بسمه تعالی

عنوان:

گزارشکار تمرین دوم

«طراحی سیستم بانک»

استاد:

مهندس نرگس بطحائیان

دانشجو:

نسترن منصوری

۹۶۱۲۳۵۸۰۴۱

ترم تحصیلی:

۴۰۰۱

فهرست

هدف: ۴

مقدمه: ۴

کلاس Date: ۵

Date.h: ۵

Date.cpp: ۶

کانستراکتور ۶

تابع set_year: ۷

تابع get_year: ۷

تابع print: ۸

کلاس Client: ۹

Client.h: ۹

Client.cpp ۱۰

تابع createAccount: ۱۰

تابع set_username ۱۱

تابع get_username ۱۱

تابع set_ip ۱۲

تابع get_ip ۱۲

تابع set_cardNumber ۱۳

تابع get_cardNumber ۱۳
تابع set_inventory ۱۳
تابع get_inventory ۱۴
تابع deposit ۱۴
تابع withdraw ۱۴
تابع set_createAccountDate ۱۵
تابع set_acounExpirationAuto ۱۵
تابع set_acounExpirationManual ۱۵
تابع print ۱۶
برنامه Main: ۱۶
تابع commands: ۱۸
تابع add_acount: ۱۹
تابع renew: ۲۰
تابع invent: ۲۱
تابع transfer: ۲۲

هدف:

هدف از پیاده سازی این تمرین طراحی و پیاده سازی سیستم بانکی است.

مقدمه:

کد پیاده سازی شده برای این تمرین به طور کلی از سه قسمت تشکیل شده است. دو کلاس و main برنامه که د رادامه به طور مفصل توضیح داده می شود.

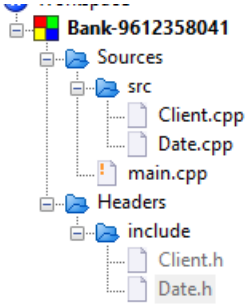

Figure ۱:اجزا پیاده سازی شده

کلاس Date:

این کلاس برای ذخیره تواریخ موجود در سیستم مورد استفاده قرار میگیرد که از دو بخش h. و cpp. تشکیل شده است.

Date.h:

include\Date.h

```
1   #ifndef DATE_H
2   #define DATE_H
3
4
5   class Date
6   {
7       public:
8           static const unsigned int month_per_year=12;    //number of month per year
9           explicit Date(int=1,int=1,int=2020);
10          void set_year(int);          //set year
11          unsigned int get_year()const;   //return year
12
13          void print() const;     //print Date
14          virtual ~Date();
15
16      private:
17          unsigned int month; //save month
18          unsigned int day;   //save day
19          unsigned int year;  //save year
20
21          unsigned int checkDay(int)const;    //Checks 31-day and 30-day months.
22  };
23
24  #endif // DATE_H
25
```

همانطر که در تصویر مشخص است، در بخش private کلاس سه متغیر از نوع unsigned int وجود دارد که به ترتیب برای ذخیره سازی ماه، روز و سال هستند. یک تابع هم وجود دارد به نام checkday که به عنوان ورودی یک متغیر از نوع integer می گیرد و برای بررسی ۳۱ یا ۳۰ روزه بودن ماه های سال است.

در بخش public متغیر ثابتی تعریف شده است از نوع integer به نام month_oper_year که نشان دهنده ی تعداد ماه های یک سال است. در خط بعدی کاسنتراکتور کلاس وجود دارد که مقادیر پیش فرض 1/1/2020 را در متغیرهای خصوصی قرار می دهد.

در خط بعد تابع set_year وجود دارد برای تغییر متغیر year که در ادامه دلیل پیاده سازی آن را متوجه خواهید شد و د

رنهایت تابع get_yearکه مقدار موجود در متغیر year را برمی گرداند.

