超星大雅 相似度分析 www.dayainfo.com

全文检测报告

基本信息

报告编号：20231205132412118E4C412490

检测文献：Back Account Management System

过滤操作：已过滤自引"黄家睿"的相似影响

作　　者：黄家睿

检测范围：大雅全文库

检测时间：2023-12-05 09:31:21

检测结论

总文献相似度	文献相似度（去除自引、参考）	去除可能自引文献相似度	去除参考文献相似度	文献原创度	正文字符数
2.96%	2.96%	2.96%	2.96%	97.04%	10396

单篇最大相似度：0.54%

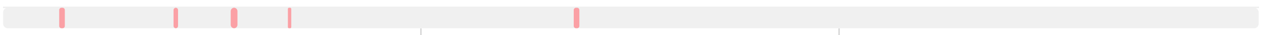
最相似文献名称：以人因工程观点评估销售点系统

相似文献类型分布

相似图书：1.55%（161字符数）

相似网络文档：1.03%（107字符数）

相似片段分布



前部		中部		尾部	
最密集相似段：	0	密集相似段：	0	非密集相似段：	5
前部相似段：	4	中部相似段：	1	尾部相似段：	0

相似文献详情

相似图书　　相似度：1.55%（161字）

序号	题名	作者	出处	相似度	是否引用
1	云之安全、隐私与数字取证	陈磊;哈桑·塔卡比（HassanTakabi）;N.A 勒哈克（Nhien-AnLe-Khac）	北京：高等教育出版社，2019.08	0.4%	否
2	R语言机器学习参考手册 英文	丘祐玮	南京：东南大学出版社，2016.01	0.4%	否

相似网络文档　　相似度：1.03%（107字）

序号	题名	作者	相似度	是否引用
1	以人因工程观点评估销售点系统	萧博庠	0.54%	否
2	基于Windows平台的WEB技术应用研究	候关士	0.37%	否
3	以使用性评估陆客导览学习系统建置之研究	蔡厚辉	0.37%	否
4	基于HS3282的ГOCT18977数据收发	张士亮	0.34%	否

北京大雅因和信息技术有限公司 相似度检测专用章

全文对比

成绩:
EMBED CorelDRAW.Graphic.11
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:
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.
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.
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.
addLast
“addLast” function is used to add a new accountNode to the last of the linkList. A accountNode “headNode” is added in this function as sentinel node to simplify the code and prevent error.
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 1: code of addLast function
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
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. 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
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.
Figure 4: code of delete function
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.
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.

Fugure 5: code of find function

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.

Fugure 6: code of find function

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. 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.

Fugure 7: code of getDataByIndex

User interface:

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

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.

Figure 8: user interface panel

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.

Figure 9: code of add count

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.

Figure 10: code of delete count

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.

Figure 11: code of add balance

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. 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.

Figure 12: code of reduce balance

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

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.

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.

Figure 13: code of change account

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.

Figure 15: code of show recording

exit

3 Test result

4 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. 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）
Pre
accountNode
Next
accountNode

说明

1.去除可能自引文献相似度=辅助排除本人已发表文献后，送检文献中相似字符数/送检文献总字符数

2.去除参考文献相似度=排除参考文献后，送检文献中相似字符数/送检文献总字符数

3.总文献相似度=送检文献中相似字符数/送检文献总字符数

4.单篇最大相似度:送检文献与某-文献的相似度高于全部其他文献

5.检测字符数:送检文献检测部分的总字符数，不包括关键词、目录、图片、表格、附录、参考文献等

6.是否引用：该相似文献是否被送检文献标注为其参考文献引用

7.红色文字表示相似；绿色文字表示自引；黄色表示引用他人；灰色文字代表不参与检测

北京大雅因和信息技术有限公司
相似度检测专用章