VIVEKANANDA GLOBAL UNIVERSITY

Master of Computer Application

OBJECT ORIENTED PROGRAMMING USING JAVA (PGCSA111)

Project Title: " Tip Calculator Application - For Restaurants

Project Report

PROJECT GUIDE:

Mr. Narayan vyas

SUBMITTED BY:

PARV KHANDELWAL	24CSA3BC095
PARVEEN SAINI	24CSA3BC015
NITYA CHAUDHARY	24CSA3BC079
AARATI PATIL	24CSA3BC047
PANKAJ SHARMA	24CSA3BC076

ACKNOWLEDGEMENT

I have taken this opportunity to express my gratitude and humble regards to the Vivekananda Global University to provide an opportunity to present a project on the” Tip Calculator Application For Restaurants which is a “OBJECT ORIENTED PROGRAMMING USING JAVA” based project.

I would also be thankful to my project guide Mr.Narayan Vyas to help me in the completion of my project and the documentation. I have taken efforts in this project, but the success of this project would not be possible without their support and encouragement.

I would like to thanks our principal sir “Dr R C Tripathi” to help us in providing all the necessary books and other stuffs as and when required. I show my gratitude to the authors whose books has been proved as the guide in the completion of my project I am also thankful to my classmates and friends who have encouraged me in the course of completion of the project.

DECLARATION

We hereby declare that this Project Reporttitled “Build a Quiz Application with Multiple-Choice Questions”. Submitted by us and approved by our project guide, to the Vivekananda Global University, Jaipur is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Project Guide: Mr.Narayan Vya

Table of Contents

1. Introduction

In restaurants, tipping is an essential part of the service industry, directly affecting the income of servers. However, calculating the correct tip can sometimes be a cumbersome and error-prone process for restaurant customers. To address this, the **Tip Calculator Application for Restaurants** was developed in Java. The purpose of this application is to allow customers to easily calculate the tip based on their total bill, the chosen tip percentage, and the number of people sharing the bill.

This project is designed to be user-friendly, intuitive, and highly efficient. The application ensures that users can calculate tips accurately and split bills effortlessly without the need for manual calculations. By implementing a **Graphical User Interface (GUI)** with **Java Swing**, the program allows users to input their bill details, compute tips, and display results in real-time.

---

2. Objectives of the Project

The **Tip Calculator Application for Restaurants** has the following objectives:

- **Simplify tip calculation**: Provide an automatic way to calculate the appropriate tip based on the bill amount and the chosen percentage.
- **Allow bill splitting**: Enable users to split the total bill (including the tip) among multiple individuals.
- **User-friendly interface**: The GUI should be intuitive enough for users with minimal technical knowledge.
- **Error handling**: The application must be capable of handling invalid inputs, ensuring that only correct data is used for calculations.
- **Provide flexibility**: Allow users to select from predefined tip percentages or enter custom percentages.
- **Promote efficiency**: Quickly compute and display results, saving time for users in a restaurant setting.

---

3. Project Architecture Overview

The **Tip Calculator Application** follows a **two-tier architecture** consisting of:

1. **Frontend (UI Layer)**: This is the graphical user interface, built using **Java Swing** components like text fields, buttons, labels, and combo boxes. The UI captures user input (total bill, tip percentage, number of people) and displays the result.
2. **Backend (Logic Layer)**: The backend processes the calculations for the tip, total bill, and the amount per person. It takes inputs, performs necessary mathematical operations, and outputs the results. This logic is encapsulated in methods that are triggered by the user’s actions (button clicks).

This separation ensures that the program is modular and easy to maintain. The UI handles user interactions, while the backend performs the calculations and updates the display.

---

4. Tools and Technologies Used

The **Tip Calculator Application** was developed using the following technologies:

- **Java**: The primary programming language used to build the application. Java is platform-independent, making the application portable across various operating systems.
- **Java Swing**: A GUI toolkit used to design the user interface. Swing provides components like buttons, labels, text fields, and combo boxes, which are essential for the functionality of the application.
- **IDE**: Any modern Java IDE like **IntelliJ IDEA**, **Eclipse**, or **NetBeans** was used to write, test, and debug the code.
- **JDK (Java Development Kit)**: The application was developed using **Java JDK 8 or later**, which provides the necessary tools for compiling and running the Java application.

