# 实验报告

### Bank

#include<iostream>

#include<iomanip>

using namespace std;

class Account {

public:

virtual void credit(int money) {

}

virtual void debit(int money) {

}

virtual double getBalance() {

return this->balance;

}

private:

double balance = 0.0;

};

class SavingsAccount :public Account {

public:

double getBalance() {

return this->balance;

}

void credit(int money) {

this->balance += money;

//this->balance += this->interestRate \* this->balance;

//cout << this->balance << endl;

}

void debit(int money) {

if (this->balance < money)

{

cout << "Debit amount exceeded account balance" << endl;

}

else

{

this->balance -= money;

//this->balance += this->balance \* this->interestRate;

//cout << "取出后" << this->balance << endl;

}

}

SavingsAccount(double banlance, double intersetRate) {

//cout << banlance << endl;

this->interestRate = intersetRate / 100;

//cout << this->interestRate << endl;

this->balance = banlance;

//cout << this->balance << endl;

}

double calculateInterest() {

this->balance += this->balance \* this->interestRate;

return this->balance \* this->interestRate;

}

private:

double interestRate = 0.0;

double balance = 0.0;

};

class CheckingAccount :public Account {

public:

CheckingAccount(int balance, int service) {

this->balance = balance;

this->service = service;

}

void credit(int money) {

if ((this->balance) - this->service >= 0)

{

this->balance -= this->service;

this->balance += money;

}

else {

cout << "Transaction fee exceeded account balance while crediting" << endl;

}

}

void debit(int money) {

if ((this->balance) - money >= 0)

{

if (this->balance-money-this->service>=0)

{

this->balance -= this->service;

this->balance -= money;

}

else

{

cout << "Transaction fee exceeded account balance while debiting" << endl;

}

//cout << "leave money: " << this->balance << endl;

}

else {

cout << "Debit amount exceeded account balance" << endl;

}

}

double getBalance() {

return this->balance;

}

private:

double balance = 0.0;

double service = 0.0;

};

int main() {

Account\* accounts[3];

accounts[0] = new SavingsAccount(100, 3); //余额100元，利息3%

accounts[1] = new CheckingAccount(100, 5); //余额100元，交易费5元

accounts[2] = new CheckingAccount(50, 5); //余额50元，交易费5元

for (int i = 0; i < 3; i++) {

cout << "第" << i + 1 << "次循环的结果：" << endl;

accounts[i]->debit(200); //借款200元

accounts[i]->debit(40);

accounts[i]->credit(50); //存款50元

accounts[i]->debit(49);

accounts[i]->debit(43);

accounts[i]->credit(1);

//将Account指针强制转换为SavingAccount指针

SavingsAccount\* derivedPtr =

dynamic\_cast<SavingsAccount\*>(accounts[i]);

if (derivedPtr != NULL) //如果类型兼容，转换成功

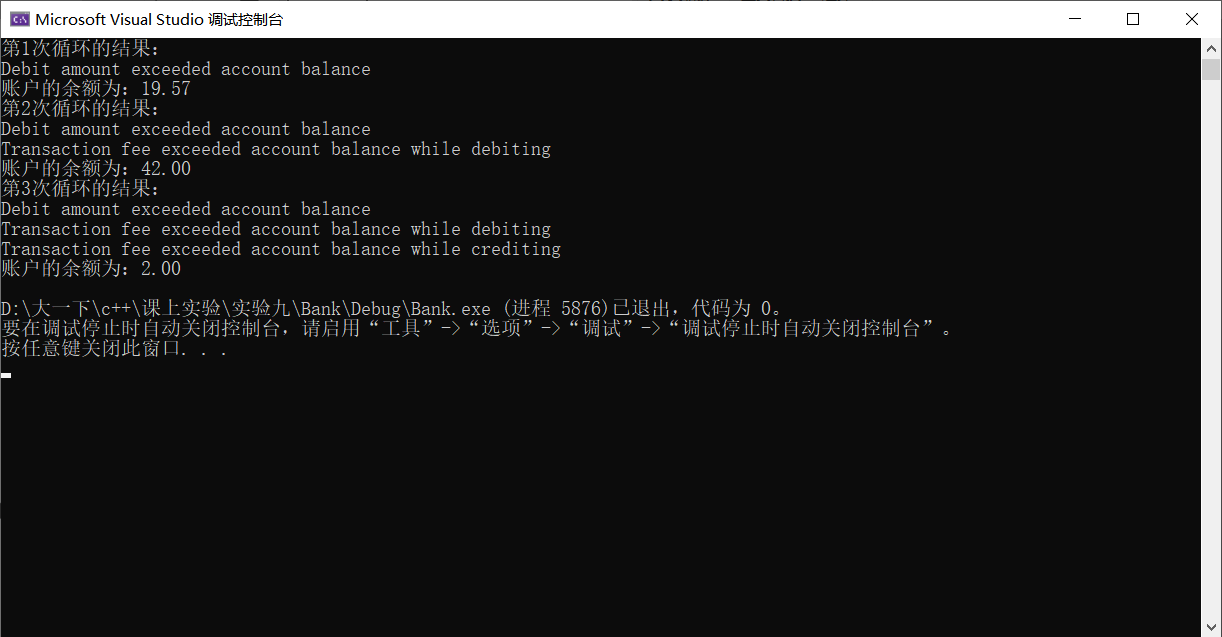
derivedPtr->credit(derivedPtr->calculateInterest());

