成绩：_____

南京信息工程大学

课程设计（数据结构与算法）

题目_____

院、系 沃特福德学院 专业_____

学号姓名 _____ 学号姓名_____

学号姓名 _____ 学号姓名_____

指导教师_____文学志_____

二O二三 年 十二 月 二十 日

目　录

Back Account Management System

202283890036

（全部英文）南京信息工程大学沃特福德学院，南京 210044

Abstract:

The Bank Account Management System is a program that manages bank accounts using a doubly linked list. The system allows users to create, delete, and modify accounts, as well as deposit and withdraw funds. The program is designed to be user-friendly and efficient, with a simple interface that allows users to quickly access the information they need. The use of a doubly linked list ensures that the system is fast and reliable, with minimal downtime. Overall, the Bank Account Management System is an effective tool for managing bank accounts and ensuring that users have access to the information they need when they need it.

Keywords：Bank Account Management System

1 Introduction

This program is written in Java language. I used a doubly linked list to implement account creation and deletion, and a traversal method to search for accounts, and then perform operations such as deposit and withdrawal.2 Development process

2 Structure design

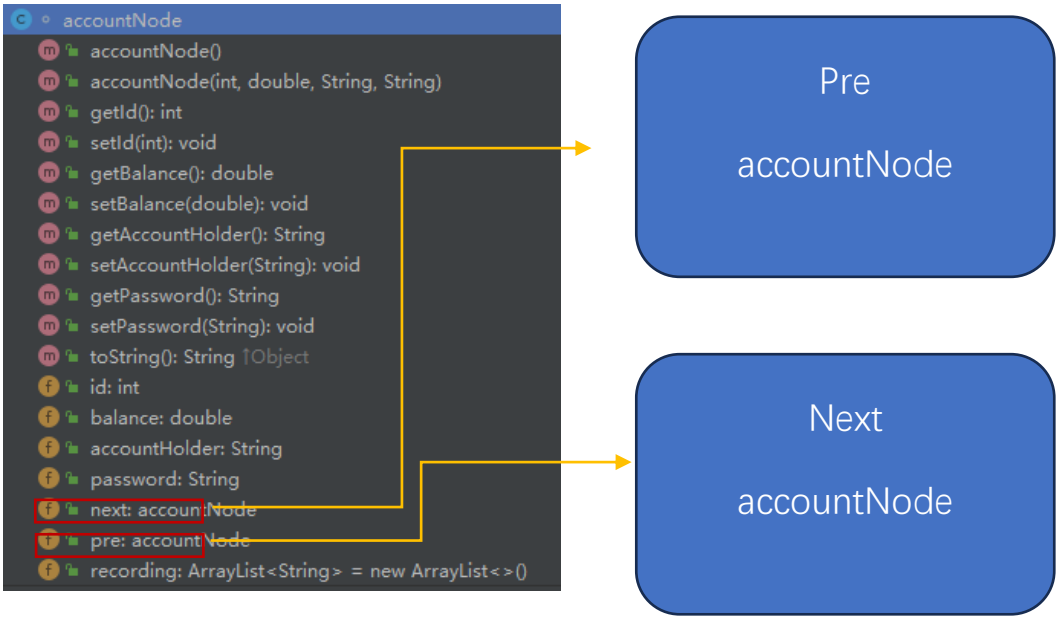
This bank account management system consists of three main parts to achieve all the functions. These parts may include:

2.1 Account information storage:

In this section, we defined a new class named accountNode with four attributes: account ID, account holder, whether the account has a balance, and account password. Each attribute has a get and set method to set and store the accountNode properties[7].

Each accountNode has a pre and a next to connect the previous accountNode and the next accountNode.

The accountNode also has a recording array of type String to record the current account changes.



2.2 Account management functions:

In this section, including account opening, account closing, inquiry, deposit, withdrawal and other functions, so that users can easily manage their accounts [3][5].