:Date.cpp

*src\Date.cpp

```
1   #include "Date.h"
2   #include <iostream>
3   #include <stdexcept>
4   #include <array>
5
6   using namespace std;
7
8   ////////////////////////////////////////////////
9   Date::Date(int dy,int mn, int yr)
10  {
11      if (mn>0 && mn<=month_per_year) //Check the number of months.
12          month=mn;
13      else
14          throw invalid_argument("month must be 1-12");
15      year = yr; // could validate yr
16      day = checkDay( dy );    //call checkday
17
18  }
```

کتابخانه های استفاده شده در این کلاس همانطور که مشخص است به جز iostream، کتابخانه stdexcept می باشد که برای

رخ دادن استثناهاست.

کانستراکتور:

کانستاکتور این کلاس با دریافت سه متغیر صحیح به نام mn,dy,yr کا رخود را آغاز می کند. ابتدا چک میکند که مقدار متغیر

mn از 0 بزرگتر بوده و از month_per_year کوچکتر باشد که در صورت داشتن این شرط آن را در متغیر month قرار

می دهد در غیر اینصورت یک استثنا پرتاب می کند. سپسyr را در year قرار داده و برای قرار دادن متغیر dy در day تابع

checkday را فراخوانی می کند.

تابع checkday:

```
41  unsigned int Date::checkDay(int testDay) const
42  {
43     static const array< int, month_per_year + 1 > daysPerMonth =
44      { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
45
46  // determine whether testDay is valid for specified month
47      if ( testDay > 0 && testDay <= daysPerMonth[ month ] )      //Checks 31-day and 30-day months.
48      return testDay;
49
50
51      if ( month == 2 && testDay == 29 && ( year % 400 == 0 ||
52      ( year % 4 == 0 && year % 100 != 0 ) ) )
53      return testDay;
54
55      throw invalid_argument( "Invalid day for current month and year" );
56
57  }
```

این تابع به عنوان ورودی یک متغیر صحیح میگیرد به نام testDay. سپس با تعریف آرایه ای از نوع int به طول month_per_year+1 به نام daysPerMonth و ان را با مقادیری که در تصویر مشخص است مقداردهی می کند. در خط ۴۷ چک می شود که اگر ورودی تابع از 0 بزرگتر و از تعداد روزهای ماه مورد نظر کمتر بود؛ مقدار ورودی را برمی گرداند. اما اگر ماه دوم باشیم و مقدار ورودی ۲۹ باشد و سال ورودی کبیسه باشد د راینصورت هم مقدار ورودی مرد پذیرش است و برگردانده می شود. در غیر اینصورت یک استثنا پرتاب خواهد شد.

تابع set_year:

```
25  void Date::set_year(int y)
26  {
27      year=y;
28  }
```

این تابع مقدار ورودی y را در year ذخیره می کند.

تابع get_year:

```
unsigned int Date::get_year()const
{
    return year;

}
```

این تابع هم مقدار موجود در متغیر year را برمی گرداند.

تابع print:

```
36  void Date::print() const
37  {
38      cout << day << '/' << month << '/' << year; //Prints the contents of the date object.
39  }
```

این تابع تاریخ ورودی را چاپ می کند.

کلاس Client:

این کلاس برای ذخیره اطلاعات مشتریان بانک در سیستم مورد استفاده قرار میگیرد که از دو بخش h. و cpp. تشکیل شده است.

Client.h:

include\Client.h

```h
#include <string>
#include "Date.h"

#ifndef CLIENT_H
#define CLIENT_H

class Client
{
    public:
        Client();
        virtual ~Client();

        void creatAccount(std::string,std::string,int=0);     //This function creates a new account.

        void set_username(std::string);
        std::string get_username()const;   //This function check and set username

        void set_ip(std::string);
        std::string get_ip()const;     //This function check and set IP

        void set_cardNumber(std::string,std::string);        //This function sets the card number.
        unsigned int get_cardNumber()const;           //This function returns card number

        void set_inventory(char,int);       //This function sets the inventory.
        int get_inventory() const;     //This function returns the inventory.
        void deposit(int);              //This function deposits the inventory.
        void withdraw(int);             //This function withdraws the inventory.

        void set_creatAccountDate(int,int,int);        //This function sets the account opening date.
        void set_acounExpirationAuto();         //This function sets the account expiration date.
        void set_acounExpirationManual();       //This function sets the account expiration date.

        void print() const;     //This function displays information.
    private:
        std::string userName;       //Variable to store customer name.
        std::string IP;         //Variable to store client ip.

        unsigned int cardNumber;    //Variable to store card number.
        int Inventory;              //Variable to save account balance.

        Date creatAccountDate;      //Variable to save account opening date.
        Date acounExpiration;       //Variable to save account expiration date.

};
#endif // CLIENT_H

