINAH HEPHZIBAH M**

**Place:** Samayapuram

**Date:** 03/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To serve the society by offering top-notch technical education on par with global standards.

## **MISSION OF THE INSTITUTION**

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

## **VISION AND MISSION OF THE DEPARTMENT**

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOS)**

**PEO 1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

**PEO 2:** Provide industry-specific solutions for the society with effective communication and ethics.

**PEO 3:** Hone their professional skills through research and lifelong learning initiatives.

### **PROGRAM OUTCOMES**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

## **ABSTRACT**

The Budget Calculator is a Java application designed to assist users in managing their finances effectively. The application enables users to record, categorize, and manage income and expenses, providing an organized view of monthly and yearly financial data. It allows users to visualize income versus expenses and calculate net savings, promoting better financial planning. With features to add, edit, and delete transactions, the application offers a user-friendly interface for seamless financial management. By leveraging Java's robust programming capabilities, the project delivers an efficient and scalable solution for personal budgeting needs. Its adaptability and wide range of features make it relevant for personal, household, or business use in achieving long-term financial stability and success. The core functionality of the budget calculator revolves around real-time calculations, categorization of expenses, and the visualization of financial trends. The budget calculator can be designed as a standalone application, a web-based platform, or an integrated feature in comprehensive financial management software. It employs intuitive interfaces and user-friendly design to cater to diverse audiences, including individuals with minimal financial expertise.



## TABLE OF CONTENTS

<b>CHAPTER No.</b>	<b>TITLE</b>	<b>PAGE No.</b>
	<b>ABSTRACT</b>	<b>viii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
<b>2</b>	<b>PROJECT METHODOLOGY</b>	<b>2</b>
	2.1 PROPOSED WORK	2
	2.2 BLOCK DIAGRAM	2
<b>3</b>	<b>JAVA PROGRAMMING CONCEPTS</b>	<b>4</b>
	3.1 OBJECT-ORIENTED PROGRAMMING	4
	3.2 GUI COMPONENTS	4
<b>4</b>	<b>MODULE DESCRIPTION</b>	<b>5</b>
	4.1 TRANSACTION MANAGEMENT MODULE	5
	4.2 CATEGORY MANAGEMENT MODULE	5
	4.3 REPORTS MODULE	5
	4.4 SUMMARIZATION MODULE	5
	4.5 DATA PERSISTENCE MODULE	6
<b>5</b>	<b>CONCLUSION</b>	<b>7</b>
	<b>REFERENCES</b>	<b>8</b>
	<b>APPENDICES</b>	<b>9</b>
	Appendix A – Source code	9
	Appendix B – Screen shots	17

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 INTRODUCTION**

The Budget Calculator is designed to help users track their income, monitor expenses, and allocate funds efficiently. By providing real-time insights into financial habits, it empowers users to identify areas of improvement, reduce unnecessary expenditures, and prioritize savings. This project leverages modern technology to create a versatile solution suitable for a wide range of users, from individuals seeking personal financial stability to small business owners aiming to optimize their operational budgets.

### **1.2 OBJECTIVE**

The primary objective of the Budget Calculator Project is to create a versatile, user-friendly tool that aids individuals, households, and businesses in managing their finances effectively. With the increasing complexity of financial responsibilities in modern life, the project seeks to empower users by providing a comprehensive solution for tracking income, monitoring expenses, and achieving financial goals. The calculator is designed to address the common challenges of financial mismanagement, such as overspending, inadequate savings, and a lack of insight into financial habits. By offering a streamlined, intuitive interface, the project aims to make budgeting accessible to users of all financial literacy levels, encouraging better decision-making and fostering long-term financial stability.

