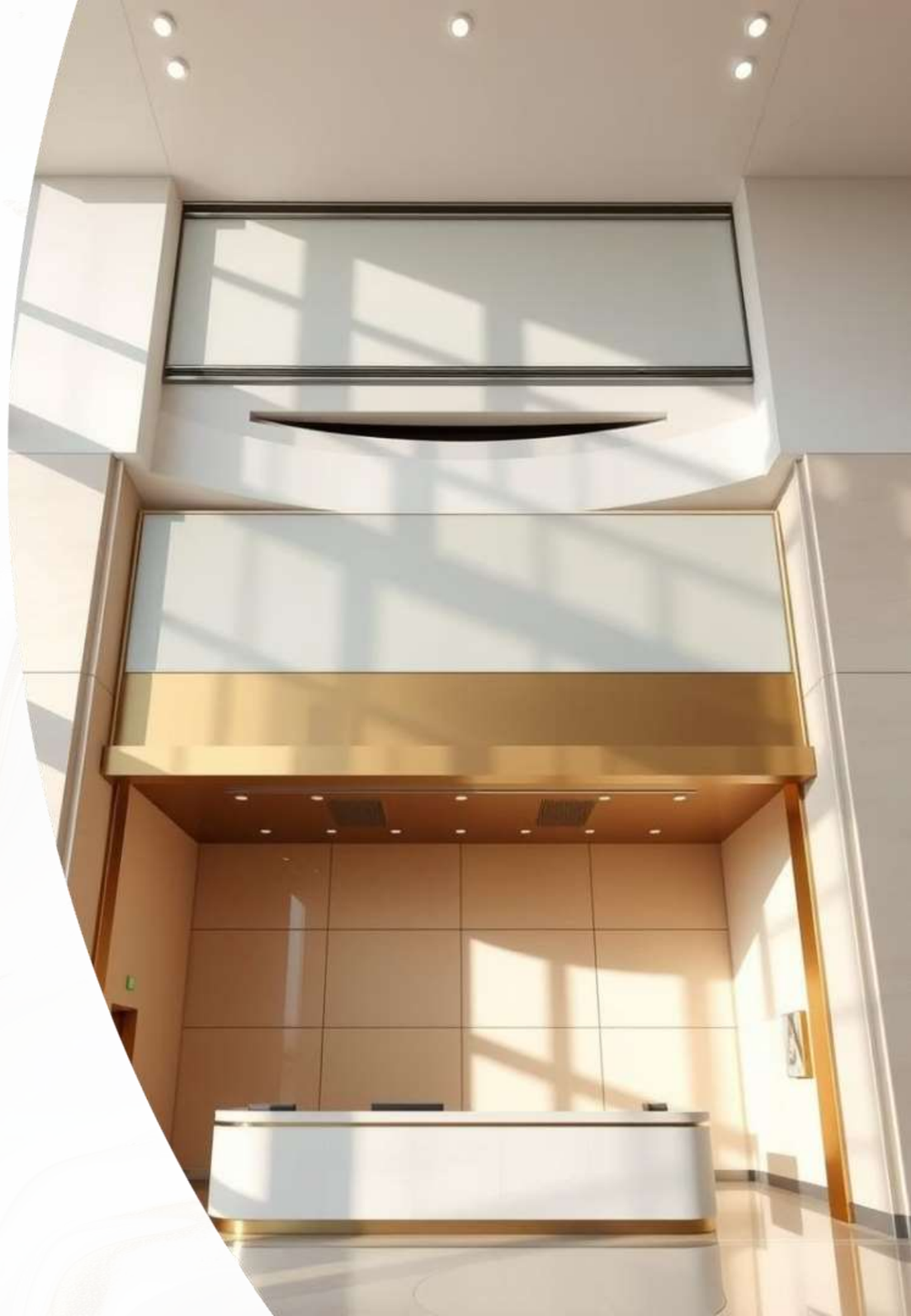
Multi-User Banking System Using Inheritance in Java

Review submission presenting a scalable multi-user banking system built with Java OOP inheritance.