```

در بخش private دو متغیر از نوع رشته وجود دارد برای ذخیره username و IP ها. یک متغیر صحیح مثبت برای ذخیره شماره کارت. یک متغیر صحیح برای ذخیره موجودی و دو متغیر از نوع کلاس تاریخ برایذخیره تاریخ افتتاح حسب و انقضا کارت.

در بخش public:

- ✓ تابعی برای ایجاد حساب
- ✓ تابعی برای ثبت وبرگرداندن username
- ✓ تابعی برای ثبت و برگرداندن IPها
- ✓ تابعی برای ثبت و برگرداندن شماره کارت
- ✓ تابعی برای تغییر موجودی،افزایش و کاهش
- ✓ تابعی برای ثبت تاریخ افتتاح حساب
- ✓ تابعی برای چاپ اطلاعات

Client.cpp:

src\Client.cpp

```cpp
//Client.cpp definition

#include <iostream>
#include <string>
#include <cstdlib>      //for rand & srand
#include <ctime>        //for time

#include "Client.h"
```

در این بخش از کلاس مشتری از کتابخانه های iostream, string, cstdlib و ctime(برای زمان) استفاده شده است.

تابع createAccount:

```cpp
void Client::creatAccount(string u,string ip,int a)
{
    set_username(u);        //call set_username
    set_ip(ip);             //call set_ip
    set_cardNumber(u,ip);   //call set_cardNumber
    int m,d,y;
    cout<<"Enter the day, month and year of account opening, respectively."<<endl;
    cin>>d>>m>>y;
    set_creatAccountDate(d,m,y);        //call set_createAccountDate
```

این تابع دو متغیر از نوع رشته و یک متغیر از نوع integer را به عنوان ورودی دریافت می کند. سپس با هر یک از پارامترهای ورودی یک تابع را فراخوانی می کند.(توابع Set_ip و set_username و set_cardNumber) سپس با دریافت روز و ماه و

سال افتتاح از کاربر تابع Set_creatAccountDate را فراخوانی می کند. پارامترهای اول دوم به عنوان username و IP و پارامتر سوم مشخص کننده موجودی کاربر است که در هدر به صورت پیش فرض مقدار آن 0 قرار داده شد.

تابع set_username:

```
35  void Client::set_username(string name)
36  {
37
38          if(name.size()<=25)      //check size of username an substr the username if it's longer than 25 carachture
39          {
40              userName=name;
41          }
42          else
43          {
44              userName = name.substr( 0, 25 );
45              cerr << "Name :" << name << "\" exceeds maximum length (25).\n"<< "Limiting username to first 25 characters.\n" << endl;
46
47          }
48
49  }
```

این تابع تنها یک پارامتر ورودی دارد از نوع رشته. که این پارامتر را بررسی می کند که مقدار آن از ۲۵ کاراکتر بیشتر نباشد. اگر بیش از ۲۵ کاراکتر بود تنها ۲۵ تای اول آن را به عنوان نام ذخیره می کند و این موضوع را به کاربر اطلاع می دهد.

تابع get_username:

```
51  string Client::get_username()const
52  {
53      return userName;
54  }
55
```

این تابع هم نام ذخیره شده را برمی گرداند.

