PROJECT 2 (Chapters 2-3)

Objective:

Develop a Python program that manages a simple bank account. The program will simulate actions like depositing money, withdrawing money, and checking the account balance, with certain conditions that involve decision structures and Boolean logic. The program will be structured using methods to ensure modularity, readability, and maintainability.

Instructions:

1. **Define Named Constants:**
    - Define constants like MIN_BALANCE (minimum required balance for transactions) and OVERDRAFT_FEE (a fee applied if an account goes below a minimum balance).
2. **Main Menu:**
    - The program should display a menu of actions:
        - **1. Check Balance**
        - **2. Deposit**
        - **3. Withdraw**
        - **4. Exit**
    - Use decision structures to handle the user's input.
3. **Account Actions in Methods:**
    - **check_balance():** Display the current account balance.
    - **deposit(balance):** Prompt the user for an amount to deposit. Add this amount to the current balance and display the new balance. Return the updated balance.
    - **withdraw(balance):** Prompt the user for an amount to withdraw. Before completing the withdrawal, check if the amount is available in the balance.
        - If the balance after withdrawal goes below MIN_BALANCE, deduct the OVERDRAFT_FEE from the balance.
        - If the withdrawal amount exceeds the current balance, display a warning message and prevent the withdrawal. Return the updated balance.
    - **display_menu():** Display the menu and return the user's choice.
    - **exit_program():** Display a farewell message and terminate the program.
4. **Conditional Logic for Overdraft Protection:**
    - If the balance is less than MIN_BALANCE after a withdrawal, apply an overdraft fee.
5. **Exiting the Program:**

- The program should keep running until the user chooses the "Exit" option.

Method Requirements:

- **All code must be placed inside methods.** Each method should be responsible for a single task and should be complete and self-documented.
- Methods should be kept as simple as possible.
- The main() function should only contain method calls and handle data movement between methods.

Requirements:

- **Methods:** Every functional task is broken down into a method, which is self-contained and has a clear responsibility. The main() function only manages calling the methods and moving data (balance) between them.
- **Decision Structures:** Use decision structures (if, else, if-elif-else) to handle the logic for deposits, withdrawals, overdraft protection, and menu navigation.
- **Boolean Logic:** Apply relational operators (>, >=, <, <=, ==, !=) to check conditions like overdraft and balance.
- **User Input Validation:** Ensure valid numeric inputs for deposits and withdrawals.

Final guidance:

Ensure that each method is simple, complete, and well-documented with clear docstrings.

No external libraries are allowed.

This project reinforces all concepts from **Chapter 3**: decision structures, Boolean expressions, and logical operators, as well as basic input/output and variable handling from **Chapter 2**.

SAMPLE PROGRAM OUTPUT

```
----------------------------------------
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
----------------------------------------
Enter your choice: 2
Enter the deposit amount: 200.00
Deposit successful! Your new balance is $200.00.

----------------------------------------
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
----------------------------------------
Enter your choice: 3
Enter the withdrawal amount: 150.00
Withdrawal successful! Your new balance is $15.00.
Overdraft! A fee of $35.00 has been applied.
Your new balance is -$20.00.

----------------------------------------
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
----------------------------------------
Enter your choice: 1
Your current balance is -$20.00.
```