Add icon Add cover

Bank Account Management

Complexity: Medium

Time Estimate: 10 hours

Objectives

By completing this project, you will be able to:

- Apply OOP principles (encapsulation, inheritance, polymorphism) to design Java classes and interfaces for real-world problems
- Create well-structured applications integrating primitive data types, control structures, and custom objects
- Analyze class relationships to choose between inheritance, composition, abstract classes, and interfaces
- Evaluate code quality using proper encapsulation, naming conventions, and OOP best practices
- Apply polymorphic behavior with method overriding to build flexible, extensible code
- Apply fundamental Data Structures and Algorithms concepts to manage, search, and organize account and transaction data efficiently

What You'll Build

A console application with these features:

Core Features

- Create Account - Register new bank accounts for customers
- View Accounts - Display all accounts with their details
- Process Transaction - Deposit or withdraw money from accounts
- View Transactions - Display transaction history for an account
- Simple Menu - Navigate through options

Account Types

- Savings Account - earns interest (interest rate: 3.5% annually, minimum balance: $500)
- Checking Account - no interest, has overdraft limit (overdraft limit: $1000, monthly fee: $10)

Customer Types

- Regular Customer - standard banking services
- Premium Customer - higher transaction limits, waived fees (minimum balance: $10,000, benefits: no monthly fees, priority service)

Console Output Examples

Screenshot 1: Main Menu

Plain Text

```
1 ╔═══════════════════════════════════════════════════════════
```

```
╔═══════════════════════════════════════╗
║   BANK ACCOUNT MANAGEMENT - MAIN MENU      ║
╚═══════════════════════════════════════╝

1. Create Account
2. View Accounts
3. Process Transaction
4. View Transaction History
5. Exit

Enter choice: _
```

Screenshot 2: View Accounts

Plain Text

```
Enter choice: 2

ACCOUNT LISTING
──────────────────────────────────────────────────────────────────────────
ACC NO   | CUSTOMER NAME     | TYPE              | BALANCE      | STATUS
──────────────────────────────────────────────────────────────────────────
ACC001   | John Smith        | Savings           | $5,250.00    | Active
         | Interest Rate: 3.5% | Min Balance: $500.00
──────────────────────────────────────────────────────────────────────────
ACC002   | Sarah Johnson     | Checking          | $3,450.00    | Active
         | Overdraft Limit: $1,000.00 | Monthly Fee: $10.00
──────────────────────────────────────────────────────────────────────────
ACC003   | Michael Chen      | Savings           | $15,750.00   | Active
         | Interest Rate: 3.5% | Min Balance: $500.00
──────────────────────────────────────────────────────────────────────────
ACC004   | Emily Brown       | Checking          | $890.00      | Active
         | Overdraft Limit: $1,000.00 | Monthly Fee: $10.00
──────────────────────────────────────────────────────────────────────────
ACC005   | David Wilson      | Savings           | $25,300.00   | Active
         | Interest Rate: 3.5% | Min Balance: $500.00
──────────────────────────────────────────────────────────────────────────

Total Accounts: 5
Total Bank Balance: $50,640.00

Press Enter to continue...
```

Screenshot 3: Create Account (Savings)

Plain Text

```
Enter choice: 1

ACCOUNT CREATION
──────────────────────────────────────────

Enter customer name: Robert Martinez
Enter customer age: 35
Enter customer contact: +1-555-7890
Enter customer address: 123 Oak Street, Springfield

Customer type:
1. Regular Customer (Standard banking services)
2. Premium Customer (Enhanced benefits, min balance $10,000)

Select type (1-2): 1

Account type:
1. Savings Account (Interest: 3.5%, Min Balance: $500)
2. Checking Account (Overdraft: $1,000, Monthly Fee: $10)
```

```
Select type (1-2): 1

Enter initial deposit amount: $2500

✓ Account created successfully!
  Account Number: ACC006
  Customer: Robert Martinez (Regular)
  Account Type: Savings
  Initial Balance: $2,500.00
  Interest Rate: 3.5%
  Minimum Balance: $500.00
  Status: Active

Press Enter to continue...

```

Screenshot 4: Create Account (Premium Customer + Checking)

Plain Text

```
Enter choice: 1

ACCOUNT CREATION
──────────────────────────────────────────

Enter customer name: Jennifer Lee
Enter customer age: 42
Enter customer contact: +1-555-4321
Enter customer address: 456 Maple Avenue, Springfield

Customer type:
1. Regular Customer (Standard banking services)
2. Premium Customer (Enhanced benefits, min balance $10,000)

Select type (1-2): 2

Account type:
1. Savings Account (Interest: 3.5%, Min Balance: $500)
2. Checking Account (Overdraft: $1,000, Monthly Fee: $10)

Select type (1-2): 2

Enter initial deposit amount: $15000

✓ Account created successfully!
  Account Number: ACC007
  Customer: Jennifer Lee (Premium)
  Account Type: Checking
  Initial Balance: $15,000.00
  Overdraft Limit: $1,000.00
  Monthly Fee: $0.00 (WAIVED - Premium Customer)
  Status: Active

Press Enter to continue...