تابع set_ip:

```
57  void Client::set_ip(string ip)
58  {
59      string st,ipp;
60      ipp=ip;
61      bool temp=true;
62
63      for(int i=0;i<4;i++)        //In this loop, it is checked that the different
64       {                          //parts of the ip are greater than 0 and smaller
65                                   //than 255, otherwise it will throw an exception.
66          auto pos=ipp.find(".");
67          st=ipp.substr(0,pos);
68          if(!(stoi(st)>=0 && stoi(st)<=255))
69          {
70              temp=false;
71              throw invalid_argument("The IP entered is incorrect.");
72          }
73
74          ipp.erase(0,pos+1);
75
76      }
77      if (temp==true)
78          IP=ip;
79      else
80          cout<<"We have incorrect IP"<<endl;
81  }
```

این تابع پارامتر ورودی خود را که از نوع رشته است بعد از دریافت به ۴ بخش تقسیم می کند(به این دلیل که هر IP از ۴ بخش تشکیل می شود.)سپس هر یک از این ۴ بخش را به عدد صحیح تبدیل می کند و بررسی میکند که مقدار آن ا ز0 بیشتر یا مساوی و از ۲۵۵ کوچکتر یا مساوی باشد. اگر قسمتی از این پارامتر در این بازه نباشد یک استثنا پرتاب می شود.

تابع get_ip:

```
83  string Client::get_ip()const
84  {
85      return IP;
86  }
```

این تابع مقدار ip ذخیره شده را بر می گرداند.

تابع set_cardNumber:

```
87  ////////////////////////////////////////////////
88  void Client::set_cardNumber(string us,string ip)
89  {
90      srand( static_cast<unsigned int>( time( 0 ) ) );        //Here random 4-digit numbers are
91      while(1)                                                //created for the card number.
92      {
93          unsigned int number = 1000 + rand() %10000 ;
94          if(number>=1000)
95          {
96              cardNumber=number;
97              break;
98          }
99          else
100             continue;
101
102     }
103
104 }
```

در این تابع به طور رندوم یک عدد ۴ رقمی ایجاد می شود. و در متغیر cardNumber ذخیره می گردد.

تابع get_cardNumber:

```
106 unsigned int Client::get_cardNumber()const
107 {
108     return cardNumber;
109 }
```

این تابع مقدار موجود در متغیر cardNumber را بر می گرداند.

تابع set_inventory:

```
111 void Client::set_inventory(char ch,int m)
112 {
113     //Is it checked here that it should be withdrawn from the account or added to it?
114     switch (ch)
115     {
116     case 'w':
117         withdraw(m);        //call withdraw func
118         break;
119     case 'd':
120         deposit(m);         //call deposit func
121         break;
122     default:
123         throw invalid_argument("The input value must be p or d.");
124     }
125     cout<<endl;
126
127 }
```

این تابع دو پارامتر ورودی دارد.یکی نشان دهنده نوع برداشت یا واریز و دیگری برای مشخص کردن مقدار برداشتی یا واریزی می باشد. در این تابع ابتدا بررسی می شود که مقدار کاراکتر ورودی چیست؟ اگر 'w' بود تابع withdraw فراخوانی می شود و اگر 'd' بود تابع deposit فراخوانی می شود.

تابع get_inventory:

```
129   int Client::get_inventory()const
130   {
131       return Inventory;
132   }
```

این تابع هم مقدار موجودی را برمی گرداند.

تابع deposit:

```
134   void Client::deposit(int mon)
135   {
136       Inventory+=mon;
137   }
```

این تابع مقدار پارامتر ورودی را به مقدار موجودی مشتری اضافه می کند.

تابع withdraw:

```
139   void Client::withdraw(int mon)
140   {
141       int balance=get_inventory()- mon;
142       if(balance<0)           //Here it is checked that if an exception is made
143          {                    //by withdrawing from the account, an exception will be thrown
144             throw invalid_argument("The input amount is more than the inventory.");
145          }
146          else
147          {
148              Inventory-=mon;
149          }
150
151   }
```

این تابع بعد از دریافت پارامتر ورودی بررسی می کند که آیا با کم کردن این مقدار از موجودی مشتری مقدار موجود منفی خواهد بود یا خیر؟ اگر منفی باشد یک استثنا پرتاب خواهد کرد درغیر اینصورت مقدار ورودی را از موجودی کم می کند.

