1)Use Case: Bank Account Management

Description:

The bank account management system allows customers to perform deposit and withdrawal operations on their bank accounts. It provides an interface for users to interact with their accounts and maintains a list of bank account numbers.

Preconditions:

- The system is running and ready to accept user input.

Postconditions:

- The system updates the account balance based on user actions.

- The system displays the updated balance and maintains a list of bank account numbers.

Main Flow:

1. The system prompts the user to enter their bank account number.

2. The user enters their bank account number.

3. The system creates an instance of the BankAccount class with the provided account number.

4. The system adds the account number to the list of bank accounts.

5. The system prompts the user to choose an action (deposit or withdrawal).

6. The user enters their choice.

7. If the user chooses deposit:

a. The system prompts the user to enter the amount to deposit.

b. The user enters the deposit amount.

c. The system calls the deposit() method of the BankAccount instance to perform the deposit operation.

d. The system displays the updated balance after the deposit.

8. If the user chooses withdrawal:

a. The system prompts the user to enter the amount to withdraw.

b. The user enters the withdrawal amount.

c. The system calls the withdraw() method of the BankAccount instance to perform the withdrawal operation.

d. The system displays the updated balance after the withdrawal.

9. The system displays the current balance of the account.

10. The system displays the list of bank account numbers.

Alternative Flow:

- If the user enters an invalid choice (neither 1 nor 2) in step 7 or step 8:

a. The system displays an error message indicating an invalid choice.

b. The flow returns to step 5, prompting the user to choose an action again.

Exceptions:

- If the withdrawal amount entered by the user in step 8b exceeds the account balance:

a. The system displays an error message indicating insufficient balance.

b. The flow returns to step 5, prompting the user to choose an action again.

--------------------------------------------------------------------------------------------------------------------------

2)Certainly! Here's a use case for the bank account program described above.

---

\*\*Use Case: Bank Account Management System\*\*

\*\*Use Case Description:\*\*

This use case describes the basic functionalities of a Bank Account Management System, allowing users to perform deposit and withdrawal operations on their accounts.

\*\*Actors:\*\*

- User

- Bank

\*\*Preconditions:\*\*

- The user must have a valid bank account number.

- The Bank Account Management System must be operational.

\*\*Flow of Events:\*\*

1. User launches the Bank Account Management System.

2. The system prompts the user to enter their bank account number.

3. User enters their bank account number.

4. The system creates an instance of the BankAccount class with the provided account number and sets the account balance to 0.

5. The system adds the account number to the list of bank accounts.

6. The system presents the user with the available actions:

- Deposit

- Withdraw

7. User selects an action by entering the corresponding choice:

- If the user selects "Deposit":

- The system prompts the user to enter the deposit amount.

- User enters the amount to deposit.

- The system calls the `deposit()` method of the BankAccount instance, passing the deposit amount.

- The system updates the account balance accordingly.

- The system displays a success message with the updated balance.

- If the user selects "Withdraw":

- The system prompts the user to enter the withdrawal amount.

- User enters the amount to withdraw.

- The system calls the `withdraw()` method of the BankAccount instance, passing the withdrawal amount.

- The system checks if the account has sufficient funds for the withdrawal:

- If the account balance is greater than or equal to the withdrawal amount:

- The system deducts the withdrawal amount from the account balance.

- The system displays a success message with the updated balance.

- If the account balance is less than the withdrawal amount:

- The system displays an error message indicating insufficient funds.

8. The system displays the current balance of the account.

9. The system presents the user with an option to perform another transaction or exit the program:

- If the user chooses to perform another transaction:

- The system redirects the user to step 6 to select an action.

- If the user chooses to exit the program:

- The system terminates.

\*\*Postconditions:\*\*

- The user's bank account balance is updated after each successful deposit or withdrawal operation.

- The Bank Account Management System maintains a list of bank account numbers for reference.

- The system allows the user to perform multiple transactions during a session.

---

This use case outlines the basic flow of events for the Bank Account Management System. It covers the process of creating a bank account, performing deposit and withdrawal operations, and maintaining a list of bank account numbers. Depending on the specific requirements, additional features like account balance inquiries, transaction history, and user authentication could be added to enhance the functionality of the system.

3)Mycode----------------------------------------

Certainly! Here's an updated use case for the modified code:

---

\*\*Use Case: Bank Account Management System\*\*

\*\*Use Case Description:\*\*

This use case describes the basic functionalities of a Bank Account Management System, allowing users to create bank accounts, perform deposit and withdrawal operations, and retrieve the account balance.

\*\*Actors:\*\*

- User

- Bank

\*\*Preconditions:\*\*

- The Bank Account Management System must be operational.

\*\*Flow of Events:\*\*

1. User launches the Bank Account Management System.

2. User is prompted to enter their bank account number.

3. User enters their bank account number.

4. The system creates an instance of the BankAccount class with the provided account number and initializes the account balance to 0.

5. The system adds the account number to the list of bank accounts.

6. The system displays a success message indicating that the account has been created.

7. The system presents the user with the available actions:

- Deposit

- Withdraw

8. User selects an action by entering the corresponding choice:

- If the user selects "Deposit":

- The system prompts the user to enter the deposit amount.

- User enters the amount to deposit.

- The system calls the `deposit()` method of the BankAccount instance, passing the deposit amount.

- The system updates the account balance accordingly.

- The system displays a success message with the updated balance.

- If the user selects "Withdraw":

- The system prompts the user to enter the withdrawal amount.

- User enters the amount to withdraw.

- The system calls the `withdraw()` method of the BankAccount instance, passing the withdrawal amount.

- The system checks if the account has sufficient funds for the withdrawal:

- If the account balance is greater than or equal to the withdrawal amount:

- The system deducts the withdrawal amount from the account balance.

- The system displays a success message with the updated balance.

- If the account balance is less than the withdrawal amount:

- The system displays an error message indicating insufficient funds.

9. The system retrieves the current balance of the account using the `get\_balance()` method.

10. The system displays the account number and the current balance.

11. The system terminates.

\*\*Postconditions:\*\*

- The user can create a bank account and perform deposit or withdrawal operations on that account.

- The Bank Account Management System maintains a list of bank account numbers for reference.

- The system provides the user with the current balance of the account.

---

This use case outlines the basic flow of events for the Bank Account Management System. It covers the process of creating a bank account, performing deposit and withdrawal operations, and retrieving the account balance. The system also maintains a list of bank account numbers for reference. Additional features like account balance inquiries, transaction history, and user authentication could be added to enhance the functionality of the system.