## **CHAPTER 2**

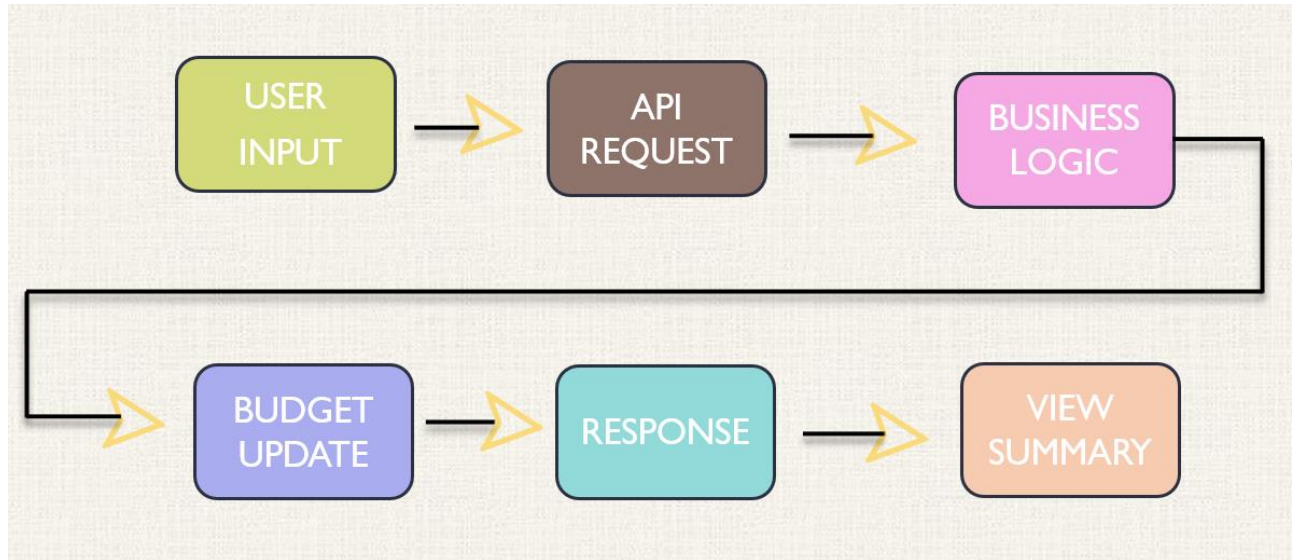
### **PROJECT METHODOLOGY**

#### **2.1 PROPOSED WORK**

The proposed work for the Budget Calculator application aims to create a functional and user-friendly tool for managing personal finances. This includes developing features that enable users to track their income and expenses effectively, calculate net savings, and gain insights through visual and tabular data representations

- **User-Friendly Interface:**
  - A clear and intuitive graphical interface for seamless user interaction.
  - Allow users to view, add, edit, and delete transactions efficiently.
- **Income and Expense Management:**
  - Categorize transactions into income and expenses.
  - Maintain a detailed log of transactions.
- **Financial Summary Generation:**
  - Provide monthly and yearly summaries of financial activities.
  - Display a comparison of income versus expenses.
  - Calculate net savings automatically.
- **Data Persistence:**
  - Enable saving and loading transaction data for future use.
  - Support export to formats like CSV or Excel for external analysis.
- **Error Handling:**
  - Include validations for user inputs to ensure accuracy .

## 2.2 BLOCK DIAGRAM



## **CHAPTER 3**

### **JAVA PROGRAMMING CONCEPTS**

#### **3.1 OBJECT-ORIENTED PROGRAMMING (OOP):**

- Classes and objects to represent income, expenses, and transactions.
- Encapsulation for secure data handling.
- **Java Swing:**
  - Used to create a graphical user interface (GUI) for enhanced user experience.
- **Exception Handling:**
  - To manage errors and ensure smooth operation (e.g., invalid inputs or null entries).
- **File I/O:**
  - For saving and retrieving data to maintain continuity across sessions.

#### **3.2 GUI COMPONENTS:**

- **Text Fields:**
  - Allow users to input income, expenses, and other financial details manually. For example, text fields for entering amounts and descriptions.
- **Dropdown Menus:**
  - Provide predefined categories such as "Housing," "Food," "Transportation," and "Savings," making it easier to categorize expenses.

## **CHAPTER 4**

### **MODULE DESCRIPTION**

#### **4.1 Transaction Management Module:**

- Add new income or expense transactions with details such as date, category and amount.
- Edit existing transactions for corrections.
- Delete transactions to remove incorrect or unwanted entries.