2.2.1 addLast

"addLast" function is used to add a new accountNode to the last of the linkList. An accountNode "headNode" is added in this function as sentinel node to simplify the code and prevent error[6].

This method first uses the "contain" method to determine whether the id contained in this accountNode has already appeared in the linkList. Only when this id has never appeared before, the newly added accountNode will be added to the linked list.

When a new accountNode is successfully added to the linkList, the size of the linked list increases by 1.

```
public void addLast(accountNode account) {
    if(contain(account.id) != true) {
        accountNode temp = headNode;
        while(true){
            if(temp.next == null){
                break;
            }
            temp = temp.next;
        }
        temp.next = account;
        account.pre = temp;
        size++;
    }
    System.out.println("this account id is exist");
}
```

Figure 1：code of addLast function

2.2.2 addFirst

The addFirst method is used to add the newly set accountNode to the head of the linked list. Like addFirst, it uses a sentinel node called headNode to simplify the program and avoid errors.

This method first uses the “contain” method to determine whether the id contained in this accountNode has already appeared in the linkList. Only when this id has never appeared before, the newly added accountNode will be added to the linked list.

When a new accountNode is successfully added to the linkList, the size of the linked list increases by 1.

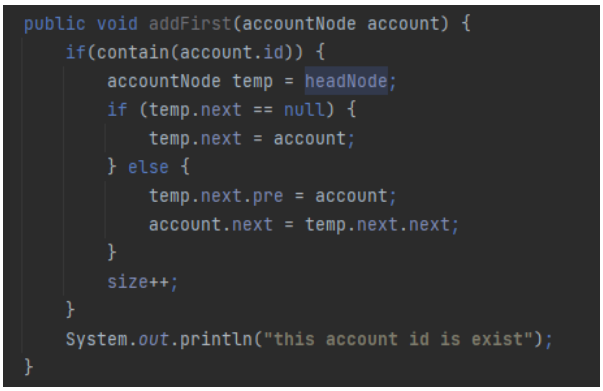


Figure 2: code of addFirst function

2.2.3 contain

The “contain” method is used to identify whether the id passed in this method already exists in the linked list. When using this method, an integer type data is first input as the id, and a while(true) loop is used to determine whether it exists[2]. If it does not exist, a Boolean variable with a value of true will be returned. If it exists, a Boolean variable with a value of false will be returned.

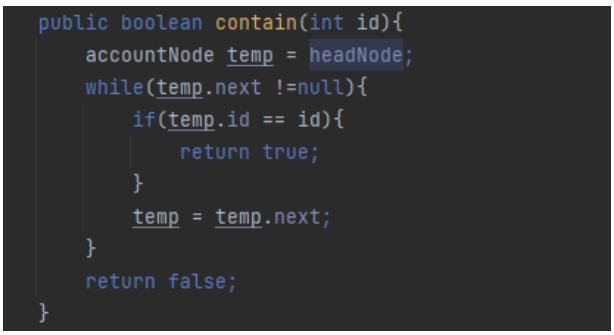


Figure 3: code of contain function

2.2.4 delete

When using this function, you will enter the ID of the accountNode you want to delete into the function.

The delete method will first find the accountNode corresponding to this ID in the linked list, and then point the previous node to the next node.

