main.cpp

// Name: Keshav Mehta, Kenry Yu

// Date: Februrary 8, 2022

#include "Account.h"

using namespace std;

int main()

{

Account my\_account(100); // Set up my account with $100

my\_account.deposit(50);

my\_account.withdraw(175); // Penalty of $20 will apply

my\_account.withdraw(25);

cout << "Account balance: " << my\_account.get\_balance() << "\n";

my\_account.withdraw(my\_account.get\_balance()); // withdraw all

cout << "Account balance: " << my\_account.get\_balance() << "\n";

return 0;

}

Account.h

#ifndef ACCOUNT\_H

#define ACCOUNT\_H

#include <iostream>

using namespace std;

class Account

{

// Private members

private:

  int balance;

  void penalty();

// Public members

public:

// Default and overload contructor

  Account();

  Account(int amount);

// Deposit function that add amount to balance

  void deposit(int amount);

// Withdraw function that take money out from balance

  void withdraw(int amount);

// get\_balance function that return balance

  int get\_balance();

// display function that output statement with balance

void display();

};

#endif

Account.cpp

#include "Account.h"

#include <iostream>

using namespace std;

// Default constructor: sets up an account without an initial balance

Account::Account() {

this->balance = 0;

cout << "Set up my account with $" << get\_balance() << endl;

}

// Overload constructor: sets up an account with a specific initial balance

Account::Account(int amount) {

this->balance = amount;

cout << "Set up my account with $" << get\_balance() << endl;

}

// Deposit function: deposits a valid non-negative ammount

void Account::deposit(int amount) {

if (amount < 0)

cout << "Input error, please try again.\n";

else {

cout << "$" << amount << " deposited into account.\n";

this->balance += amount;

}

}

// Withdraw function: withdraw a valid positive ammount

void Account::withdraw(int amount) {

if (amount < 0)

cout << "Input error, please try again.\n";

else {

this->balance -= amount;

cout << "$" << amount << " withdraw from account.\n";

if (balance < 0) {

penalty();

}

}

}

// GetBalance function: returns the current balance of the account

int Account::get\_balance() { return this->balance; }

// Penalty function: penalise if the balance is lower than the withdraw ammount

void Account::penalty() {

this->balance -= 20;

cout << "Withdrawal amount over available balance, penalty of $20 applied."

<< endl;

}

// Display function: displays the account status

void Account::display() {

cout << "Current available balance in account is $" << get\_balance() << endl;

}

Output

A screenshot of a computer

Description automatically generated with medium confidence