Bank Application Development Task

Class Structure & Requirements

1. Business Logic Class (BLC) - `BankApplication`

This class will be responsible for managing customer account details and performing transactions.

Attributes (Instance Variables)

- `customerName` *(String)* → Stores the customer's name.
- `customerAddress` *(String)* → Stores the customer's address.
- `phoneNumber` *(long)* → Stores the customer's contact number.
- `balance` *(double)* → Stores the customer's current account balance.

Constructor

- A **parameterized constructor** to initialize the above attributes when creating a new account.
- After successful creation, the program should print:
  "Account Created Successfully".

Methods

1. **`deposit(double amount)`**
   - Adds the given amount to the balance.
   - Displays a success message with the updated balance.
   - Edge Case: If the deposit amount is **negative or zero**, display an appropriate error message.
2. **`withdraw(double amount)`**
   - Deducts the given amount from the balance if sufficient funds are available.
   - If the withdrawal is successful, display the remaining balance.
   - Edge Cases:
     - If the withdrawal amount is **zero or negative**, print **"Invalid withdrawal amount"**.
     - If the withdrawal amount is **greater than the available balance**, print **"Insufficient funds"**.
3. **`showBalance()`**
   - Displays the current balance.

---

2. Execution Logic Class (ELC)

This class contains the `main` method and is responsible for **interacting with the user** via console inputs.

Steps to Implement

1. **Take user input** for:
   - Name
   - Address
   - Phone Number
   - Initial deposit amount

    Create an object of `BankApplication` with these values.

2. **Display Menu Options**
   The program should repeatedly show the following menu and prompt the user to select an option:

   ```
   **** Select an Option from Below ****
   1. Withdraw
   2. Deposit
   3. Show Balance
   4. Exit
   Enter your option [1-4]:
   ```

3. **Process User Input**
   - If the user enters `1`:
     - Ask for the withdrawal amount.
     - Call `withdraw(amount)` and display the result.
   - If the user enters `2`:
     - Ask for the deposit amount.
     - Call `deposit(amount)` and display the result.
   - If the user enters `3`:
     - Call `showBalance()` and display the balance.
   - If the user enters `4`:
     - Print **"Thank You!"** and exit the program.
   - If the user enters an **invalid option**, print **"Invalid option"** and re-display the menu.

4. **Loop Until Exit**
   - The program should keep running **until the user selects option 4 (Exit)**.

Example Execution Flow

```
Enter your name: Hemanth
Enter your address: xxx
Enter your phone number: 1234567
Enter your initial deposit balance: 5000
Account Created Successfully!

**** Select an Option from Below ****
1. Withdraw
2. Deposit
3. Show Balance
4. Exit
Enter your option [1-4]: 1

Enter your withdrawal amount: 1000
Withdrawal successful, remaining balance: 4000

**** Select an Option from Below ****
1. Withdraw
2. Deposit
```

```
3. Show Balance
4. Exit
Enter your option [1-4]: 34324

Invalid option

**** Select an Option from Below ****
1. Withdraw
2. Deposit
3. Show Balance
4. Exit
Enter your option [1-4]: 2

Enter deposit money: 3000
Deposit success, available balance: 7000

**** Select an Option from Below ****
1. Withdraw
2. Deposit
3. Show Balance
4. Exit
Enter your option [1-4]: 3

Your balance is 7000

**** Select an Option from Below ****
1. Withdraw
2. Deposit
3. Show Balance
4. Exit
Enter your option [1-4]: 4

Thank You!
```