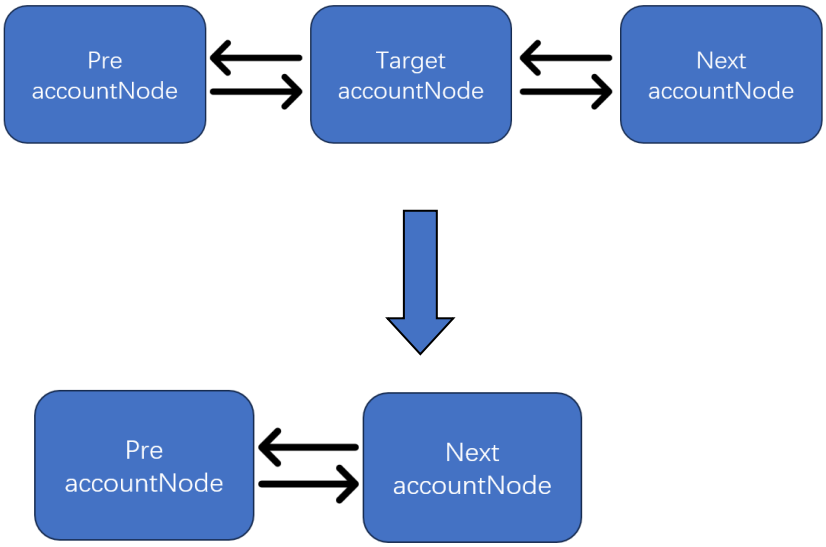


Figure 4: sketch map of delete function

This method first traverses the linked list to search for the accountNode corresponding to the input id. If it is found, the method uses the delete method described above to delete the accountNode and prints “delete correctly” to the console. If the entire linked list is traversed and the accountNode corresponding to the input id is not found, the method prints “can’t find this account” to the console and exits.

```java
public void delete(int id) {
    accountNode temp = headNode;
    while(true){
        if(temp.next == null){
            if(temp.id == id){
                temp.pre.next = null;
                System.out.println("delete correctly");
                break;
            }
            System.out.println("can't find this account");
            break;
        }else{
            if(temp.id == id){
                temp.pre.next = temp.next;
                temp.next.pre = temp.pre;
                System.out.println("delete correctly");
                break;
            }
        }
        temp = temp.next;
    }
}
```

Figure 4: code of delete function

2.2.5 getSize

In the Bank class, an int variable named size is created to get the size of the linked list and return it. Whenever the addFirst or addLast method is used, size is incremented by 1.

2.2.6 find

The find function is used to search for the accountNode corresponding to the input id.

The method uses a while(true) loop to traverse the linked list to implement the search functionality. If the entire linked list is traversed and the accountNode is not found, the method returns an empty accountNode “temp” and prints “can’t find this account” to the console. If the accountNode corresponding to the input id is found, it is assigned to temp and returned.

```java
public accountNode find(int id) {
    accountNode temp = headNode;
    while(true){
        if(temp.next == null){
            if(temp.id == id){
                return temp;
            }
            System.out.println("can't find this account");
            break;
        }else{
            if(temp.id == id){
                return temp;
            }
        }
        temp = temp.next;
    }
    return null;
}
```

Fugure 5: code of find function

2.2.7 print

This method is used to print the information stored in accountNode objects in the current linked list to the console.

The method first checks if size is empty. If size=0, the method prints “this system doesn’t have account” to the console. If size is not 0, the method uses a for loop and getDataByIndex to print the information stored in all accountNode objects in the system to the console.

```java
public void print() {
    accountNode temp = headNode;
    if(getSize() == 0){
        System.out.println("this system dones't have account");
        return;
    }
    System.out.println("id\taccountHolder\tbalance\tpassword\t");
    for(int i =1; i <= getSize();i++){
        System.out.println(getDataByIndex(i).getId()+ "\t\t" +getDataByIndex(i).getAccountHolder()+"\t\t\t" +getDataByIndex(i).getBalance()+"\t\t"+getDataByIndex(i).getPassword()+"\t\t");
    }
}
```

Fugure 6: code of find function

2.2.8 getDataByIndex

This method is used to search for an accountNode stored in a linked list by its index.

The method first checks if the index is positive, and then initializes a counter variable i to 0 and a temp variable to the headNode. The method then enters a while loop that continues indefinitely. Within the loop, the method checks if the next field of temp is null.

