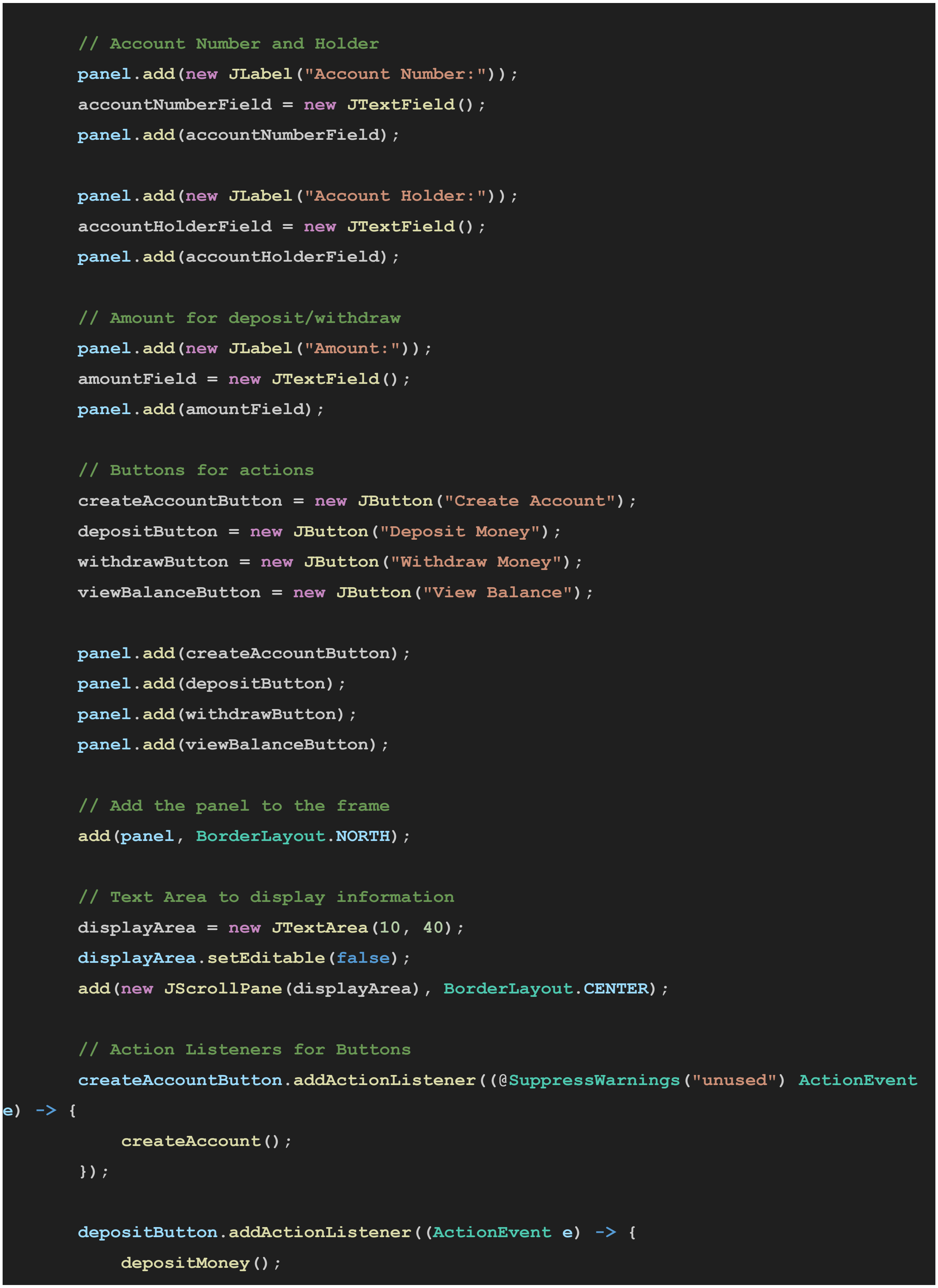
# MINI PROJECT - BANKING MANAGEMENT

## AIM:

To create a java program using the swing programming on the topic of banking management