cout << fixed << setprecision(2); //使用定点数格式，2位小数部分

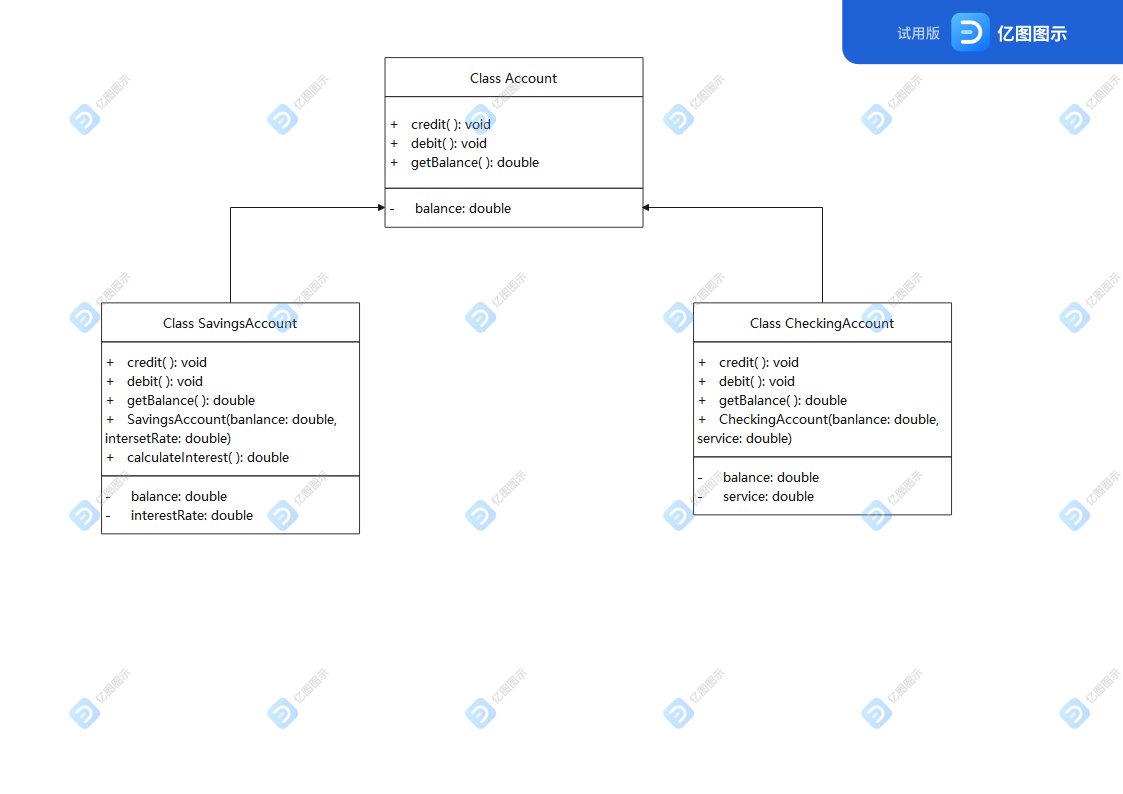
cout << "账户的余额为：" << accounts[i]->getBalance() << endl;

}

}



类图：



### Park

#include<iostream>

#include<string>

using namespace std;

class Park;

class Automobile {

public:

virtual void enter(Park\* park) {

}

virtual void leave(Park\* park) {

}

virtual string getNumber() {

return this->name;

}

virtual int getMoney() {

return this->money;

}

protected:

virtual void pay(Park& park) = 0;

private:

string name;

int money;

};

class Park {

public:

Park(int N) {

this->spaces = new Automobile \* [N];

for (int i = 0; i < N; i++)

{

this->spaces[i] = NULL;

}

this->Max = N;

}

void showInfo() {

cout << "停车场目前停放了" << this->num << "辆汽车：";

if (this->num != 0) {

for (int i = 0; i < this->num; i++)

