```cpp
#include "Account.h"
#include "SavingsAcc.h"
#include "CheckingAcc.h"
#include <iostream>
#include <vector>
#include <typeinfo>
using namespace std;

int main() {
    vector<Account*> accounts;
    double withdraw = 0.0;
    double deposit = 0.0;

    // Create SavingsAccount and CheckingAccount objects
    accounts.push_back(new SavingsAccount(500.0, 0.05)); // Savings account with $500 initial balance and 5% interest rate
    accounts.push_back(new CheckingAccount(200.0, 1.0)); // Checking account with $200 initial balance and $1 fee per transaction

    // Process each account

    for (Account *account : accounts){  // pointer "account" 遍及 accounts(可知每格為何種object)

        SavingsAccount *savings = dynamic_cast <SavingsAccount*> (account);
        CheckingAccount *checking = dynamic_cast <CheckingAccount*> (account);

        if(savings != 0){
            cout << "Processing SavingsAccount account." << endl;

            cout << "Enter amount to deposit: ";
            cin >> deposit;
            account -> credit(deposit);

            cout << "Enter amount to withdraw: ";
            cin >> withdraw;
            account -> debit(withdraw);

            double interest = savings -> calculateInterest();
            cout << "Interest added: " << interest << endl;
            cout << "Updated balance: " << savings -> getBalance() + interest << endl << endl;
        }
```

```cpp
        else if (checking != 0){
            cout << "Processing CheckingAccount account." << endl;

            cout << "Enter amount to deposit: ";
            cin >> deposit;
            account -> credit(deposit);

            cout << "Enter amount to withdraw: ";
            cin >> withdraw;
            account -> debit(withdraw);

            cout << "Updated balance: " << checking -> getBalance()<< endl << endl;
        }

    }

    // Clean up dynamically allocated memory

    for (Account* account : accounts){
        delete account;
    }

    return 0;
}
```

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

class Account{
    public:
        Account();
        Account(double);

        virtual void credit(double);
        virtual bool debit(double);

        void setBalance(double);
        double getBalance() const;

    private:
        double balance;
};

```

```
#include "Account.h"
#include <iostream>
using namespace std;

Account::Account(){}

Account::Account(double ba) : balance(0.0){
    setBalance(ba);
}

void Account::credit(double cr){
    balance += cr;
}

bool Account::debit(double deb){
    if(deb > balance){
        cout << "Debit amount exceeded account balance" << endl;
        return false;
    }
    else{
        balance -= deb;
        return true;
    }
}

void Account::setBalance(double ba){
    balance = ba;
}

double Account::getBalance() const{
    return balance;
}

```

```h
#ifndef SAVINGSACCOUNT_H
#define SAVINGSACCOUNT_H

#include "Account.h"

class SavingsAccount : public Account{
    public:
        SavingsAccount();
        SavingsAccount(double, double);

        double calculateInterest() const;

        void setInterestRate(double);
        double getInterestRate() const;

    private:
        double interest_rate;


};

#endif
```

```cpp
#include "SavingsAcc.h"
#include <iostream>
using namespace std;

SavingsAccount::SavingsAccount(){}

SavingsAccount::SavingsAccount(double ba, double rate) : interest_rate(0.0){
  Account::setBalance(ba);
  setInterestRate(rate);
}

double SavingsAccount::calculateInterest() const{
  return (Account::getBalance() * interest_rate);
}

void SavingsAccount::setInterestRate(double rate){
  if (rate < 0.0)
    {
      cout << "The initial interest rate is invalid." << endl;
    }
  else
    {
      interest_rate = rate;
    }
}

double SavingsAccount::getInterestRate() const{
  return interest_rate;
}

```

```
#ifndef CHECKINGACCOUNT_H
#define CHECKINGACCOUNT_H

#include "Account.h"

class CheckingAccount : public Account{
    public:
        CheckingAccount();
        CheckingAccount(double, double);

        virtual void credit(double);
        virtual bool debit(double);

        void setTransactionFee(double);
        double getTransactionFee();

    private:
        double transaction_fee;

};

#endif
```

```cpp
#include "CheckingAcc.h"
#include <iostream>
using namespace std;

CheckingAccount::CheckingAccount(){}

CheckingAccount::CheckingAccount(double ba, double fee) : transaction_fee(0.0){
    Account::setBalance(ba);
    setTransactionFee(fee);
}

void CheckingAccount::credit(double cr){
    Account::credit(cr - transaction_fee);
}

bool CheckingAccount::debit(double deb){
    if (Account::debit(deb + transaction_fee)){
        return true;
    }
    else{
        cout << "Debit amount exceeded account balance" << endl;
        return false;
    }
}

void CheckingAccount::setTransactionFee(double fee){
    if (fee < 0.0){
        cout << "The initial transaction fee is invalid." << endl;
    }
    else{
        transaction_fee = fee;
    }
}

double CheckingAccount::getTransactionFee(){
    return transaction_fee;
}
```

```
Processing SavingsAccount account.
Enter amount to deposit: 100
Enter amount to withdraw: 50
Interest added: 27.5
Updated balance: 577.5

Processing CheckingAccount account.
Enter amount to deposit: 100
Enter amount to withdraw: 50
Updated balance: 248
```