```

Screenshot 5: Process Transaction (Deposit)

Plain Text

```
Enter choice: 3

PROCESS TRANSACTION
──────────────────────────────────────────

Enter Account Number: ACC001
```

```
Account Details:
Customer: John Smith
Account Type: Savings
Current Balance: $5,250.00

Transaction type:
1. Deposit
2. Withdrawal

Select type (1-2): 1

Enter amount: $1500

TRANSACTION CONFIRMATION
──────────────────────────────────────
Transaction ID: TXN001
Account: ACC001
Type: DEPOSIT
Amount: $1,500.00
Previous Balance: $5,250.00
New Balance: $6,750.00
Date/Time: 30-10-2025 10:30 AM
──────────────────────────────────────

Confirm transaction? (Y/N): Y

✓ Transaction completed successfully!

Press Enter to continue...

```

Screenshot 6: Process Transaction (Withdrawal)

Plain Text

```
Enter choice: 3

PROCESS TRANSACTION
──────────────────────────────────────

Enter Account Number: ACC002

Account Details:
Customer: Sarah Johnson
Account Type: Checking
Current Balance: $3,450.00

Transaction type:
1. Deposit
2. Withdrawal

Select type (1-2): 2

Enter amount: $500

TRANSACTION CONFIRMATION
──────────────────────────────────────
Transaction ID: TXN002
Account: ACC002
Type: WITHDRAWAL
Amount: $500.00
Previous Balance: $3,450.00
New Balance: $2,950.00
Date/Time: 30-10-2025 11:15 AM
──────────────────────────────────────

Confirm transaction? (Y/N): Y

```

```
✓ Transaction completed successfully!

Press Enter to continue...

```

Screenshot 7: View Transaction History (Empty)

Plain Text

```
Enter choice: 4

VIEW TRANSACTION HISTORY
─────────────────────────

Enter Account Number: ACC005

Account: ACC005 - David Wilson
Account Type: Savings
Current Balance: $25,300.00

──────────────────────────────────────────────
No transactions recorded for this account.
──────────────────────────────────────────────

Press Enter to continue...

```

Screenshot 8: View Transaction History (With Records)

Plain Text

```
Enter choice: 4

VIEW TRANSACTION HISTORY
─────────────────────────

Enter Account Number: ACC001

Account: ACC001 - John Smith
Account Type: Savings
Current Balance: $6,750.00

TRANSACTION HISTORY
──────────────────────────────────────────────────────────────────────────
TXN ID  | DATE/TIME           | TYPE       | AMOUNT      | BALANCE
──────────────────────────────────────────────────────────────────────────
TXN001  | 30-10-2025 10:30 AM | DEPOSIT    | +$1,500.00  | $6,750.00
TXN002  | 28-10-2025 02:15 PM | WITHDRAWAL | -$750.00    | $5,250.00
TXN003  | 25-10-2025 09:45 AM | DEPOSIT    | +$2,000.00  | $6,000.00
TXN004  | 20-10-2025 03:30 PM | WITHDRAWAL | -$500.00    | $4,000.00
──────────────────────────────────────────────────────────────────────────

Total Transactions: 4
Total Deposits: $3,500.00
Total Withdrawals: $1,250.00
Net Change: +$2,250.00

Press Enter to continue...

```

Screenshot 9: Exit Application

Plain Text

```
Enter choice: 5

Thank you for using Bank Account Management System!
Goodbye!

```

User Stories

US-1: View Accounts

As a bank staff member

I want to view all bank accounts

So that I can see account details and balances

Acceptance Criteria:

- Display minimum 5 accounts (3 Savings, 2 Checking)
- Show account number, customer name, type, balance, and status
- Savings accounts show interest rate and minimum balance
- Checking accounts show overdraft limit and monthly fee
- Display total accounts and total bank balance

Classes to Create:

Account (abstract class)

- Private fields: `accountNumber` (String), `customer` (Customer), `balance` (double), `status` (String)
- Static field: `accountCounter` (int) - for generating unique account numbers
- Constructor, getters, setters
- Abstract method: `displayAccountDetails()`
- Abstract method: `getAccountType()`
- Methods: `deposit(double amount)`, `withdraw(double amount)`

SavingsAccount extends Account

- Private fields: `interestRate` (double), `minimumBalance` (double)
- Constructor (sets interest rate to 3.5%, minimum balance to $500)
- Override `displayAccountDetails()` to show account info + interest details
- Override `getAccountType()` to return "Savings"
- Override `withdraw()` to check minimum balance
- Method: `calculateInterest()` - returns interest earned

