## Banking System Code

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Account {

protected:

string accountNumber;

double balance;

public:

//initialie account

Account(string accNum, double bal) : accountNumber(accNum), balance(bal) {}

//deposit function

virtual void deposit(double amount)

{

balance += amount;

cout << "Deposited $" << amount << " into account " << accountNumber << ". New balance: $" << balance << endl;

}

//withdrawle function

virtual void withdraw(double amount)

{

if (amount > balance)

{

cout << "Insufficient funds in account " << accountNumber << "!" << endl;

}

else

{

balance -= amount;

cout << "Withdrawn $" << amount << " from account " << accountNumber << ". Remaining balance: $" << balance << endl;

}

}

//display account

virtual void display()

{

cout << "Account Number: " << accountNumber << " | Balance: $" << balance << endl;

}

};

class CheckingAccount : public Account

{

public:

//initialize checkin account

CheckingAccount(string accNum, double bal) : Account(accNum, bal) {}

//display account

void display() override

{

cout << "Checking Account: " << accountNumber << " | Balance: $" << balance << endl;

}

};

class SavingsAccount : public Account

{

public:

//initialize savings account

SavingsAccount(string accNum, double bal) : Account(accNum, bal) {}

//display account

void display() override

{

cout << "Savings Account: " << accountNumber << " | Balance: $" << balance << endl;

}

};

class User

{

private:

string username;

string password;

vector<Account\*> accounts;

public:

//initialize user

User(string user, string pass) : username(user), password(pass) {}

//veryify password

bool authenticate()

{

string inputPass;

cout << "Enter password for " << username << ": ";

cin >> inputPass;

//ask till they enter the correct password

while (inputPass != password)

{

cout << "Incorrect password. Try again: ";

cin >> inputPass;

}

cout << "Login successful!\n";

return true;

}

//add account

void addAccount(Account\* acc)

{

accounts.push\_back(acc);

}

//display account

void showAccounts()

{

for (size\_t i = 0; i < accounts.size(); i++)

{

cout << i + 1 << ". ";

accounts[i]->display();

}

}

//verify selected account exists

Account\* selectAccount(int index)

{

if (index >= 1 && index <= accounts.size())

{

return accounts[index - 1];

}

return nullptr;

}

};

int main()

{

cout << "Welcome to the Banking System!\n";

// create user

string userName, userPass;

cout << "Create a username: ";

cin >> userName;

cout << "Create a password: ";

cin >> userPass;

User user(userName, userPass);

//login

user.authenticate();

// test accounts

Account\* savings = new SavingsAccount("Savings1", 5000);

Account\* checking = new CheckingAccount("Checkings1", 2500);

user.addAccount(savings);

user.addAccount(checking);

cout << "Your Accounts:\n";

user.showAccounts();

int choice;

cout << "Select an account (1 or 2): ";

cin >> choice;

Account\* selected = user.selectAccount(choice);

//let user select what action they want to do

if (selected)

{

int action;

double amount;

do

{

cout << "\n1. Deposit\n2. Withdraw\n3. Show Balance\n4. Exit\n";

cout << "Choose an option: ";

cin >> action;

switch (action)

{

case 1:

cout << "Enter amount to deposit: ";

cin >> amount;

selected->deposit(amount);

break;

case 2:

cout << "Enter amount to withdraw: ";

cin >> amount;

selected->withdraw(amount);

break;

case 3:

selected->display();

break;

case 4:

cout << "Exiting...\n";

break;

default:

cout << "Invalid option! Try again.\n";

}

} while (action != 4);

}

else

{

cout << "Invalid account selection!\n";

}

// Free memory

delete savings;

delete checking;

return 0;

}

## Screenshots of output

Testing the savings account

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Testing the checking account

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## Testing

* Create Account: Successfully created a username and password an account
* Password Validation: Successfully validates the password
* Deposit: Successfully lets the user deposit funds
* Withdrawal: Successfully lets the user withdrawal funds
* Withdrawal more than balance: Successfully tells the user they have insufficient funds