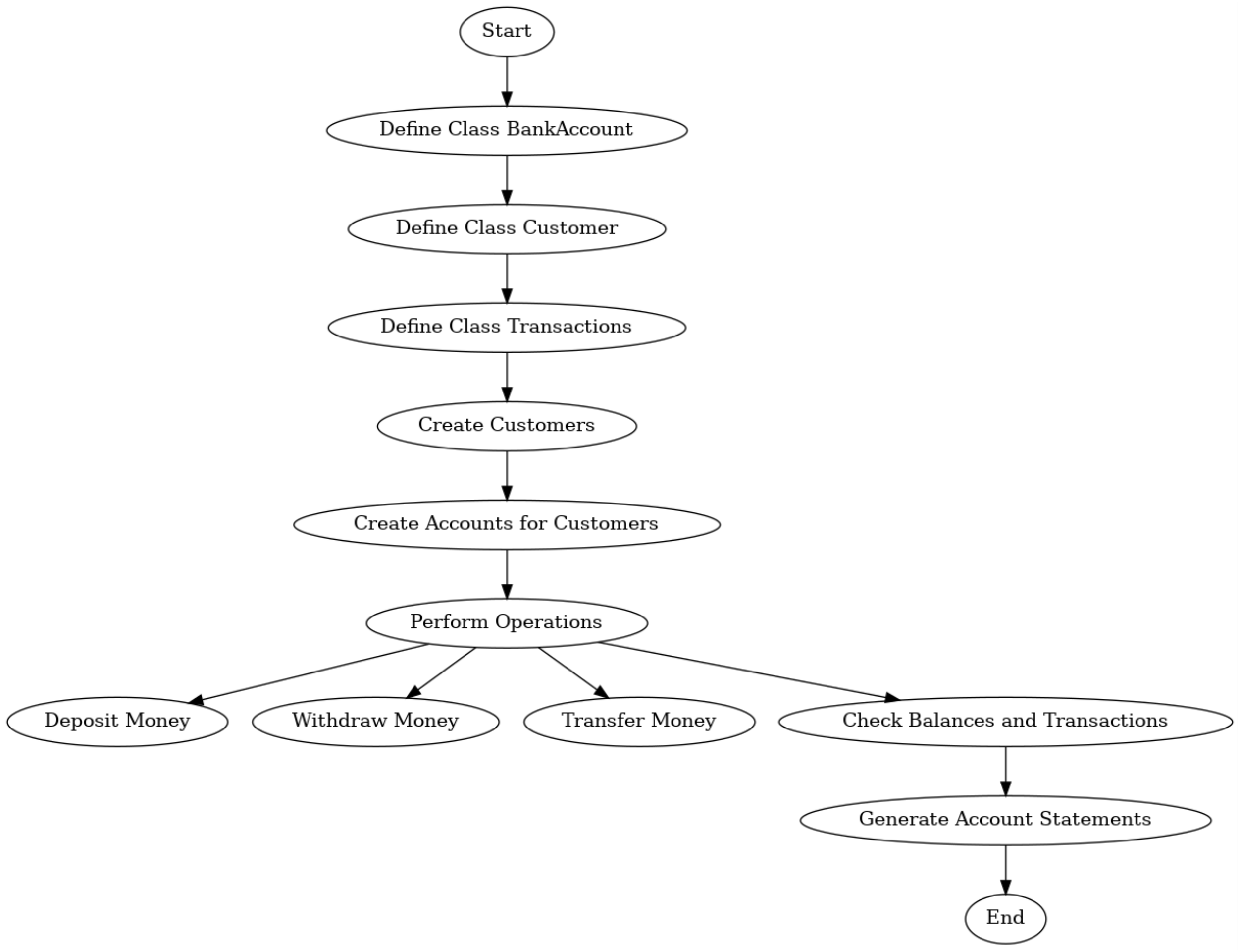
step-by-step explanation of how the program works, following the flowchart:



1. Start

   - The program execution begins.

2. Define Class `BankAccount`

   - The `BankAccount` class is defined with methods for initializing an account, depositing money, withdrawing money, transferring money, and getting the account balance and transactions.

3. Define Class `Customer`

   - The `Customer` class is defined with methods for initializing a customer, creating accounts for the customer, and retrieving the customer's accounts.

4. Define Class `Transactions`

   - The `Transactions` class is defined with methods for adding a transaction and getting the transaction history.

5. Create Customers

   - Instances of the `Customer` class are created. For example, `customer1` is created with the name "John Doe" and customer ID "C001", and `customer2` is created with the name "Jane Smith" and customer ID "C002".

6. Create Accounts for Customers

   - Accounts are created for each customer using the `create_account` method. For example, `customer1` creates a savings account with an initial deposit of 1000 and a checking account with an initial deposit of 500. `customer2` creates a savings account with an initial deposit of 1500.

7. Perform Operations

- Various banking operations are performed. This step is divided into three sub-steps:

  - Deposit Money: Money is deposited into an account. For example, `customer1` deposits 500 into their first account.

  - Withdraw Money: Money is withdrawn from an account. For example, `customer1` withdraws 200 from their second account.

  - Transfer Money: Money is transferred from one account to another. For example, `customer1` transfers 300 from their first account to `customer2`'s first account.

8. Check Balances and Transactions

- The balances and transactions for each account are checked and printed. This involves retrieving the balance and transaction history for each account.

9. Generate Account Statements

- Account statements are generated for each account, summarizing the transactions and the current balance. This is done using the `generate_statement` function.

10. End

- The program execution ends.

Additional Details:

- Class `BankAccount` Methods:

  - `__init__(self, account_number, account_type, balance=0.0)`: Initializes the account with the given account number, type, and balance.

  - `deposit(self, amount)`: Deposits the specified amount into the account.

  - `withdraw(self, amount)`: Withdraws the specified amount from the account, if sufficient funds are available.

  - `transfer(self, amount, target_account)`: Transfers the specified amount to another account, if sufficient funds are available.

  - `get_balance(self)`: Returns the current balance of the account.

  - `get_transactions(self)`: Returns the transaction history of the account.

- Class `Customer` Methods:

  - `__init__(self, name, customer_id)`: Initializes the customer with the given name and customer ID.

  - `create_account(self, account_type, initial_deposit=0.0)`: Creates a new account for the customer with the specified account type and initial deposit.

  - `get_accounts(self)`: Returns the list of accounts for the customer.

- Class `Transactions` Methods:

  - `__init__(self)`: Initializes the transaction history.

  - `add_transaction(self, transaction)`: Adds a transaction to the transaction history.

  - `get_transaction_history(self)`: Returns the transaction history.

- Function `generate_statement(account)`:

  - Generates a statement for the given account, including the account number, type, transaction history, and current balance.