---

5. System Requirements

5.1 Software Requirements:

- **Java JDK 8 or later**: To compile and run the Java code.
- **IDE**: IntelliJ IDEA, Eclipse, or NetBeans for writing the application code.
- **Java Swing**: This framework is used to build the graphical user interface components.

5.2 Hardware Requirements:

- **Operating System**: The application can be run on any system supporting Java, including **Windows**, **macOS**, and **Linux**.
- **Processor**: An Intel i3 processor or higher should be sufficient for running the application.
- **Memory**: 4 GB of RAM or more.
- **Storage**: 100 MB of free disk space.

---

6. Data Structures for Quiz Management

In this tip calculator application, we primarily use simple data types for handling the input and output:

- **Double**: To store the values for the bill amount, tip percentage, total tip, and total bill.
- **Integer**: To store the number of people splitting the bill and the selected tip percentage.
- **String**: To store user inputs and error messages (e.g., invalid input).

For simplicity, more complex data structures such as arrays, lists, or hash maps are not necessary for this project. The focus is on managing simple numeric data and performing basic calculations.

---

7. User Interface Design

The user interface is designed with simplicity and functionality in mind:

1. **Main Window (JFrame)**: The main window of the application is created using **JFrame**. This window contains all the input fields, buttons, and output labels.
2. **Input Fields (JTextField)**:
   - **Bill Amount**: A text field where the user inputs the total bill.
   - **Number of People**: A text field to enter the number of people sharing the bill.
3. **Dropdown Menu (JComboBox)**: A dropdown menu allows the user to choose a predefined tip percentage (e.g., 10%, 15%, 20%).
4. **Output Labels (JLabel)**: These labels display the results of the calculation, such as the **total tip**, **total bill**, and **amount per person**.
5. **Buttons (JButton)**:
   - **Calculate**: Triggers the calculation of the tip and the total bill, then updates the output labels with the results.
   - **Clear**: Clears all input fields and resets the output labels for a fresh calculation.

The layout is organized in a **FlowLayout** to ensure that the components are displayed clearly and logically.

---

8. Security Features

Since this application does not involve sensitive information (like payment details or personal data), security concerns are minimal. However, it does feature basic **input validation**:

- Ensuring that all user inputs are numeric (i.e., bill amount, number of people, and tip percentage).
- If the user enters invalid data (e.g., letters instead of numbers), the program will show a message dialog with an error message, prompting the user to input valid values.

While this is a simple application, future versions could include features such as **data encryption** for saving calculation history or integration with payment systems.

---

9. Future Enhancements and Scaling

The **Tip Calculator Application** can be further enhanced and scaled by adding:

1. **Custom Tip Percentage**: Allow users to input their preferred tip percentage instead of choosing from predefined values.

2. **Currency Support**: Introduce a feature where users can select their currency (e.g., USD, EUR) to display results in different currencies.
3. **Mobile Version**: Developing a mobile app version for iOS and Android devices to make the calculator available on the go.
4. **History Tracking**: Allow users to save their previous calculations for reference, possibly syncing them across multiple devices.
5. **Localization**: Adding localization for different languages to make the application accessible to users from different regions.

---

10. Deployment Strategy

The deployment process involves packaging the application and making it available for use:

1. **Compiling the Java Application**: The Java source code is compiled into a **JAR (Java Archive)** file that can be executed on any machine with Java installed.
2. **Distribution**: The compiled JAR file can be distributed via email, file sharing services, or hosted on a website for download.
3. **Cross-platform Deployment**: Since Java is cross-platform, the application can be run on any platform (Windows, macOS, or Linux) as long as the correct version of Java is installed.
4. **Web-Based Version**: A web-based version of the tip calculator can be developed in the future using Java Web Start or cloud-based platforms to provide access from any device with an internet connection.