#### **4.2 Category Management Module:**

- Predefined categories (e.g., salary, utilities, groceries) and user-defined categories.
- Allows assigning categories to transactions.

#### **4.3 Summarization Module:**

- Filters and calculates totals for monthly and yearly income and expenses.
- Provides a clear breakdown of spending and savings.

#### **4.4 Report Module:**

- Generates reports showing trends in income and expenses.
- Displays bar charts or pie charts for better visualization.
- Highlights net savings for the selected period.

#### **4.5 Data Persistence Module:**

- Ensures user data is stored in files or a lightweight database.
- Automatically loads data upon applications.
- Save transaction data (income, expenses) to a permanent storage medium.
- Load transaction data when the application starts.
- Allow users to continue from where they left off in the previous session.
- Ensure data integrity and prevent data loss.

## CHAPTER 5

### CONCLUSION

The **Budget Calculator** Java project successfully implements a simple yet effective tool for managing personal finances. It allows users to record their income, track their expenses, and monitor their budget. The project demonstrates the power of Java's object-oriented programming features, such as classes and exception handling, while also showcasing practical use cases for GUI development. This tool can help individuals make informed financial decisions and take control of their financial situation. By providing tools for tracking income and expenses, setting budgets, and generating insightful reports, the application empowers individuals to make informed financial decisions. The user-friendly interface ensures accessibility for everyone, regardless of their financial knowledge. As users engage with the budget calculator, they are likely to develop better spending habits and a clearer understanding of their financial situation. Ultimately, the project aims to enhance users' financial literacy and help them achieve their financial goals, leading to a more secure and organized financial future.



## REFERENCES:

- <https://programthat.wordpress.com/2013/07/25/student-budget-calculator-java-program/>
- <https://github.com/madhu-java/Budget-Planner>
- [https://www.tutorjoes.in/Java\\_example\\_programs/calculate\\_total\\_expenses\\_in\\_java](https://www.tutorjoes.in/Java_example_programs/calculate_total_expenses_in_java)

## APPENDICES

### APPENDIX A – SOURCE CODE

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

class Transaction {
    private String description;
    private double amount;
    private boolean isIncome;

    public Transaction(String description, double amount, boolean isIncome) {
        this.description = description;
        this.amount = amount;
        this.isIncome = isIncome;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getAmount() {
        return amount;
    }
}
```

```
public void setAmount(double amount) {  
    this.amount = amount;  
}
```

```
public boolean isIncome() {  
    return isIncome;  
}
```

```
public void setIncome(boolean isIncome) {  
    this.isIncome = isIncome;  
}
```

```
@Override  
public String toString() {  
    String type = isIncome ? "Income" : "Expense";  
    return type + " | " + description + " | $" + amount;  
}  
}
```

```
public class BudgetCalculatorAWT extends Frame {  
    private ArrayList<Transaction> transactions = new ArrayList<>();  
    private List transactionList;  
    private TextArea summaryArea;
```

```
    public BudgetCalculatorAWT() {  
        setTitle("Budget Calculator");  
        setSize(600, 400);  
        setLayout(new BorderLayout());
```

```

// Transaction List Panel

Panel listPanel = new Panel(new BorderLayout());
transactionList = new List();
listPanel.add(new Label("Transactions:"), BorderLayout.NORTH);
listPanel.add(transactionList, BorderLayout.CENTER);


// Buttons Panel

Panel buttonPanel = new Panel(new GridLayout(1, 5));
Button addButton = new Button("Add");
Button editButton = new Button("Edit");
Button deleteButton = new Button("Delete");
Button viewSummaryButton = new Button("Summary");
Button exitButton = new Button("Exit");


buttonPanel.add(addButton);
buttonPanel.add(editButton);
buttonPanel.add(deleteButton);
buttonPanel.add(viewSummaryButton);
buttonPanel.add(exitButton);


// Summary Panel

summaryArea = new TextArea();
summaryArea.setEditable(false);


// Add Panels to Frame

add(listPanel, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);
add(summaryArea, BorderLayout.EAST);

```