{

cout << this->spaces[i]->getNumber() << ",";

}

}

cout << "共收入" << sumMoney << "元停车费" << endl;

}

void enter(Automobile\* SomeCar) {

if (this->num < this->Max)

{

for (int i = 0; i < this->Max; i++) {

if (this->spaces[i] == NULL)

{

this->spaces[i] = SomeCar;

this->num++;

break;

}

}

cout << SomeCar->getNumber() << "进入停车场，分配停车位" << endl;

}

else {

cout << "无法为" << SomeCar->getNumber() << "分配停车位" << endl;

}

}

void leave(Automobile\* SomeCar) {

cout << SomeCar->getNumber() << "离开停车场，缴纳停车费" << SomeCar->getMoney() << "元" << endl;

for (int i = 0; i < this->Max; i++)

{

if (this->spaces[i] == SomeCar) {

this->spaces[i] = NULL;

this->num--;

}

}

sumMoney += SomeCar->getMoney();

}

void reclaimSpace(Automobile\* Somecar) {

for (int i = 0; i < this->Max; i++)

{

if (this->spaces[i] == Somecar) {

this->spaces[i] = NULL;

this->num--;

}

}

}

void getpaid(int money) {

this->sumMoney += money;

}

~Park() {

delete[] this->spaces;

}

private:

Automobile\*\* spaces;

int num = 0;

int Max;

int sumMoney = 0;

};

class Truck :public Automobile {

public:

Truck(string name, int weight) {

this->name = name;

this->weight = weight;

}

void enter(Park\* park) {

park->enter(this);

}

void leave(Park\* park) {

park->reclaimSpace(this); // 让停车场收回停车位

pay(\*park);

//park->leave(this);

}

string getNumber() {

return this->name;

}

int getMoney() {

return this->money;

}

protected:

void pay(Park& park) {

park.getpaid(3);

cout << this->getNumber() << "离开停车场，缴纳停车费" << this->getMoney() << "元" << endl;

}

private:

string name;

int money = 3;

int weight = 0;

};

class Car :public Automobile {

public:

Car(string name, string type) {

this->name = name;

this->type = type;

}

void enter(Park\* park) {

park->enter(this);

}

void leave(Park\* park) {

park->reclaimSpace(this); // 让停车场收回停车位

pay(\*park);

//park->leave(this);

}

string getNumber() {

return this->name;

}

int getMoney() {

return this->money;

}

protected:

void pay(Park& park) {

park.getpaid(1);

cout << this->getNumber() << "离开停车场，缴纳停车费" << this->getMoney() << "元" << endl;

}

private:

string name;

int money = 1;

string type;

};

class Bus :public Automobile {

public:

Bus(string name, int people) {

this->name = name;

this->people = people;

}

void enter(Park\* park) {

park->enter(this);

}

void leave(Park\* park) {

park->reclaimSpace(this); // 让停车场收回停车位

pay(\*park);

//park->leave(this);

}

string getNumber() {

return this->name;

}

int getMoney() {

return this->money;

}

protected:

void pay(Park& park) {

park.getpaid(2);

cout << this->getNumber() << "离开停车场，缴纳停车费" << this->getMoney() << "元" << endl;

}

private:

string name;

int money = 2;

int people = 0;

};

void main() {

int N = 0;

cout << "请输入停车位数量：";

cin >> N;// 输入停车位数量，此处输入2

Park park(N);// 创建一个停车场对象

Automobile\* auto1 = new Car("鲁B-12345", "奥迪A6"); // 创建轿车对象

Automobile\* auto2 = new Truck("鲁B-23456", 15); // 创建卡车对象

Automobile\* auto3 = new Bus("鲁B-34567", 50); // 公交车对象

Automobile\* auto4 = new Car("鲁B-45678", "宝马320");// 创建轿车对象

auto1->enter(&park); // car进入停车场，分配停车位

auto2->enter(&park); // truck进入停车场，分配车位

auto1->leave(&park); // car离开停车场，缴纳停车费

auto3->enter(&park); // bus进入停车场，分配车位

/\* 显示当前停放的车辆的车牌号码，以及当前的全部停车费收入\*/

park.showInfo();

auto4->enter(&park); // car进入停车场，分配停车位

// car进入停车场，分配停车位。因为没有空余停车位，所以无法分配

auto3->leave(&park); // bus离开停车场，缴纳停车费

auto2->leave(&park); // truck离开停车场，缴纳停车费

/\* 显示当前停放的车辆的车牌号码，以及当前的全部停车费收入\*/

park.showInfo();

//return 0;

}