Project Introduction

Purpose

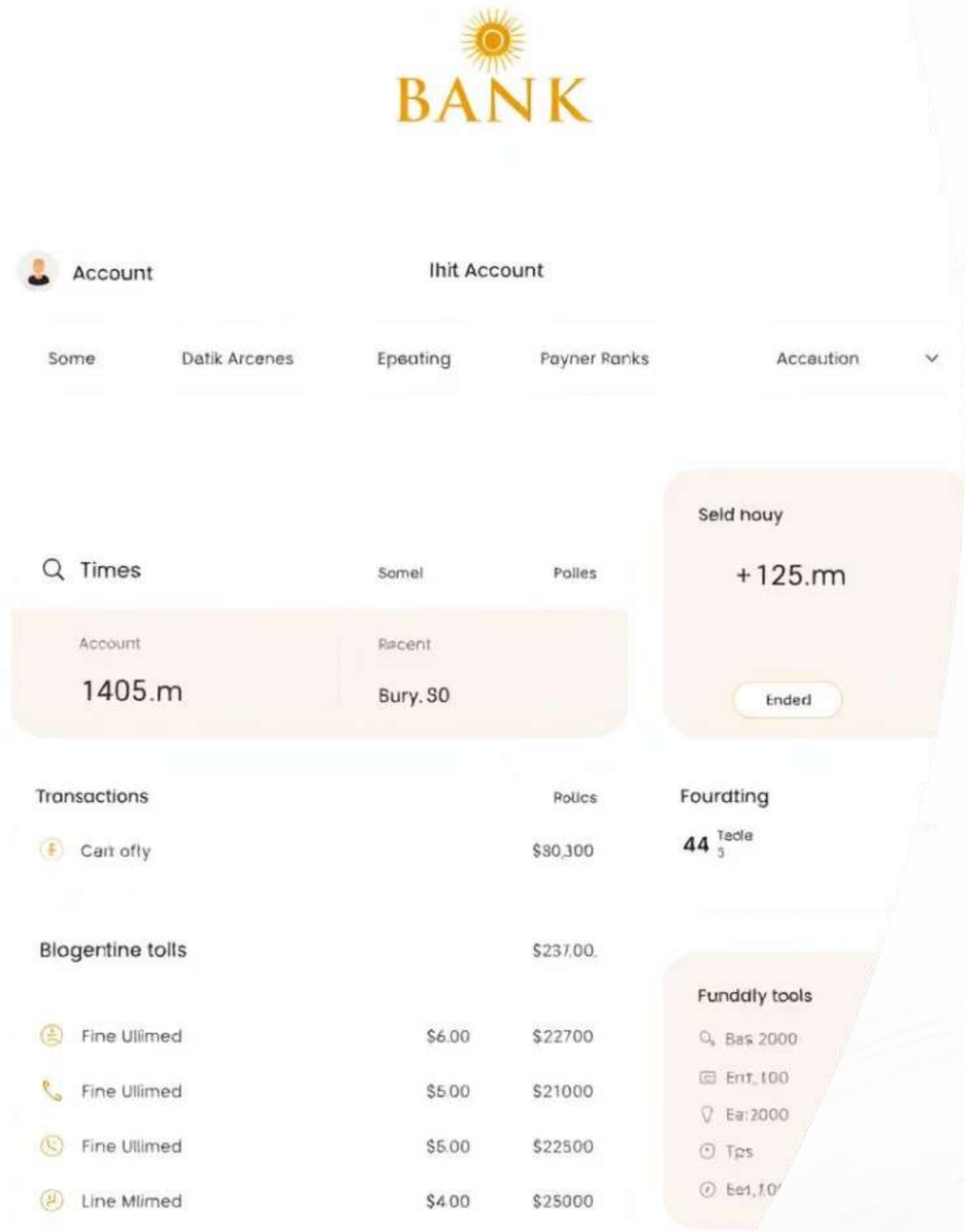
Build scalable multi-user banking system

Functions

Manage accounts, perform transactions

Design focus

Use inheritance for modular architecture



Key System Features

User Management

- User registration
- Login authentication

Account Handling

- Create/manage multiple accounts
- View balances

Transactions

- Deposits
- Withdrawals

Admin Role

- View all accounts
- Monitor user activities

Technology Stack & OOP Concepts

Technology

- Java JDK 17
- IntelliJ IDEA / Eclipse IDE

OOP Principles

- Inheritance for roles
- Encapsulation for security
- Polymorphism for flexible ops
- Abstraction to reduce complexity

