

CPP Problem Design

Subject: ATM

Contributor: 謝宜杭, 謝公耀, 廖宣瑋

Main testing concept: Exception Handling

Basics

- ☐ C++ BASICS
- ☐ FLOW OF CONTROL
- ☐ FUNCTION BASICS
- ☐ PARAMETERS AND OVERLOADING
- ☐ ARRAYS
- ☐ STRUCTURES AND CLASSES
- ☐ CONSTRUCTORS AND OTHER TOOLS
- ☐ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES
- ☐ STRINGS
- ☐ POINTERS AND DYNAMIC ARRAYS

Functions

- ☐ SEPARATE COMPILATION AND NAMESPACES
- ☐ STREAMS AND FILE I/O
- ☐ RECURSION
- ☐ INHERITANCE
- ☐ POLYMORPHISM AND VIRTUAL FUNCTIONS
- ☐ TEMPLATES
- ☐ LINKED DATA STRUCTURES
- ☒ EXCEPTION HANDLING
- ☐ STANDARD TEMPLATE LIBRARY
- ☐ PATTERNS AND UML

Description:

A function that returns a special error code is usually accomplished by throwing an exception. The following class maintains an account balance.

```
class Account
{
private:
    double balance;
public:
    Account() { balance = 0; }
    Account(double initialDeposit) { balance = initialDeposit; }
    double getBalance() { return balance; }

    //returns new balance or -1 if error
    double deposit(double amount)
    {
        if (amount > 0) balance += amount;
        else return -1;
        return balance;
    }

    //return new balance or -1 if invalid amount
    double withdraw(double amount)
    {
        if ((amount > balance) || (amount < 0)) return -1;
        else balance -= amount;
        return balance;
    }
};
```

Rewrite the class so that it throws appropriate exceptions instead of returning -1 as an error code. You should implement following two classes to handle exceptions:

NegativeDeposit: Handling errors generated by the function deposit(double).

InsufficientFunds: Handling errors generated by the function withdraw(double).

Input:

No inputs.

**The main() function in your submission will be replaced when judging.

**You can use the main() function in “Other Notes” to test your program.

Output:

The result of executing your program with the given main function.

Sample Input / Output :

Sample Input	Sample Output
No inputs.	Depositing 50 New balance: 150 Withdraw 25 New balance: 125 Withdraw 250 Not enough money to withdraw that amount. Enter a character to exit

■ Easy, only basic programming syntax and structure are required.

□ Medium, multiple programming grammars and structures are required.

□ Hard, need to use multiple program structures or more complex data types.

Expected solving time:

20 minutes

Other notes:

```
#include<iostream>
#include "Account.h"
using namespace std;
int main()
{
    Account a(100);
    try
    {
        cout << "Depositing 50" << endl;
        cout << "New balance: " << a.deposit(50) << endl;
        cout << "Withdraw 25" << endl;
        cout << "New balance: " << a.withdraw(25) << endl;
        cout << "Withdraw 250" << endl;
        cout << "New balance: " << a.withdraw(250) << endl;
    }
    catch (InsufficientFunds)
    {
        cout << "Not enough money to withdraw that amount." << endl;
    }
    catch (NegativeDeposit)
    {
        cout << "You may only deposit a positive amount." << endl;
    }
    cout << "Enter a character to exit" << endl;
    char wait;
    cin >> wait;
    return 0;
}
```