```

// Add Button Listeners
addButton.addActionListener(e -> addTransactionDialog());
editButton.addActionListener(e -> editTransactionDialog());
deleteButton.addActionListener(e -> deleteTransaction());
viewSummaryButton.addActionListener(e -> viewSummary());
exitButton.addActionListener(e -> System.exit(0));

setVisible(true);
}

private void addTransactionDialog() {
    Dialog dialog = new Dialog(this, "Add Transaction", true);
    dialog.setSize(400, 300);
    dialog.setLayout(new GridLayout(5, 2));

    TextField descriptionField = new TextField();
    TextField amountField = new TextField();
    CheckboxGroup typeGroup = new CheckboxGroup();
    Checkbox incomeCheckbox = new Checkbox("Income", typeGroup, true);
    Checkbox expenseCheckbox = new Checkbox("Expense", typeGroup, false);

    dialog.add(new Label("Description:"));
    dialog.add(descriptionField);
    dialog.add(new Label("Amount:"));
    dialog.add(amountField);
    dialog.add(new Label("Type:"));
    Panel typePanel = new Panel(new GridLayout(1, 2));
    typePanel.add(incomeCheckbox);

```

```

typePanel.add(expenseCheckbox);
dialog.add(typePanel);

Button saveButton = new Button("Save");
dialog.add(saveButton);

saveButton.addActionListener(e -> {
    try {
        String description = descriptionField.getText();
        double amount = Double.parseDouble(amountField.getText());
        boolean isIncome = incomeCheckbox.getState();

        transactions.add(new Transaction(description, amount, isIncome));
        transactionList.add(transactions.get(transactions.size() - 1).toString());
        dialog.dispose();
    } catch (NumberFormatException ex) {
        new Dialog(this, "Error", true).add(new Label("Invalid input!"));
    }
});

dialog.setVisible(true);
}

private void editTransactionDialog() {
    int selectedIndex = transactionList.getSelectedIndex();
    if (selectedIndex == -1) {
        showErrorDialog("No transaction selected!");
        return;
    }
}

```

```
Transaction transaction = transactions.get(selectedIndex);
```

```
Dialog dialog = new Dialog(this, "Edit Transaction", true);
```

```
dialog.setSize(400, 300);
```

```
dialog.setLayout(new GridLayout(5, 2));
```

```
TextField descriptionField = new TextField(transaction.getDescription());
```

```
TextField amountField = new TextField(String.valueOf(transaction.getAmount()));
```

```
CheckboxGroup typeGroup = new CheckboxGroup();
```

```
Checkbox incomeCheckbox = new Checkbox("Income", typeGroup,  
transaction.isIncome());
```

```
Checkbox expenseCheckbox = new Checkbox("Expense", typeGroup,  
!transaction.isIncome());
```

```
dialog.add(new Label("Description:"));
```

```
dialog.add(descriptionField);
```

```
dialog.add(new Label("Amount:"));
```

```
dialog.add(amountField);
```

```
dialog.add(new Label("Type:"));
```

```
Panel typePanel = new Panel(new GridLayout(1, 2));
```

```
typePanel.add(incomeCheckbox);
```

```
typePanel.add(expenseCheckbox);
```

```
dialog.add(typePanel);
```

```
Button saveButton = new Button("Save");
```

```
dialog.add(saveButton);
```

```
saveButton.addActionListener(e -> {
```

```

try {
transaction.setDescription(descriptionField.getText());
transaction.setAmount(Double.parseDouble(amountField.getText()));
transaction.setIncome(incomeCheckbox.getState());

transactionList.replaceItem(transaction.toString(), selectedIndex);
dialog.dispose();
} catch (NumberFormatException ex) {
showErrorDialog("Invalid input!");
}
});

dialog.setVisible(true);
}

private void deleteTransaction() {
int selectedIndex = transactionList.getSelectedIndex();
if (selectedIndex == -1) {
showErrorDialog("No transaction selected!");
return;
}

transactions.remove(selectedIndex);
transactionList.remove(selectedIndex);
}

private void viewSummary() {
double totalIncome = 0;
double totalExpenses = 0;

```