System Architecture & Class Design

Base and Derived Classes

- User: name, userId, login()
- AccountHolder: deposit(), withdraw(), viewBalance()
- Admin: viewAllAccounts(), manageUsers()

Supporting Classes

- BankAccount: accountNumber, balance, transactions
- Transaction: amount, type, timestamp

```
    }

    public String getUserId() { return userId; }
    public String getName() { return name; }
    public double getBalance() { return balance; }
}

class SavingsAccount extends Account {
    public SavingsAccount(String userId, String name, dou
        super(userId, name, balance);
```

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Acer\OneDrive\Desktop\java> cd "c:\Users\Acer\AppData\L
gSystem.java } ; if ($?) { java BankingSystem }
No data found. Starting fresh.

===== Multi-User Banking System =====
1. Create Account
2. Deposit
3. Withdraw
4. View Account
5. Delete Account
```

Welcome | BankingSystem.java 9

```
public class BankingSystem {
    public static void main(String[] args) {
                if (acc != null) {
                    System.out.print("Amount to withdraw: ");
                    double amt = sc.nextDouble();
                    if (acc.withdraw(amt)) {
                        dao.updateAccount(acc);
                        System.out.println("✅ Withdrawal successful.");
                    } else System.out.println("❌ Withdrawal failed. Check balance or limit.");
                } else System.out.println("❌ Account not found.");
            }
            case 4 -> {
                sc.nextLine();
                System.out.print("User ID: ");
                String id = sc.nextLine();
                Account acc = dao.getAccount(id);
                if (acc != null) {
                    System.out.println("\n--- Account Details ---");
                    System.out.println("User ID       : " + acc.getUserId());
                    System.out.println("Name          : " + acc.getName());
                    System.out.println("Account Type  : " + acc.getType());
                    System.out.println("Balance       : ₹" + acc.getBalance());
                    System.out.println("Interest Rate : " + acc.getInterestRate() + "%");
                    System.out.println("Withdraw Limit: ₹" + acc.getWithdrawalLimit());
                } else System.out.println("❌ Account not found.");
            }
```

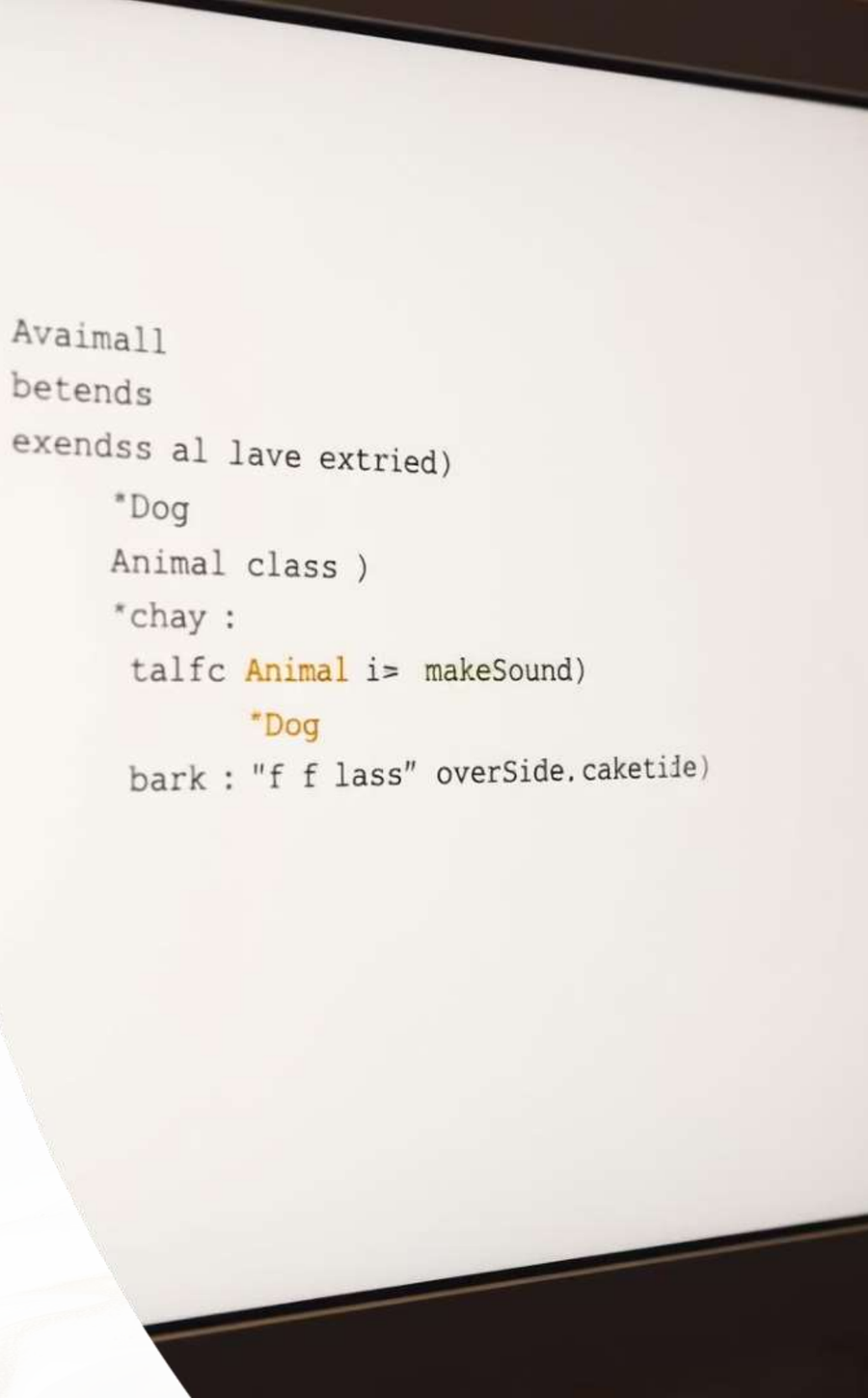
Welcome | BankingSystem.java 9