---

11. Basic Enhancements for the Tip Calculator Application

Here are some basic enhancements to improve user experience:

1. **Custom Tip Percentage Input**: Allow the user to enter any desired tip percentage, instead of only predefined options like 10%, 15%, and 20%.
2. **Keyboard Shortcuts**: Implement keyboard shortcuts to enhance the user experience, e.g., pressing "Enter" to trigger the calculation.
3. **Improved Error Handling**: Show clear error messages when the user enters invalid data, such as non-numeric characters or negative values.

---

12. Advanced Enhancements for the Tip Calculator Application

Advanced features can be added to further extend the functionality:

1. **Voice Input**: Add the ability for users to input values (bill amount, number of people) using voice commands.
2. **Integration with Payment Systems**: Allow users to directly pay the bill and tip via integrated payment platforms like PayPal or credit card payments.

3. **Save Calculation History**: Allow users to save previous calculations to track their tipping history or make future calculations easier.
4. **Mobile Application**: Developing mobile apps for iOS and Android platforms would make the tool more accessible and convenient for users.

---

13. Code Implementation

Below is the core implementation for calculating the tip, total bill, and per-person amounts.

java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TipCalculator {

public static void main(String[] args) {
JFrame frame = new JFrame("Restaurant Tip Calculator");
frame.setSize(400, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout(new FlowLayout());

// Create input fields and labels
JLabel billLabel = new JLabel("Bill Amount:");
JTextField billAmountField = new JTextField(15);

JLabel tipLabel = new JLabel("Tip Percentage:");
JComboBox<Integer> tipPercentageBox = new JComboBox<>(new
Integer[]{10, 15, 20});

JLabel peopleLabel = new JLabel("Number of People:");
JTextField numPeopleField = new JTextField(5);

// Output labels
JLabel totalTipLabel = new JLabel("Total Tip: $0.00");
JLabel totalBillLabel = new JLabel("Total Bill (Including Tip): $0.00");
JLabel perPersonLabel = new JLabel("Amount Per Person: $0.00");

// Create buttons
JButton calculateButton = new JButton("Calculate");
JButton clearButton = new JButton("Clear");
```

```
// Add components to the frame
frame.add(billLabel);
frame.add(billAmountField);
frame.add(tipLabel);
frame.add(tipPercentageBox);
frame.add(peopleLabel);
frame.add(numPeopleField);
frame.add(calculateButton);
frame.add(clearButton);
frame.add(totalTipLabel);
frame.add(totalBillLabel);
frame.add(perPersonLabel);

// Action Listeners
calculateButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
try {
double billAmount = Double.parseDouble(billAmountField.getText());
int tipPercentage = (int) tipPercentageBox.getSelectedItem();
int numPeople = Integer.parseInt(numPeopleField.getText());

double tipAmount = billAmount * tipPercentage / 100;
double totalBill = billAmount + tipAmount;
double amountPerPerson = totalBill / numPeople;

totalTipLabel.setText("Total Tip: $" + String.format("%.2f", tipAmount));
totalBillLabel.setText("Total Bill: $" + String.format("%.2f", totalBill));
perPersonLabel.setText("Amount Per Person: $" + String.format("%.2f",
amountPerPerson));
} catch (NumberFormatException ex) {
JOptionPane.showMessageDialog(frame, "Please enter valid numbers!");
}
}
});

clearButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
billAmountField.setText("");
numPeopleField.setText("");
totalTipLabel.setText("Total Tip: $0.00");
totalBillLabel.setText("Total Bill: $0.00");
perPersonLabel.setText("Amount Per Person: $0.00");
}
});

// Display the frame
frame.setVisible(true);
}
}
```

---

14. Expected Output

For a sample input of a $100 bill, 15% tip, and 4 people, the output should be:

- **Total Tip**: $15.00
- **Total Bill (Including Tip)**: $115.00
- **Amount Per Person**: $28.75

---

15. Conclusion

The **Tip Calculator Application for Restaurants** simplifies the process of calculating tips and splitting bills in a restaurant setting. By providing a user-friendly interface and quick calculations, this tool can greatly enhance the dining experience for patrons. Future enhancements and scaling can expand its reach across different platforms, increasing its utility for users worldwide.