C++ Assignment: Exception Handling in a Banking System

Assignment Description:

You will create a simple banking system in C++ to practice exception handling, exception classes, and exception handling techniques. This system will simulate depositing, withdrawing, and checking account balances. Your program should handle exceptions such as overdrafts, invalid amounts, and other possible errors using custom exception classes and the `try`/`catch` mechanism.

Objectives:

1. Implement basic C++ exception handling using `try`/`catch`.

2. Create and throw custom exception classes.

3. Handle exceptions effectively using different strategies: fix, log, rethrow, and terminate.

4. Understand stack unwinding by nesting function calls that may throw exceptions.

Assignment Requirements:

1. Define Custom Exception Classes:

- Create two custom exception classes:

- `InvalidAmountException`: Triggered if a user tries to deposit or withdraw an invalid amount (like negative values).

- `InsufficientFundsException`: Triggered when a withdrawal exceeds the account balance.

2. Define a BankAccount Class:

- Implement a class `BankAccount` that includes:

- Attributes:

private:

string accountHolderName;

string bankName;

int accountNumber;

balance` (double)

- **Methods:**

- `deposit(double amount)`: Adds the amount to the balance. Should throw `InvalidAmountException` for invalid deposits.

- `withdraw(double amount)`: Deducts the amount from the balance. Should throw `InsufficientFundsException` if the withdrawal amount exceeds the balance or `InvalidAmountException` if the amount is negative.

- `displayBalance()`: Prints the current balance.

3. Handle Exceptions in Main Program:

- Use `try`/`catch` blocks in `main()` to:

- Catch exceptions from `deposit` and `withdraw` operations.

- Fix the Error and Continue: If an invalid amount is given, prompt the user to enter a valid amount and retry.

- Log the Error and Continue: If an overdraft occurs, log a message without terminating the program.

- Terminate the Program: Use `terminate()` for any unexpected errors that don't match known exception types.

4. Implement Rethrowing:

- Within the `withdraw()` method, use a `try`/`catch` block to catch any thrown exceptions. Rethrow the exception to be handled at the higher level in `main()`.

5. Simulate Stack Unwinding:

- Create an additional function `performTransaction()` that calls the `deposit()` and `withdraw()` methods. Call `performTransaction()` in `main()`. Test stack unwinding by having `performTransaction()` throw an exception, which should be caught in `main()`.

Detailed Instructions:

1. Define Exception Classes:

- Implement `InvalidAmountException` and `InsufficientFundsException` with appropriate error messages.

2. Define BankAccount Class:

- Define `BankAccount` with methods as described.

- Ensure `withdraw()` and `deposit()` have proper exception handling and rethrow where applicable.

3. Exception Handling in Main Program:

- In `main()`, create a `BankAccount` instance and attempt multiple transactions with varying amounts.

- Use `try`/`catch` in `main()` to:

  - Fix errors by allowing the user to reenter values if an invalid amount is entered.

  - Log overdraft errors and prompt the user for another transaction.

  - Handle any unexpected exceptions by terminating the program.

4. Example Output:

- Test cases should cover:

  - Normal deposit and withdrawal

  - Invalid deposits/withdrawals (negative numbers)

  - Attempted overdraft

  - Unexpected error handling

```
Enter your name: Sumanth Burugula
Enter bank name: US Bank
Enter account number: 1738273
Enter initial balance: $10000
Enter deposit amount: $2015
Enter withdraw amount: $3043

Bank Account Summary:
Account Holder: Sumanth Burugula
Bank Name: US Bank
Account Number: 1738273
Current Balance: $8972

Would you like to enter another account? (Y/N): y
Enter your name: Emily E
Enter bank name: SOfi
Enter account number: 2772772
Enter initial balance: $5000
Enter deposit amount: $3221
Enter withdraw amount: $3298
```

```
Would you like to enter another account? (Y/N): y
Enter your name: Emily E
Enter bank name: SOfi
Enter account number: 2772772
Enter initial balance: $5000
Enter deposit amount: $3221
Enter withdraw amount: $3298

Bank Account Summary:
Account Holder: Emily E
Bank Name: SOfi
Account Number: 2772772
Current Balance: $4923

Would you like to enter another account? (Y/N): y
Enter your name: Josh
Enter bank name: PCB
Enter account number: 4555255
Enter initial balance: $1000
Enter deposit amount: $-1909
Enter withdraw amount: $-2992
Invalid amount entered! Program terminating.

Thank you for using the Bank Account Manager!
```

```
Enter your name: Sumanth
Enter bank name: US bank
Enter account number: 1832200
Enter initial balance: $900
Enter deposit amount: $1000
Enter withdraw amount: $2000
Insufficient funds for this withdrawal! Program terminating.

Thank you for using the Bank Account Manager!
```

Evaluation:

- Ensure correct use of custom exceptions and appropriate handling for each case.

- Test with different inputs to observe stack unwinding, rethrowing, and continuation after handling exceptions.

- Write comments in your code explaining each component.

Good luck with the assignment, and feel free to ask questions if you encounter any issues!