```
public class BankingSystem {
    public static void main(String[] args) {
                if (acc != null) {
                    System.out.print("Amount to withdraw: ");
                    double amt = sc.nextDouble();
                    if (acc.withdraw(amt)) {
                        dao.updateAccount(acc);
                        System.out.println("✅ Withdrawal successful.");
                    } else System.out.println("❌ Withdrawal failed. Check balance or limit.");
                } else System.out.println("❌ Account not found.");
            }
            case 4 -> {
                sc.nextLine();
                System.out.print("User ID: ");
                String id = sc.nextLine();
                Account acc = dao.getAccount(id);
                if (acc != null) {
                    System.out.println("\n--- Account Details ---");
                    System.out.println("User ID       : " + acc.getUserId());
                    System.out.println("Name          : " + acc.getName());
                    System.out.println("Account Type  : " + acc.getType());
                    System.out.println("Balance       : ₹" + acc.getBalance());
                    System.out.println("Interest Rate : " + acc.getInterestRate() + "%");
                    System.out.println("Withdraw Limit: ₹" + acc.getWithdrawalLimit());
                } else System.out.println("❌ Account not found.");
            }
            case 5 -> {
                sc.nextLine();
                System.out.print("User ID: ");
                String id = sc.nextLine();
                if (dao.deleteAccount(id))
                    System.out.println("✅ Account deleted.");
```

Java: Ready | Ln 192, Col 28 | Spaces: 4 | UTF-8 | CRLF | Java | Go Live | 110%

Code Snippet Example