تابع set_createAccountDate:

```
153   void Client::set_creatAccountDate(int dy,int mn, int yr)
154   {
155       Date d(dy,mn,yr);            //Input values taken from the user were stored in
156       creatAccountDate=d;          //the variable. The automatic expiration function
157                                    //is then selected to set the expiration date.
158       set_acounExpirationAuto();
159       cout<<endl;
160
161   }
```

این تابع ابتدا یک شی با ورودی هایی که می گیرد می سازد، سپس محتوای شی ساخته شده در متغیر creatAccountDate قرار می دهد. بعد تابع set_acounExpirationAuto را فراخوانی می کند.

تابع set_acounExpirationAuto:

```
163   void Client::set_acounExpirationAuto()
164   {
165                                        //In this function, by placing the available values
166       acounExpiration=creatAccountDate;//of the opening variable in the expiration variable,
167       acounExpiration.set_year(creatAccountDate.get_year()+2);// the year value of the expiration variable increases by 2 years.
168       cout<<"Account expiration date:";
169       acounExpiration.print();
170       cout<<endl;
171   }
```

این تابع محتوای موجود در متغیر creatAccountDate را در متغیر accounExpiration ذخیره می کند. سپس با فراخوانی تابع set_year و get_year از کلاس Date مقدار سال را دوسال افزایش می دهد. و بعد تاریخ انقضا را چاپ می کند.

تابع set_acounExpirationManual:

```
173   void Client::set_acounExpirationManual()
174   {
175       int y=acounExpiration.get_year();       //This function is to extend the expiration.
176       acounExpiration.set_year(y+2);
177       set_inventory('w',5000);
178       cout<<"Account expiration new date:";
179       acounExpiration.print();
180       cout<<endl;
181   }
```

این تابع ابتدا سال تاریخ انقضا را در یک متغیری به نام y ذخیره می کند. سپس متغیر سال تاریخ انقضا را ۲ سال افزایش می دهد و بعد تابع set_inventory را فراخوانی می کند تا مبلغ مورد نیاز برای تمدید حساب(۵۰۰۰) از موجودی مشتری کاسته شود.

تابع print:

```
183  void Client::print() const
184  {
185      cout<<"Account No. : "<<IP<<endl;       //By calling this function, customer information is displayed.
186      cout<<"Account Holder Name : "<<userName<<endl;
187      cout<<"Balance amount : "<<Inventory<<endl;
188  }
189  ////////////////////////////////////////////////////
190
```

این تابع هم اطلاعات مشتریان را چاپ می کند.

برنامه: Main

main.cpp

```
1   #include <iostream>
2   #include <string>
3   #include <fstream>
4   #include <cctype>
5   #include <iomanip>
6
7   #include "Client.h"
8   #include "Date.h"
9
10  using namespace std;
11
12  void commands();
13  void add_acount(string,string);      //Function to create a new account
14  void renew(string,string);       //Function for account renewal
15  void invent(string,string,int);      //Function to reduce and increase inventory
16  void transfer(string,string,string,int);        //Function for transferring money
17
```

همانطور که در تصویر مشخص است، از کتابخانه های بالا در این قسمت کد استفاده شده است. همچنین هردو کلاس Client و Date هم include شده اند. در ادامه ۵ تابع داریم که توضیح داده خواهد شد.

```
19  int main()
20  {
21      cout << "--------Welcome to the bank---------"<<endl;
22      cout<<"Please use the guide below to enter commands." << endl;
23      cout<<"_____________________________________________________________________________________"<<endl<<endl;
24      commands();
25      cout<<"_____________________________________________________________________________________"<<endl;
26      do
27      {
28          string strl="",us="",ipAccount="";
29          int money=0;
30          cout<<">>> ";
31          getline(cin,strl);
32          cout<<strl<<endl;
33
34          if(strl.substr(0,6)=="create")      //In this condition, if the input value is "create"
35          {                                   // by the user, the "add" function will be called
36              strl.erase(0,7);                //after separating the components.
37              auto pos=strl.find(":");
38              us=strl.substr(0,pos);
39              ipAccount=strl.substr(pos+1);
40              add_acount(us,ipAccount);
41
42          }
43
```

