Bank Account Management System - C++

Objective

Develop a C++ program for a bank account management system. This system should offer functionalities for creating, managing, performing transactions, and viewing details of bank accounts.

Class Details

1. Account Class (Base Account)
    - Member Variables
        - string `accountId`: Identifier for the account.
        - double `balance`: Current balance of the account.
        - string `ownerName`: Name of the account owner.
    - Constructor
        - Account(string id, double bal, string owner)
    - Member Functions
        - void `deposit(double amount)`: Deposits money into the account.
        - bool `withdraw(double amount)`: Withdraws money from the account. Fails if the amount exceeds the balance.
        - double `getBalance()` const: Returns the current balance.
        - virtual void `displayAccount()`: Displays details of the account.
2. CheckingAccount Class (Checking Account)
    - Inheritance
        - Inherits from the Account class.
    - Additional Member Variable
        - double `fee`: Fee charged per transaction.
    - Constructor
        - `CheckingAccount(string id, double bal, string owner, double f)`
    - Overridden Member Functions
        - void `deposit(double amount)`: Deposit function with a transaction fee.
        - bool `withdraw(double amount)`: Withdrawal function with a transaction fee.

3. SavingsAccount Class (Savings Account)
    - Inheritance
        - Inherits from the Account class.
    - Additional Member Variable
        - double `interestRate`: Annual interest rate.
    - Constructor
        - `SavingsAccount(string id, double bal, string owner, double rate)`
    - Additional Member Functions
        - void `applyInterest()`: Applies interest to the balance.
4. Bank Class (Bank)
    - Member Variable
        - vector<Account*> `accounts`: List of all accounts managed by the bank.
    - Member Functions
        - void `addAccount(Account* account)`: Adds a new account.
        - void `displayAllAccounts()`: Displays information of all accounts.
    - Destructor
        - Frees memory allocated to Account objects.

Implementation Requirements

1. Account Class
    - Define and implement all necessary member variables and functions.
    - Ensure accurate functioning of deposit, withdrawal, and balance inquiry.
2. CheckingAccount Class
    - Extend Account class functionalities to include transaction fees.
    - Modify deposit and withdrawal functions to include the fee.
3. SavingsAccount Class
    - Extend Account class functionalities to handle interest rates.
    - Implement a function to apply interest to the balance.
4. Bank Class
    - Capable of managing various account types.
    - Implement functionality to display details of all accounts.
    - Destructor should handle freeing of allocated memory.

Testing

- Create and test functionalities in the main function.
- Include testing for deposit, withdrawal, balance check, and interest application for each account type.
- Ensure all account information is correctly displayed.