Bank Account Menu Lab

Objective

Create a Java program that simulates a simple bank account menu using **while loops** and **switch statements**.

Requirements

Core Functionality

Your program must implement a menu-driven bank account application with the following features:

1. **While Loop**: The program should continue running until the user chooses to exit
2. **Switch Statement**: Handle menu choices using a switch statement
3. **Menu Options**:
    - Option 1: Add Money
    - Option 2: Withdraw Money
    - Option 3: Check Balance
    - Option 4: Exit the program

Bank Account Logic

- **Balance**: Store account balance as a `double` (e.g., `double balance = 0.0;`)
- **Add Money**: Prompt for amount, add if positive, show error if negative
- **Withdraw Money**: Prompt for amount, subtract if positive and sufficient funds, show error otherwise
- **Check Balance**: Display current balance formatted to exactly 2 decimal places

Program Structure

```java
import java.util.Scanner;

// DO NOT CHANGE THE CLASS NAME, IT WILL BREAK THE AUTO GRADER
public class BankAccountMenu {
    public static void main(String[] args) {
        // Your implementation here
    }
}
```

⚠ **Important**: Do not change the class name `BankAccountMenu` as it will break the autograder!

Getting Started

Menu Display

- Display a clear menu with options 1-4
- Show the menu repeatedly until the user exits
- Format: "--- Bank Account Menu ---"

Input Handling

- Use `Scanner` to read user input
- Use `scanner.nextInt()` for menu choices
- Use `scanner.nextDouble()` for money amounts
- Handle the input in a switch statement

Money Formatting

- Display money amounts to exactly 2 decimal places
- Include dollar signs ($) in output
- Example: "Added $50.00" or "New balance: $50.00"

Error Handling

- **Add Money**: If amount ≤ 0, print error message and don't change balance
- **Withdraw Money**: If amount ≤ 0 or amount > balance, print error message and don't change balance
- **Insufficient Funds**: Print "Insufficient funds" when trying to withdraw more than available

Example Output

```
--- Bank Account Menu ---
1. Add Money
2. Withdraw Money
3. Check Balance
4. Exit
Enter your choice: 1
Enter amount to add: 50
Added $50.00
New balance: $50.00

--- Bank Account Menu ---
1. Add Money
2. Withdraw Money
3. Check Balance
4. Exit
Enter your choice: 3
Current balance: $50.00

--- Bank Account Menu ---
1. Add Money
2. Withdraw Money
3. Check Balance
4. Exit
Enter your choice: 2
Enter amount to withdraw: 25
Withdrew $25.00
New balance: $25.00

--- Bank Account Menu ---
1. Add Money
2. Withdraw Money
3. Check Balance
4. Exit
Enter your choice: 2
Enter amount to withdraw: 50
Insufficient funds

--- Bank Account Menu ---
1. Add Money
2. Withdraw Money
3. Check Balance
4. Exit
Enter your choice: 4
Goodbye!
```

Try using IntelliJ or Bluejay to Complete This Lab

What Must Stay the Same

- **Class name**: `BankAccountMenu`
- **Method signature**: `public static void main(String[] args)`
- **Balance variable**: Must be a `double` type
- **Menu options**: 1, 2, 3, 4 with the specified functionality
- **Money formatting**: Exactly 2 decimal places with $ symbol

Compilation Errors

- Make sure your class name is `BankAccountMenu`
- Check that you have proper Java syntax
- Make all braces `{}` are properly matched
- Ensure Scanner is properly imported

Runtime Errors

- Make sure you're using `scanner.nextInt()` for menu choices
- Use `scanner.nextDouble()` for money amounts
- Check that your while loop has a proper exit condition
- Verify that your switch statement handles all cases

Learning Objectives

By completing this lab, you will demonstrate understanding of:

- **While loops**: Creating repeating program flow

- **Switch statements**: Handling multiple conditional branches
- **Scanner input**: Reading different types of user input (int, double)
- **Input validation**: Checking for valid amounts and sufficient funds
- **Control flow**: Combining loops and conditionals
- **Problem solving**: Breaking down complex requirements into manageable parts