If it is, the method checks if the counter variable i is equal to the index. If it is, the method returns temp. If i is not equal to the index, the method prints a message to the console indicating that the account cannot be found. If the next field of temp is not null, the method checks if i is equal to the index[8]. If it is, the method returns temp. If i is not equal to the index, the method increments i and sets temp to the next field of temp. If the method exits the while loop without returning a value, it returns null.

```
public accountNode getDataByIndex(int index) {
    if (index < 0) {
        throw new IndexOutOfBoundsException("Index should be positive: " + index);
    }
    int i = 0;
    accountNode temp = headNode;
    while (true) {
        if (temp.next == null) {
            if (i == index) {
                return temp;
            }
            System.out.println("can't find this account");
            break;
        } else {
            if (i == index) {
                return temp;
            }
        }
        temp = temp.next;
        i++;
    }
    return null;
}
```

Fugure 7: code of getDataByIndex

2.3 User interface:

Provide a simple and easy-to-use user interface so that users can quickly access their account information and perform operations[1].

This sentence is in Chinese. Here is the translation: “The user interface is set inside the main method. When the program is launched, a new Bank class is generated using a parameter less constructor, followed by the creation of a user interface.

```
while(true){
    System.out.println("welcome to bank management system");
    System.out.println("1:add count");
    System.out.println("2:delete count");
    System.out.println("3:add balance by serial number");
    System.out.println("4:reduce balance by serial number");
    System.out.println("5:show all account");
    System.out.println("6:changeAccount");
    System.out.println("7:showRecording");
    System.out.println("8:exit");
    System.out.println("please enter your choice");
    Scanner sc = new Scanner(System.in);
    String choose = sc.next();
    switch (choose){
        case"1" -> addCount(bank1);
        case"2" -> deleteCount(bank1);
        case"3" -> addBalance(bank1);
        case"4" -> reduceBalance(bank1);
        case"5" -> showAccount(bank1);
        case"6" -> changeAccount(bank1);
        case"7" -> showRecording(bank1);
        case"8" -> {
            System.out.println("EXIT");
            //break loop;
            System.exit( status: 0);
        }
        default -> System.out.println("don't have this choice");
    }
}
```

Figure 8: user interface panel

2.3.1 add Count

It creates a new accountNode object and sets its properties using user input. Then it adds the new account to the end of the bank1 list. Finally, it prints the size of the bank1 list.

```java
public static void addCount(bank bank1){
    accountNode account = new accountNode();
    Scanner sc = new Scanner(System.in);
    System.out.println("please import account id(five-digit integer)");
    int id = sc.nextInt();
    if(bank1.contain(id) != true) {
        account.setId(id);
        System.out.println("please import account balance");
        double balance = sc.nextInt();
        account.setBalance(balance);
        System.out.println("please import account accountHolder");
        String accountHolder = sc.next();
        account.setAccountHolder(accountHolder);
        System.out.println("please import account password");
        String password = sc.next();
        account.setPassword(password);
        bank1.addLast(account);
    }
    else{
        System.out.println("this account id is exist");
    }
}
```

Figure 9: code of add count

2.3.2 delete Count

It prompts the user to enter the account ID of the account they want to delete. Then it calls the delete method of the bank1 object with the account ID as an argument.

```java
public static void deleteCount(bank bank1) {
    Scanner sc = new Scanner(System.in);
    System.out.println("please import account id you want to delete");
    int id = sc.nextInt();
    bank1.delete(id);
}
```

Figure 10: code of delete count

2.3.3 add balance

Adds a specified amount of balance to an account in a bank. It prompts the user to enter the account ID of the account they want to add balance to and the amount of balance they want to add. Then it adds the specified balance to the

account's current balance and records the transaction in the account's transaction history.

```
public static void addBalance(bank bank1){
    Scanner sc = new Scanner(System.in);
    System.out.println("please import account id you want to add");
    int id = sc.nextInt();
    System.out.println("please import the add balance");
    double b1 = sc.nextInt();
    double result = b1 + bank1.find(id).balance;
    bank1.find(id).setBalance(result);
    bank1.find(id).recording.add("add"+"\t" +b1 +"\t"+"into the account");
}
```