در شروع main بعد از خوش آمد گویی به کاربر و فراخوانی تابع راهنمای دستورات وارد یک حلقه do...while می شود.سه متغیر از نوع رشته تعریف شده است که برای ذخیره ورودی کاربر و username و IP هستند. همچنین یک متغیر دیگر هم از نوع صحیح وجود داردکه مبلغ موردنظر مشتری در آن ذخیره می شود. از کاربر میخواهد تا دستور مورد نظر را وارد کند.

بعد از ورود دستور برنامه شروع به چک کردن دستورات می کند. به این صورت که اگر کله ی اول وارد شده توسط کاربر "create" باشد بعد از جدا سازی username و IP از هم تابع add_account فراخوانی می شود.

```
44          if(strl.substr(0,7)=="renewal")     //In this condition, if the input value is "renewal"
45          {                                   //by the user, the "renew" function will be called
46             strl.erase(0,8);                 //after separating the components.
47             auto pos=strl.find(":");
48             us=strl.substr(0,pos);
49             ipAccount=strl.substr(pos+1);
50             renew(us,ipAccount);
51
52          }
53
```

اگر دستور وارد شده با "renewal" شروع شود تابع renew بعد از جداسازی username و IP فراخوانی می شود.

```
54          if(strl.substr(0,7)=="deposit")     //In this condition, if the input value is "deposit"
55          {                                   //by the user, the "invent" function will be called
56             strl.erase(0,8);                 //after separating the components.
57             auto pos=strl.find(":");
58             us=strl.substr(0,pos);
59             strl.erase(0,pos+1);
60             auto pos2=strl.find(":");
61             ipAccount=strl.substr(0,pos2);
62             strl.erase(0,pos2+1);
63             money=stoi(strl);
64            invent(us,ipAccount,money);
65          }
```

اگر دستور وارد شده با "deposit" شروع شود تابع invent بعد از جداسازی username و IP فراخوانی می شود.

```
67          if(strl.substr(0,8)=="withdraw")    //In this condition, if the input value is "withdraw"
68          {                                   //by the user, the "invent" function is called after
69             strl.erase(0,9);                 //separating the components.
70             auto pos=strl.find(":");
71             us=strl.substr(0,pos);
72             strl.erase(0,pos+1);
73             auto pos2=strl.find(":");
74             ipAccount=strl.substr(0,pos2);
75             strl.erase(0,pos2+1);
76             money=stoi(strl);
77            invent(us,ipAccount,money);
78          }
```

اگر دستور وارد شده با "withdraw" شروع شود تابع invent بعد از جداسازی username و IP فراخوانی می شود.

```
80          if(strl.substr(0,8)=="transfer")    //In this condition, if the input value is "transfer"
81          {                                   //by the user, the "transfer" function will be called
82             string us2="";                   //after separating the components.
83             strl.erase(0,9);
84             auto pos=strl.find(":");
85             us=strl.substr(0,pos);
86             strl.erase(0,pos+1);
87             auto pos2=strl.find(":");
88             ipAccount=strl.substr(0,pos2);
89             strl.erase(0,pos2+1);
90             auto pos3=strl.find(":");
91             us2=strl.substr(0,pos3);
92             strl.erase(0,pos3+1);
93             money=stoi(strl);
94             transfer(us,ipAccount,us2,money);
95          }
```

اگر دستور وارد شده با "transfer" شروع شود تابع transfer بعد از جداسازی username و IP فرستده و username گیرنده فراخوانی می شود.