## PROGRAM:

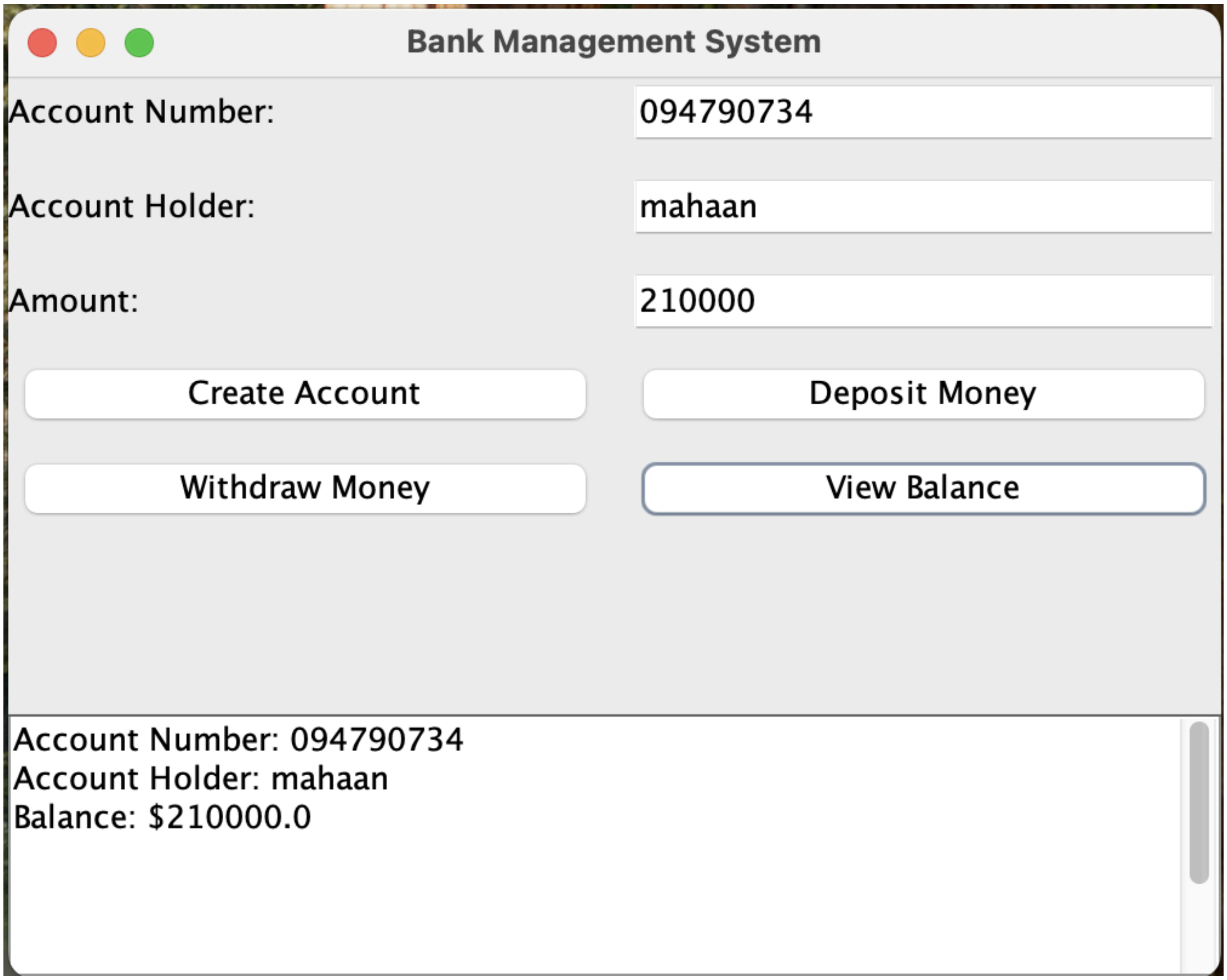
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;

public class BankManagementSystem extends JFrame {

    // Store accounts in a simple HashMap (for demonstration purposes)
    @SuppressWarnings("FieldMayBeFinal")
    private static HashMap<String, BankAccount> accounts = new HashMap<>();
    // GUI Components
    @SuppressWarnings("FieldMayBeFinal")
    private JTextField accountNumberField;
    private final JTextField accountHolderField;
    @SuppressWarnings("FieldMayBeFinal")
    private JTextField amountField;
    private final JTextArea displayArea;
    @SuppressWarnings("FieldMayBeFinal")
    private JButton createAccountButton;
    private final JButton depositButton;
    private final JButton withdrawButton;
    @SuppressWarnings("FieldMayBeFinal")
    private JButton viewBalanceButton;

    // BankAccount Class (Inner Class for simplicity)
    static class BankAccount {
        private final String accountNumber;
        private final String accountHolder;
        private double balance;

        public BankAccount(String accountNumber, String accountHolder) {
            this.accountNumber = accountNumber;
        this.accountHolder = accountHolder;
        this.balance = 0.0;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public String getAccountHolder() {
        return accountHolder;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }

    public boolean withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            return true;
        }
        return false;
    }
}

public BankManagementSystem() {
    // Frame Setup
    setTitle("Bank Management System");
    setSize(500, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    // Panel for forms and buttons
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(7, 2, 10, 10));
```



