Project Idea: Basic Banking System

Description:

This project simulates a simple banking system where users can create accounts, deposit money, withdraw money, and check their balance. The project will use basic programming constructs in Java without involving classes or objects. Instead, it will handle all the data using arrays and methods.

Features:

1. **Create a New Account**
   - Store account information like account number, customer name, and balance using arrays.
2. **Deposit Money**
   - Allow users to add money to their account by entering the account number and deposit amount.
3. **Withdraw Money**
   - Allow users to withdraw money from their account, checking for sufficient balance before making the transaction.
4. **Check Balance**
   - Users can check their current account balance by entering their account number.
5. **Display All Accounts**
   - Display all accounts with their details like account number, name, and balance.

Example Code

1. Main Program Structure

```
java
Copy code
import java.util.Scanner;

public class BasicBankingSystem {

    // Arrays to store account data (instead of using objects)
    static int[] accountNumbers = new int[100];
    static String[] customerNames = new String[100];
    static double[] balances = new double[100];

    static int totalAccounts = 0; // To track the number of accounts created

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        // Menu-driven loop
```

```
    do {
        System.out.println("\nBanking System Menu");
        System.out.println("1. Create New Account");
        System.out.println("2. Deposit Money");
        System.out.println("3. Withdraw Money");
        System.out.println("4. Check Balance");
        System.out.println("5. Display All Accounts");
        System.out.println("6. Exit");
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();

        switch (choice) {
            case 1:
                createNewAccount(scanner);
                break;
            case 2:
                depositMoney(scanner);
                break;
            case 3:
                withdrawMoney(scanner);
                break;
            case 4:
                checkBalance(scanner);
                break;
            case 5:
                displayAllAccounts();
                break;
            case 6:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    } while (choice != 6);

    scanner.close();
}

// Function to create a new account
public static void createNewAccount(Scanner scanner) {
    System.out.print("Enter account number: ");
    int accNum = scanner.nextInt();
    System.out.print("Enter customer name: ");
    String name = scanner.next();

    accountNumbers[totalAccounts] = accNum;
    customerNames[totalAccounts] = name;
    balances[totalAccounts] = 0; // Initial balance is 0
    totalAccounts++;

    System.out.println("Account created successfully!");
}

// Function to deposit money
public static void depositMoney(Scanner scanner) {
    System.out.print("Enter account number: ");
    int accNum = scanner.nextInt();
```

```
        System.out.print("Enter amount to deposit: ");
        double amount = scanner.nextDouble();

        int index = findAccount(accNum);
        if (index != -1) {
            balances[index] += amount;
            System.out.println("Deposit successful! New balance: " +
balances[index]);
        } else {
            System.out.println("Account not found!");
        }
    }

    // Function to withdraw money
    public static void withdrawMoney(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accNum = scanner.nextInt();
        System.out.print("Enter amount to withdraw: ");
        double amount = scanner.nextDouble();

        int index = findAccount(accNum);
        if (index != -1) {
            if (balances[index] >= amount) {
                balances[index] -= amount;
                System.out.println("Withdrawal successful! New balance: " +
balances[index]);
            } else {
                System.out.println("Insufficient balance!");
            }
        } else {
            System.out.println("Account not found!");
        }
    }

    // Function to check balance
    public static void checkBalance(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accNum = scanner.nextInt();

        int index = findAccount(accNum);
        if (index != -1) {
            System.out.println("Current balance: " + balances[index]);
        } else {
            System.out.println("Account not found!");
        }
    }

    // Function to display all accounts
    public static void displayAllAccounts() {
        System.out.println("\nAll Accounts:");
        System.out.println("Account Number\tCustomer Name\tBalance");
        for (int i = 0; i < totalAccounts; i++) {
            System.out.println(accountNumbers[i] + "\t\t" + customerNames[i]
+ "\t\t" + balances[i]);
        }
    }
```

```
    // Helper function to find account index by account number
    public static int findAccount(int accNum) {
        for (int i = 0; i < totalAccounts; i++) {
            if (accountNumbers[i] == accNum) {
                return i;
            }
        }
        return -1; // Return -1 if account not found
    }
}
```

2. Explanation of the Program

- **Data Storage**: The program uses arrays (`accountNumbers`, `customerNames`, and `balances`) to store information about multiple accounts. These arrays act as placeholders for each account's data.
- **Procedural Functions**:
  - `createNewAccount`: Adds a new account by filling in details into arrays.
  - `depositMoney`: Deposits money into the corresponding account by updating the balance.
  - `withdrawMoney`: Withdraws money if there's enough balance in the account.
  - `checkBalance`: Prints the balance for a specific account.
  - `displayAllAccounts`: Shows all the accounts created so far.
- **Search Function**: `findAccount` searches for the account index based on the account number. This is used in deposit, withdrawal, and balance checking operations to ensure the correct account is being referenced.
- **Input Handling**: The program uses a `Scanner` to handle user inputs and a `do-while` loop to keep the program running until the user chooses to exit.

3. To-Do List Application

- **Description**: A simple to-do list program that allows the user to add tasks, view tasks, and delete tasks.
- **Key Concepts**: Arrays, loops, methods, conditionals, and user input.
- **Features**:
  - Add tasks to the list.
  - Display the list of tasks.
  - Mark tasks as completed by deleting them.
  - Allow the user to manage multiple tasks.

Example Skeleton:

```
java
Copy code
import java.util.Scanner;

public class ToDoList {
    public static void main(String[] args) {
        String[] tasks = new String[10];
        int taskCount = 0;
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nTo-Do List:");
            System.out.println("1. Add Task");
            System.out.println("2. View Tasks");
            System.out.println("3. Delete Task");
            System.out.println("4. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    if (taskCount < tasks.length) {
                        System.out.print("Enter new task: ");
                        tasks[taskCount++] = scanner.nextLine();
                    } else {
                        System.out.println("Task list is full!");
                    }
                    break;
                case 2:
                    System.out.println("Your Tasks:");
                    for (int i = 0; i < taskCount; i++) {
                        System.out.println((i + 1) + ". " + tasks[i]);
                    }
                    break;
                case 3:
                    System.out.print("Enter task number to delete: ");
                    int taskNum = scanner.nextInt();
                    if (taskNum > 0 && taskNum <= taskCount) {
                        for (int i = taskNum - 1; i < taskCount - 1; i++) {
                            tasks[i] = tasks[i + 1];
```

```
                        }
                        taskCount--;
                    } else {
                        System.out.println("Invalid task number!");
                    }
                    break;
                case 4:
                    System.out.println("Exiting...");
                    return;
                default:
                    System.out.println("Invalid option! Try again.");
            }
        }
    }
}
```