تابع commands:

```
103 void commands()
104 {
105    cout<<"------------COMMANDS:------------"<<endl;
106    cout<<"Enter the following command to CREATE A NEW ACCOUNT:"<<endl;
107    cout<<"\tcreate username:uaserIP"<<endl<<endl;
108    cout<<"Enter the following command to RENEW YOUR ACCOUNT:"<<endl;
109    cout<<"\trenew username:uaserIP"<<endl<<endl;
110    cout<<"Enter the following command to WITHDARW FROME THE ACCOUNT:"<<endl;
111    cout<<"\twithdraw username:uaserIP:money"<<endl<<endl;
112    cout<<"Enter the following command to DEPOSIT:"<<endl;
113    cout<<"\tdeposit username:uaserIP:money"<<endl<<endl;
114    cout<<"Enter the following command to TRANSFER:"<<endl;
115    cout<<"\tcreate senderUsername:senderUserIP:receiver:money"<<endl<<endl;
116
117 }
```

این تابع دستوراتی که کاربر لازم است وارد کند را به عنوان راهنما به او شنان می دهد.

تابع add_acount:

```
119   void add_acount(string a,string b)
120   {
121       Client cl;  //Creates an object
122       ofstream outFile;   //Opens a file to write account information in.
123       outFile.open("Clients.dat",ios::binary|ios::app);
124       try
125       {
126           cl.creatAccount(a,b);   //call createAccount from class Client
127       }
128       catch (invalid_argument &e)
129       {
130           cout<<endl<<"Exception occurred: " << e.what() << endl;
131       }
132       outFile.write(reinterpret_cast<char *> (&cl), sizeof(Client)); //Close file
133       outFile.close();
134   }
135
```

این تابع برای ایجاد حساب جدید فراخوانی می شود.ابتدا یک شی از کلاس مشتری ایجاد میک ند سپس فایلی را به نام Clients باز میکند و اطلاعات مشتری را در آن وارد می کند. این کار با فراخوانی تابع creatAcccount از کلاس مشتری انجام می دهد. در انتها فایل را می بندد.

تابع renew:

```
137  void renew(string us,string ip)
138  {
139      Client c2;  ///Creates an object
140      //char ch;
141      fstream File;
142      bool found=false;
143      File.open("Clients.dat", ios::binary|ios::in|ios::out);
144      if(!File)
145      {
146          cout<<"File could not be open !! Press any Key...";
147          return;
148      }
149      while(!File.eof() && found==false)  //Checks the username and IP in the file.
150      {
151          File.read(reinterpret_cast<char *> (&c2), sizeof(Client));
152          if(c2.get_username()==us || c2.get_ip()==ip)
153          {
154             try
155             {
156                 c2.set_acounExpirationManual();  //call set_acounExpirationManual from class Client
157             }
158             catch (invalid_argument &e)
159             {
160                 cout<<endl<<"Exception occurred: " << e.what() << endl;
161             }
162          }
163          int pos=(-1)*static_cast<int>(sizeof(c2));
164          File.seekp(pos,ios::cur);
165          File.write(reinterpret_cast<char *> (&c2), sizeof(Client));
166          cout<<endl<<"\t\t Record Updated"<<endl;
167          found=true;
168      }
169      File.close();
170      if(found==false)
171          cout<<"\n\n Record Not Found ";
172  }
```

این تابع برای تمدید انقاضا حساب مشتریان فراخوانی می شود.به این وصرت که بعد از ایجاد یک شی از کلاس مشتری و باز کردن فایل Clients در آن به دنبال مشتری می گردد که اطلاعاتش را به عنوان پارامتر ورودی دریافت کرده است. در صورت پیدا کردن مشتری، تابع set_acountExpirationManual را فراخوانی کرده و تاریخ انقضا را تمدید می کند. اگر نتواند مشتری را پیدا کند پیغام "Record Not Found" را می دهد.