```

for (Transaction transaction : transactions) {
    if (transaction.isIncome()) {
        totalIncome += transaction.getAmount();
    } else {
        totalExpenses += transaction.getAmount();
    }
}

```

```

double balance = totalIncome - totalExpenses;

```

```

summaryArea.setText("Total Income: $" + totalIncome + "\n" +
    "Total Expenses: $" + totalExpenses + "\n" +
    "Balance: $" + balance);
}

```

```

private void showErrorDialog(String message) {
    Dialog dialog = new Dialog(this, "Error", true);
    dialog.setSize(300, 100);
    dialog.setLayout(new FlowLayout());
    dialog.add(new Label(message));
    Button closeButton = new Button("Close");
    closeButton.addActionListener(e -> dialog.dispose());
    dialog.add(closeButton);
    dialog.setVisible(true);
}

```

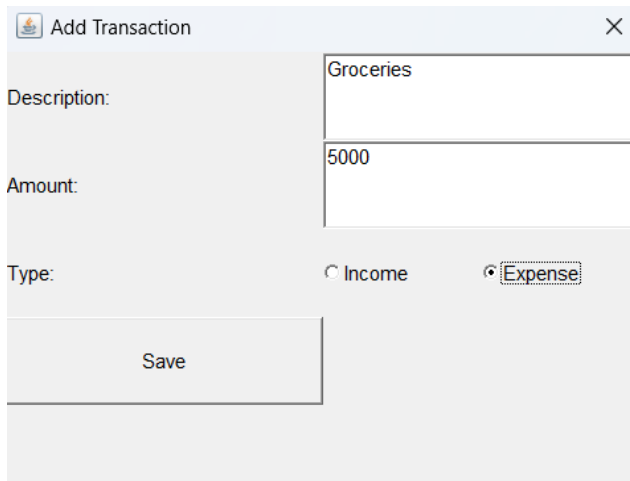
```