```
class User {
  String name;
  String userId;
  void login() { /* login logic */ }
}
class AccountHolder extends User {
  void deposit(double amount) { /* deposit logic */ }
  void withdraw(double amount) { /* withdraw logic */ }
}
class Admin extends User {
  void viewAllAccounts() { /* admin logic */ }
}
```



Expected Outcomes

Functionality

User-friendly banking app

Code Quality

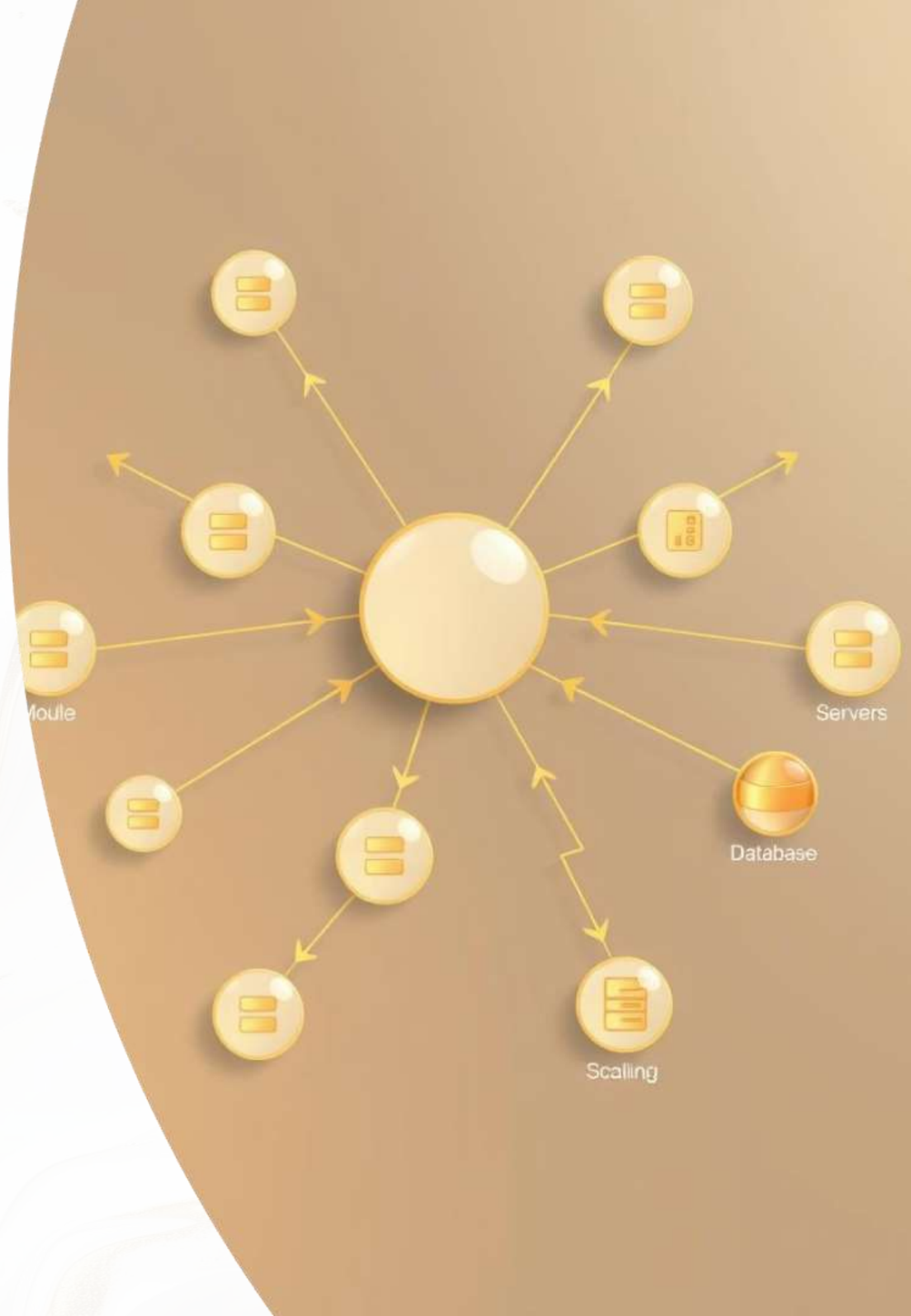
Modular and maintainable Java code

Scalability

Easy to add new features

OOP Demo

Effective inheritance use



Conclusion & Next Steps

1

OOP Strength

Inheritance enables clean role separation

2

Design

Efficient, maintainable, adaptable system

3

Current

Project structure and sample code base

4

Future

Incorporate feedback and expand features