تابع invent:

```
175  void invent(string us,string ip,int mny)
176  {
177      Client c3;  /////Creates an object
178      char ch;
179      cout<<"DEPOSIT OR WITHDRAW:(d,w)\t";    //Asks the user to select withdrawal or deposit.
180      cin>>ch;
181      cout<<endl;
182      fstream File;
183      bool found=false;
184      File.open("Clients.dat", ios::binary|ios::in|ios::out);
185      if(!File)
186      {
187          cout<<"File could not be open !! Press any Key...";
188          return;
189      }
190      while(!File.eof() && found==false)  //Checks the username and IP in the file.
191      {
192          File.read(reinterpret_cast<char *> (&c3), sizeof(Client));
193          if(c3.get_username()==us || c3.get_ip()==ip)
194          {
195               try
196             {
197                 c3.set_inventory(ch,mny);  //call set_inventory from class Client
198             }
199             catch (invalid_argument &e)
200             {
201                 cout<<endl<<"Exception occurred: " << e.what() << endl;
202             }
203
204          }
205          int pos=(-1)*static_cast<int>(sizeof(c3));
206          File.seekp(pos,ios::cur);
207          File.write(reinterpret_cast<char *> (&c3), sizeof(Client));
208          cout<<endl<<"\t\t Record Updated"<<endl;
209          found=true;
210      }
211      File.close();
212      if(found==false)
213          cout<<"\n\n Record Not Found ";
214  }
```

این تابع برای برداشت و واریز از حساب مشتری تعریف شده است. به این صورت عمل می کند که بعد از ایجاد یک شی از کلاس مشتری از کاربر میخواهد که مشخص کند میخواهد برداشت انجام دهد یا واریز(با ورود کارکتر d یا w).سپس فایل Clients را باز کرده و در آن به دنبال مشتری با اطلاعات ورودی می گردد. مانند تابع قبل اگر مشتری موردنظر را پیدا کند تابع set_inventory را با کارکتر ورودی توسط کاربر و مقدار پول فراخوانی می کند. در صورت نیافتن مشتری هم مانند تابع قبل پیغام می دهد.

تابع transfer:

```
void transfer(string us,string us2,string ip,int mny)
{
    Client c4;  /////Creates an object
    Client c5;  /////Creates an object

    fstream File;
    bool found=false;
    File.open("Clients.dat", ios::binary|ios::in|ios::out);
    if(!File)
    {
        cout<<"File could not be open !! Press any Key...";
        return;
    }
    while(!File.eof() && found==false)
    {
        File.read(reinterpret_cast<char *> (&c4), sizeof(Client));
        if((c4.get_username()==us || c4.get_ip()==ip)&&(c5.get_username()==us2))    //Checks the username and IP in the file.
        {
              try
           {
               c4.set_inventory('w',mny);   //call set_inventory from class Client for withdraw
               c5.set_inventory('d',mny);  //call set_inventory from class Client for deposit
           }
           catch (invalid_argument &e)
           {
               cout<<endl<<"Exception occurred: " << e.what() << endl;
           }


        }
        int pos=(-1)*static_cast<int>(sizeof(c4));
        File.seekp(pos,ios::cur);
        File.write(reinterpret_cast<char *> (&c4), sizeof(Client));
        cout<<endl<<"\t\t Record Updated"<<endl;
        found=true;
    }
    File.close();
    if(found==false)
        cout<<"\n\n Record Not Found ";
}
//////////////////////////////////////////////////////////////
```

در این تابع برعکس توابع دیگر اطلاعات دو مشتری گرفته می شود یکی کسی که میخواهد پول را انتقال دهد و دیگری کسی که پول به حسابش وریز خواهد شد. بعد از ایجاد دو شی از کلا سمشتری و باز کردن فایل ابتدا به دنبال اطلاعات فرستده میگردد سپس به دنبال اطلاعات گیرنده به این دلیل که هردو باید در بانک حساب داشته باشند. بعد از یافتن مشتریان موردنظر تابع set_inventory را با کاراکتر 'w' برای فرستنده فراخوانی می کند تا مقدار مبلغ موردنظر از حساب او کاسته شود. سپس تابع set_inventory را با کاراکتر 'd' برای گیرنده فراخوانی می کند تا مبلغ موردنظر به حساب او واریز شود.در صورت پیدا نکردن اطلاعات هر یکاز مشتریان در فایل پیغام "Record Not Found" را چاپ می کند.