CheckingAccount extends Account

- Private fields: `overdraftLimit` (double), `monthlyFee` (double)
- Constructor (sets overdraft limit to $1000, monthly fee to $10)
- Override `displayAccountDetails()` to show account info + overdraft details
- Override `getAccountType()` to return "Checking"
- Override `withdraw()` to allow overdraft up to limit
- Method: `applyMonthlyFee()` - deducts monthly fee from balance

US-2: Create Account

As a bank staff member

I want to create new bank accounts

So that customers can start banking

Acceptance Criteria:

- Capture customer details (name, age, contact, address)
- Support two customer types: Regular and Premium
- Support two account types: Savings and Checking
- Premium customers have waived monthly fees
- Auto-generate unique account number
- Require initial deposit
- Display confirmation with all details

Classes to Create:

Customer (abstract class)

- Private fields: `customerId` (String), `name` (String), `age` (int), `contact` (String), `address` (String)
- Static field: `customerCounter` (int) - for generating unique IDs
- Constructor, getters, setters
- Abstract method: `displayCustomerDetails()`
- Abstract method: `getCustomerType()`

RegularCustomer extends Customer

- Constructor accepting name, age, contact, address
- Override `displayCustomerDetails()` to show customer info
- Override `getCustomerType()` to return "Regular"

PremiumCustomer extends Customer

- Private field: `minimumBalance` (double) - minimum to maintain premium status ($10,000)
- Constructor accepting name, age, contact, address
- Getter and setter for minimum balance
- Override `displayCustomerDetails()` to show customer info + premium benefits
- Override `getCustomerType()` to return "Premium"
- Method: `hasWaivedFees()` - returns true (premium customers don't pay monthly fees)

US-3: Process Transaction

As a bank staff member

I want to process deposits and withdrawals

So that customers can access their money

Acceptance Criteria:

- User enters account number
- Validate that account exists
- Allow selection of transaction type (deposit/withdrawal)
- For deposits: accept any positive amount

- For withdrawals: check sufficient balance (or overdraft for checking)
- For savings withdrawals: ensure minimum balance is maintained
- Generate unique transaction ID
- Update account balance
- Show confirmation before finalizing

Classes to Create:

Transactable (interface)

- Method: `processTransaction(double amount, String type)` - returns boolean

Transaction

- Static field: `transactionCounter` (int) - for generating unique IDs
- Private fields: `transactionId` (String), `accountNumber` (String), `type` (String), `amount` (double), `balanceAfter` (double), `timestamp` (String)
- Constructor accepting account number, type, amount, balance after transaction
- Auto-generates transaction ID (TXN001, TXN002, etc.)
- Auto-generates timestamp
- Getters for all fields
- Method: `displayTransactionDetails()`

AccountManager (uses composition)

- Private field: `accounts` (Account array, size 50)
- Private field: `accountCount` (int) - tracks number of accounts
- Methods:
  - `addAccount(Account)` - adds account to array
  - `findAccount(String accountNumber)` - returns Account or null
  - `viewAllAccounts()` - displays all accounts
  - `getTotalBalance()` - sums all account balances
  - `getAccountCount()` - returns number of accounts

US-4: View Transaction History

As a bank staff member

I want to view transaction history for an account

So that I can track account activity

Acceptance Criteria:

- Display all transactions for a specific account
- Show transaction ID, date/time, type, amount, and balance after transaction
- Display summary: total deposits, total withdrawals, net change
- Handle accounts with no transactions
- Transactions displayed in reverse chronological order (newest first)

Classes to Create:

TransactionManager (uses composition)

- Private field: `transactions` (Transaction array, size 200)

- Private field: `transactionCount` (int) - tracks number of transactions
- Methods:
  - `addTransaction(Transaction)` - adds transaction to array
  - `viewTransactionsByAccount(String accountNumber)` - displays transactions for account
  - `calculateTotalDeposits(String accountNumber)` - sums deposits
  - `calculateTotalWithdrawals(String accountNumber)` - sums withdrawals
  - `getTransactionCount()` - returns total transaction count

US-5: Simple Menu Navigation

As a user

I want to navigate through menu options

So that I can use all features

Acceptance Criteria:

- Display clear menu with 5 options
- Accept and validate user input
- Execute selected option
- Loop until user exits
- Handle invalid input gracefully

Minimum Requirements

- All 11 required classes implemented
- All 5 user stories working
- Static counters work correctly for ID generation
- Use appropriate **data structures** to manage and retrieve accounts and transactions efficiently (e.g., arrays or lists).
- Application runs without errors
- Input validation implemented
- Transaction history tracks all operations
- Minimum balance enforced for savings accounts
- Overdraft limit enforced for checking accounts
- README.md included
- GitHub repository is public
- Repository link submitted