```
        // Account Number and Holder
        panel.add(new JLabel("Account Number:"));
        accountNumberField = new JTextField();
        panel.add(accountNumberField);

        panel.add(new JLabel("Account Holder:"));
        accountHolderField = new JTextField();
        panel.add(accountHolderField);

        // Amount for deposit/withdraw
        panel.add(new JLabel("Amount:"));
        amountField = new JTextField();
        panel.add(amountField);

        // Buttons for actions
        createAccountButton = new JButton("Create Account");
        depositButton = new JButton("Deposit Money");
        withdrawButton = new JButton("Withdraw Money");
        viewBalanceButton = new JButton("View Balance");

        panel.add(createAccountButton);
        panel.add(depositButton);
        panel.add(withdrawButton);
        panel.add(viewBalanceButton);

        // Add the panel to the frame
        add(panel, BorderLayout.NORTH);

        // Text Area to display information
        displayArea = new JTextArea(10, 40);
        displayArea.setEditable(false);
        add(new JScrollPane(displayArea), BorderLayout.CENTER);

        // Action Listeners for Buttons
        createAccountButton.addActionListener((@SuppressWarnings("unused") ActionEvent e) -> {
            createAccount();
        });

        depositButton.addActionListener((ActionEvent e) -> {
            depositMoney();
        });

        withdrawButton.addActionListener((ActionEvent e) -> {
            withdrawMoney();
        });

        viewBalanceButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                viewBalance();
            }
        });
    }

    // Create Account Action
    private void createAccount() {
        String accountNumber = accountNumberField.getText();
        String accountHolder = accountHolderField.getText();

        if (accountNumber.isEmpty() || accountHolder.isEmpty()) {
            displayArea.setText("Account Number and Holder cannot be empty.");
            return;
        }

        if (accounts.containsKey(accountNumber)) {
            displayArea.setText("Account already exists!");
        } else {
            BankAccount newAccount = new BankAccount(accountNumber, accountHolder);
            accounts.put(accountNumber, newAccount);
            displayArea.setText("Account created successfully!\nAccount Number: " +
accountNumber);
        }
    }

    // Deposit Money Action
    private void depositMoney() {
        String accountNumber = accountNumberField.getText();
        String amountText = amountField.getText();

        if (accountNumber.isEmpty() || amountText.isEmpty()) {
            displayArea.setText("Account Number and Amount cannot be empty.");
            return;
        }

        try {
            double amount = Double.parseDouble(amountText);
            if (amount <= 0) {
                displayArea.setText("Amount must be greater than 0.");
                return;
            }

            BankAccount account = accounts.get(accountNumber);
            if (account != null) {
                account.deposit(amount);
                displayArea.setText("Deposited $" + amount + " to account " + accountNumber + ".\nNew Balance: $" + account.getBalance());
            } else {
                displayArea.setText("Account not found!");
            }
        } catch (NumberFormatException e) {
            displayArea.setText("Invalid amount format!");
        }
    }

    // Withdraw Money Action
    private void withdrawMoney() {
        String accountNumber = accountNumberField.getText();
        String amountText = amountField.getText();

        if (accountNumber.isEmpty() || amountText.isEmpty()) {
            displayArea.setText("Account Number and Amount cannot be empty.");
            return;
        }

        try {
            double amount = Double.parseDouble(amountText);
            if (amount <= 0) {
                displayArea.setText("Amount must be greater than 0.");
                return;
            }

            BankAccount account = accounts.get(accountNumber);
            if (account != null) {
                boolean success = account.withdraw(amount);
                if (success) {
                    displayArea.setText("Withdrew $" + amount + " from account " +
accountNumber + ".\nNew Balance: $" + account.getBalance());
                } else {
                    displayArea.setText("Insufficient funds!");
                }
            } else {
                displayArea.setText("Account not found!");
            }
        } catch (NumberFormatException e) {
            displayArea.setText("Invalid amount format!");
        }
    }

    // View Balance Action
    private void viewBalance() {
        String accountNumber = accountNumberField.getText();

        if (accountNumber.isEmpty()) {
            displayArea.setText("Account Number cannot be empty.");
            return;
        }

        BankAccount account = accounts.get(accountNumber);
        if (account != null) {
            displayArea.setText("Account Number: " + accountNumber + "\nAccount Holder:
" + account.getAccountHolder() + "\nBalance: $" + account.getBalance());
        } else {
            displayArea.setText("Account not found!");
        }
    }

    public static void main(String[] args) {
        // Create and display the GUI
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                BankManagementSystem app = new BankManagementSystem();
                app.setVisible(true);
            }
        });
    }
```

```
}
```

**Output:**



Account Number: 094790734

Account Holder: mahaan

Amount: 210000

```
Account Number: 094790734
Account Holder: mahaan
Balance: $210000.0
```