public static void main(String[] args) {
    new BudgetCalculatorAWT(); } }

```

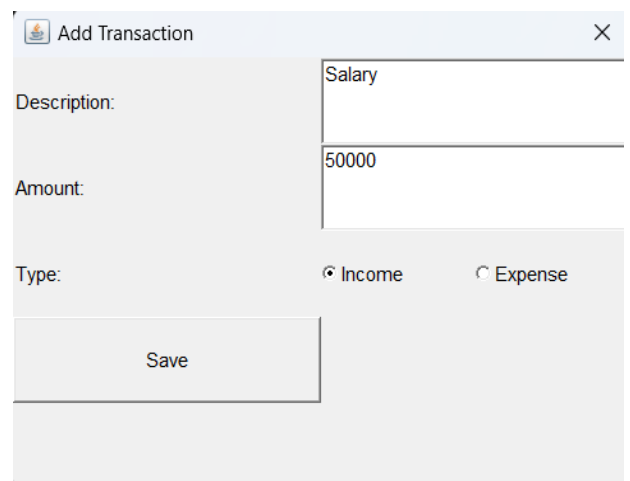
## APPENDIX B - SCREENSHOTS

### ADD TRANSACTION:



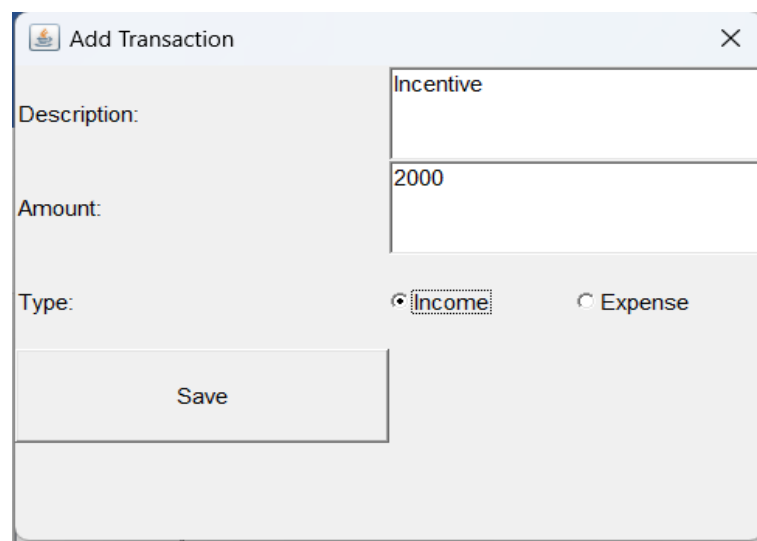
This screenshot shows the 'Add Transaction' dialog box with the following details:

- Description:** Groceries
- Amount:** 5000
- Type:** ☐ Income ☒ Expense
- Save** button



This screenshot shows the 'Add Transaction' dialog box with the following details:

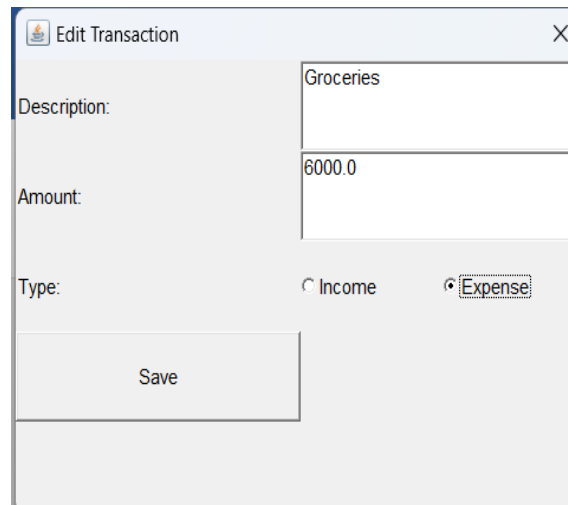
- Description:** Salary
- Amount:** 50000
- Type:** ☒ Income ☐ Expense
- Save** button



This screenshot shows the 'Add Transaction' dialog box with the following details:

- Description:** Incentive
- Amount:** 2000
- Type:** ☒ Income ☐ Expense
- Save** button

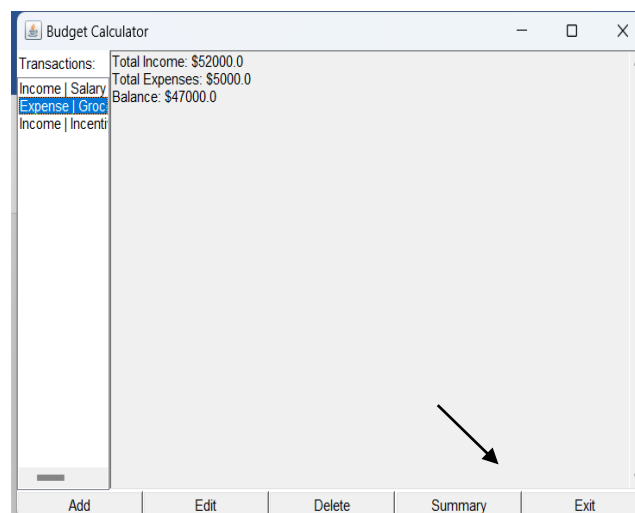
## EDIT TRANSACTION



The 'Edit Transaction' dialog box is shown with the following fields and options:

- Description:** Groceries
- Amount:** 6000.0
- Type:** ☐ Income ☒ Expense
- Save** button

## DELETE TRANSACTION




The 'Budget Calculator' application window displays the following information:

- Transactions:**
  - Income | Salary
  - Expense | Groceries
  - Income | Incentive
- Summary:**
  - Total Income: \$52000.0
  - Total Expenses: \$5000.0
  - Balance: \$47000.0

An arrow points to the 'Summary' button in the bottom navigation bar. The navigation bar includes buttons for Add, Edit, Delete, Summary, and Exit.

## VIEW SUMMARY

 Budget Calculator

Transactions:

Income | Salary

Expense | Grocery

Income | Incentive

Total Income: \$52000.0

Total Expenses: \$5000.0

Balance: \$47000.0