Classes (11 total)

- Account (abstract), SavingsAccount, CheckingAccount
- Customer (abstract), RegularCustomer, PremiumCustomer
- Transactable (interface)
- Transaction, AccountManager, TransactionManager
- Main

Features

- Display 5 accounts (3 Savings, 2 Checking)
- Create accounts (both types) with customer linking
- Process transactions (deposits and withdrawals)
- View transaction history
- Menu navigation

OOP

- Private fields with public getters/setters
- Inheritance (Account & Customer hierarchies)
- Abstract classes and methods
- Interface implementation (Transactable)
- Method overriding (displayAccountDetails, getAccountType, etc.)
- Composition (AccountManager has Account array, TransactionManager has Transaction array)
- Static members (Account counter, Customer counter, Transaction counter)

Data Structures & Algorithms (DSA)

- Apply fundamental Data Structures and Algorithms concepts to manage, search, and organize account and transaction data efficiently
- Apply **basic search algorithms** (e.g., linear search) to locate accounts and transactions by ID.
- Implement **simple sorting logic** (optional) for displaying transactions or accounts in a defined order (e.g., newest first).
- Demonstrate understanding of **time complexity** when selecting or designing structures for managing data.

Code Quality

- Proper naming conventions (camelCase, PascalCase)
- Application runs without errors
- Input validation
- README.md with setup instructions

Grading Rubric

	Criteria	Points	What We're Looking For
1	OOP Principles	25	Encapsulation (private fields), inheritance (2 hierarchies), polymorphism (method overriding), abstraction (abstract classes + interface), composition (Manager classes)
2	Functionality	25	All 5 user stories work: view accounts, create accounts, process transactions, view history, menu navigation
3	Class Design	15	All 11 required classes created, proper relationships, appropriate use of abstract classes and interfaces, correct use of static fields for ID generation
4	DSA	15	Proper use of data structures for account and transaction management, correct application of search and iteration algorithms, code efficiency and clarity
5	Code Quality	10	Clean code, proper naming, good formatting, input validation, no errors

6	Documentation	10	README with setup instructions, code comments for complex logic, clear user prompts
7	**Total**	**100**	

Testing the Application

Test Scenario 1: View Accounts

1. Run application
2. Select option 2 (View Accounts)
3. Verify 5 accounts display with correct information
4. Check Savings accounts show 3.5% interest rate and $500 minimum balance
5. Check Checking accounts show $1000 overdraft and $10 monthly fee
6. Verify total accounts and total balance calculations

Test Scenario 2: Create Savings Account (Regular Customer)

1. Select option 1 (Create Account)
2. Enter customer details (name, age, contact, address)
3. Select Regular Customer type
4. Select Savings Account type
5. Enter initial deposit ($1000)
6. Verify unique account number is generated (ACC001, ACC002, etc.)
7. Verify confirmation displays all details

Test Scenario 3: Create Checking Account (Premium Customer)

1. Select option 1 (Create Account)
2. Enter customer details
3. Select Premium Customer type
4. Select Checking Account type
5. Enter initial deposit ($15,000)
6. Verify monthly fee is waived in confirmation

Test Scenario 4: Process Deposit

1. Create at least one account first
2. Select option 3 (Process Transaction)
3. Enter valid account number
4. Select Deposit
5. Enter amount ($500)
6. Verify transaction ID is generated
7. Verify balance updates correctly
8. Confirm transaction

Test Scenario 5: Process Withdrawal (Savings - Minimum Balance Check)

1. Select option 3 (Process Transaction)
2. Enter savings account number
3. Select Withdrawal
4. Try to withdraw amount that would go below minimum balance
5. Verify system prevents withdrawal
6. Try valid withdrawal amount
7. Verify transaction succeeds

Test Scenario 6: Process Withdrawal (Checking - Overdraft)

1. Select option 3 (Process Transaction)
2. Enter checking account number with balance $500
3. Select Withdrawal
4. Enter $1200 (within overdraft limit of $1000)
5. Verify transaction succeeds with negative balance
6. Try withdrawal exceeding overdraft limit
7. Verify system prevents withdrawal

Test Scenario 7: View Transaction History (Empty)

1. Create new account
2. Select option 4 (View Transaction History)
3. Enter new account number
4. Verify "No transactions" message displays

Test Scenario 8: View Transaction History (With Records)

1. Process 3-4 transactions on one account
2. Select option 4 (View Transaction History)
3. Enter account number
4. Verify all transactions display correctly
5. Verify summary calculations (total deposits, withdrawals, net change)

Test Scenario 9: ID Auto-generation

1. Create 3 accounts
2. Verify account numbers are ACC001, ACC002, ACC003
3. Process 3 transactions
4. Verify transaction IDs are TXN001, TXN002, TXN003
5. Verify customer IDs are CUS001, CUS002, CUS003