Figure 11: code of add balance

2.3.4 reduce balance

Reduces the balance of an account in a bank. It prompts the user to enter the account ID of the account they want to reduce the balance of and the amount of balance they want to reduce[9]. If the specified amount is greater than the current balance of the account, it prints a message saying that the account does not have enough money. Otherwise, it subtracts the specified amount from the account's current balance and records the transaction in the account's transaction history.

```
public static void reduceBalance(bank bank1){
    Scanner sc = new Scanner(System.in);
    System.out.println("please import account id you want to reduce");
    int id = sc.nextInt();
    System.out.println("please import the reduce balance");
    double b1 =sc.nextInt();
    if(b1 > bank1.find(id).balance){
        System.out.println("Sorry,you don't have enough money");
    }else{
        double result = bank1.find(id).balance - b1;
        bank1.find(id).setBalance(result);
        bank1.find(id).recording.add("the account reduce by"+"\t"+ b1 +"\t"+"yuan");
    }
}
```

Figure 12: code of reduce balance

2.3.5 show all account

This code displays the bank accounts. It use print method of the bank1 object created above, which prints the bank account list.

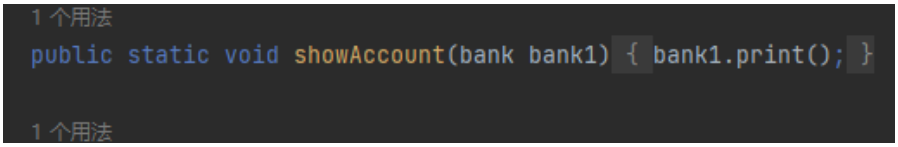


Figure 13: code of show account

2.3.6 change account

This code that allows you to add or reduce the balance of an account in a bank. It prompts the user to enter the account ID of the account they want to modify and the operation they want to perform (add or reduce). If the user chooses to add balance, it prompts them to enter the amount of balance they want to add. Then it adds the specified balance to the account's current balance and records the transaction in the account's transaction history. [4]If the user chooses to reduce balance, it prompts them to enter the amount of balance they want to reduce. If the specified amount is greater than the current balance of the account, it prints a message saying that the account does not have enough money. Otherwise, it subtracts the specified amount from the account's current balance and records the transaction in the account's transaction history.

```java
public static void changeAccount(bank bank1) {
    if (bank1.getSize() == 0) {
        bank1.print();
    } else {
        bank1.print();
        Scanner sc = new Scanner(System.in);
        int id = 0;
        String operation;
        while (true) {
            System.out.println("please enter the id you want to add or reduce ");
            id = sc.nextInt();
            if (bank1.find(id) == null) {
                System.out.println("please enter correct id");
            } else {
                break;
            }
        }
        System.out.println(bank1.find(id));
        while (true) {
            System.out.println("please choice your operation");
            operation = sc.next();
            if (operation.equals("add") || operation.equals("reduce")) {
                break;
            } else {
                System.out.println("please choice correct operation");
            }
        }
        if (operation.equals("add")) {
            System.out.println("please enter the balance you want to add");
            int balance = sc.nextInt();
            double temp = bank1.find(id).getBalance() + balance;
            bank1.find(id).setBalance(temp);
            System.out.println("operate successfully");
            bank1.find(id).recording.add("add" + "\t"+balance +"\t"+ "into the account");
        } else {
            System.out.println("please enter the balance you want to reduce");
            int balance = sc.nextInt();
            if (balance < bank1.find(id).getBalance()) {
                double temp = bank1.find(id).getBalance() - balance;
                bank1.find(id).setBalance(temp);
                System.out.println("operate successfully");
                bank1.find(id).recording.add("the account reduce by"+"\t" + balance + "\t"+"yuan");
            } else {
                System.out.println("Sorry,you don't have enough money");
            }
        }
    }
}
```

Figure 13: code of change account

2.3.7 show recoding

This code that displays the transaction history of an account in a bank. It prompts the user to enter the account ID of the account they want to view the transaction history of. Then it prints the transaction history of the account using a for loop.

```java
public static void showRecording(bank bank1){
    Scanner sc = new Scanner(System.in);
    System.out.println("please import account id you want to add");
    int id = sc.nextInt();
    for (int i = 0; i < bank1.find(id).recording.size(); i++) {
        System.out.println(bank1.find(id).recording.get(i));
    }
}
```

Figure 15: code of show recording

2.3.8 exit

3 Algorithmic analysis

Linear search, also known as sequential search, is the simplest search algorithm. It works by comparing each element of an array with the target value sequentially from the start of the array until the end.

We used this kind of algorithm in find function.

```
public void print() {
    accountNode temp = headNode;
    if(getSize() == 0){
        System.out.println("this system dones't have account");
        return;
    }
    System.out.println("id\taccountHolder\tbalance\tpassword\t");
    for(int i =1; i <= getSize();i++){
        System.out.println(getDataByIndex(i).getId()+ "\t\t" +getDataByIndex(i).getAccountHolder()+"\t\t\t" +getDataByIndex(i).getBalance()+"\t\t"+getDataByIndex(i).getPassword()+"\t\t");
    }
}
```

Linear search is simple and intuitive, it is not efficient. It is suitable when the data to be searched is small, as it can reduce the error rate. However, when the data to be searched is large, using this method is like finding a needle in a haystack, and the efficiency will be very low. Therefore, the choice of algorithm should be based on the specific situation.

4 Test result

```
welcome to bank management system
1:add count
2:delete count
3:add balance by serial number
4:reduce balance by serial number
5:show all account
6:changeAccount
7:showRecording
8:exit
please enter your choice
```

```
please enter your choice
3
please import account id you want to add
001
please import the add balance
14000
```

5 Summarize

This paper summarizes a bank account management system that uses a doubly linked list to implement various functions. Adding and deleting accounts are operations on the linked list itself, while increasing and decreasing deposits are operations that change the information stored in the linked list. Next, the author plans to use the quicksort function to sort the contents stored in the linked list according to the size of the ID in the next practice[10]. Since the main purpose of this paper is to summarize the implementation of the bank account management system, we did not discuss the application of algorithms in detail. However, we plan to use the quicksort function to sort the contents stored in the linked list according to the size of the ID in the next practice. This is a good idea because quicksort is an efficient sorting algorithm that can sort large amounts of data in a short time.

Reference（列举至少 10 篇文献，其中英文文献至少 3 篇）

[1] Hassan Takabi, Nhien-AnLe-Khac.(2019) Security privacy and digital forensics in the cloud. Beijing: Higher Education Press

[2] Jay Wengrow.(2019) A Common-Sense Guide to Data Structure and Algorithms. Beijing: People's Posts and Telecommunications Press

[3]Brelt Lantz. (2019) Machine Learning with R: Expert techniques for predictive modeling, 3rd Edition. Packet Publishing.

[4] 萧博庠《以人因工程观点评估销售点系统》

[5] 候关士《基于 Windows 平台的 WEB 技术应用研究》

[6] 蔡厚辉《以使用性评估陆客导览学习系统建置之研究》

[7] 张士亮《基于 HS3282 的Γ OCT18977 数据收发》

[8]严蔚敏. 2007 数据结构 C 语言[M]。清华大学出版社

[9]MarkM.Meerschaert.(2005) Mathematical Modeling Methods and Analysis[M]. China Machine Press.

[10] 范德宝，于晓聪，丁伟祥.提高数据结构课程教学效果的探讨[J].黑龙江科技信息，2007