VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590 018, KARNATAKA

#### vtulogo

*A Mini Project Report on*

“Implementing general ledger problem using consequential processing”

Submitted in the partial fulfillment for the requirements for the FS Lab with Mini Project (18ISL67)

in

#### **INFORMATION SCIENCE AND ENGINEERING**

#### By

**Mr. Nithin Urala M R USN: 1BY18IS076**

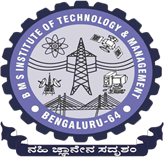
**Mr. Pranav R Deshkulkarni USN: 1BY18IS087**

Under the guidance of

**Mrs. Shanthi D L**

Assistant Professor

Department of ISE, BMSIT&M.

****

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

# BMS INSTITUTE OF TECHNOLOGY & MANAGEMNT

**YELAHANKA, BENGALURU-560064**

**2018-2022**

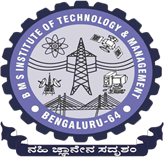
**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**BELAGAVI – 590 018, KARNATAKA**

**BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT**

**YELAHANKA, BENGALURU-560064**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

****

**CERTIFICATE**

This is to certify that the Project work entitled **“**Implementing general ledger problem using consequential processing**”** is a bonafide work carried out by **Mr. Nithin Urala M R (1BY18IS076)**, **Mr. Pranav R Deshkulkarni (1BY18IS087),** in partial fulfillment of File structures Lab with Mini Project (18ISL67) for the award of **Bachelor of Engineering Degree in Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report. The project report has been approved as it satisfies the academic requirements in respect of Mini Project work for the B.EDegree.

**Signature of the Guide Signature of the HOD**

Mrs. Shanthi D L Dr. Pushpa S.K

Assistant Professor Professor and Head

Department of ISE Department of ISE

**EXTERNAL EXAMINERS**

Name of the Examiners Signature with Date

1.

2.

**ACKNOWLEDGEMENT**

We are happy to present this mini project after completing it successfully. This mini project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this mini project a success.

We heartily thank our **Principal, Dr. Mohan Babu G.N,** **B M S Institute of Technology & Management** for his constant encouragement and inspiration in taking up this mini project.

We heartily thank our **Head of Department** Dr. Pushpa S.K**, Dept. of Information Science** **and** **Engineering, B M S Institute of Technology& Management** for his constant encouragement and inspiration in taking up this mini project.

We gracefully thank our Project guide**, Mrs. Shanthi D L, Asst. Professor, Dept. of** **Information Science and Engineering,** for her encouragement and advice throughout the course of the mini project work.

Special thanks to all the staff members of Information Science Department for their help and kind co-operation.

We also thank our parents and friends for their unconditional love and encouragement and support given to us in order to finish this precious work.

Last but not the least we would like to thank God for giving us the strength and motivation through the course of this Project.

By,

Nithin Urala M R

Pranav R Deshkulkarni

**ABSTRACT**

*A general ledger is the system employed by accountants to store and organize financial data used to create the firm's financial statements. Our project provides a solution for the general ledger problem using the concept of consequential processing. The system includes a journal file and a ledger file. The ledger file contains month-by-month summaries of the values as associated with each of the bookkeeping accounts. The journal file contains the monthly transactions that are ultimately to be posted to the ledger file. Once the journal file is complete for a given month, the journal must be posted. Posting involves associating each transaction with its account in the ledger. Our project aims to provide a system that could easily process the transactions of various accounts by maintaining a virtual ledger file that stores all transaction related information. To implement posting we have used the concept of consequential processing where we process the two lists sequentially in a parallel manner.*

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTERS** | **CONTENTS** | **Pg.no** |
| 1 | INTRODUCTION | 1 |
| 2 | REQUIREMENT SPECIFICATION | 2 |
| 3 | SYSTEM DESIGN | 3 |
| 4 | GENERAL LEDGER TABLE ENTRIES | 5 |
| 5 | IMPLEMENTATION | 8 |
| 6 | RESULT | 16 |
| 7 | CONCLUSION | 21 |
| 8 | REFERENCES | 22 |

**CHAPTER - 1**

**Introduction**

**1.1 Outline**

The project provides a solution for the general ledger problem using the concept of consequential processing. Consequential processing involves the coordinated processing of two or more sequential lists to produce a single output list. Consequential processing can be applied to problems that involve the performance of set operations on two or more sorted input files to produce one or more output files. Set operations can include union, intersection or more complex processes.

General ledger problem is based on accounting system. The system include a journal file and a ledger file :

* **Ledger file :** It contains month-by-month summaries of the values as associated with each of the bookkeeping accounts.
* **Journal file :** It contains the monthly transactions that are ultimately to be posted to the ledger file.

Once the journal file is complete for a given month, the journal must be posted to the ledger.

**1.2 Motivation and Scope**

Main motivation of the project is to implement the concept of cosequential processing to solve real time problem, that is, the general ledger problem. Since this problem involves both matching and merging concepts of cosequential processing it is one of the challenging problem which could be worked upon.

In our project we can view the contents of journal, ledger, add record to journal, post journal to ledger and get the ledger printout. The files are actually stored in the memory and the records within the files stored are of variable length where each field is separated using a separator.

**1.3 Problem Statement**

Design a general ledger posting program as part of an accounting system using cosequential processing.

**1.4 Limitations**

1. The system does not include a supporting front-end. Hence it will be hard if the number of accounts in the file increase. However the program prints the contents of the file in a well formatted way to overcome this limitation.
2. Once a record is added to the journal file, it is not possible to delete it within the program.

**CHAPTER - 2**

**Requirements Specification**

**2.1 System Requirements**

**2.3.1 Hardware Requirements**

* Intel i3 processor and above.
* Minimum of 128 MB on board memory.
* Windows 7/8/10 Operating System.
* VGA Monitor.

**2.3.2 Software Requirements**

* Requires JDK 8, Java SE.
* Text Editor such as Notepad, Sublime Text.

**2.2 Non-Functional Requirements**

* **Information Access:** Information can be accessed with ease and the file is stored as a text file in the memory which makes it easier to access.
* **Flexible**: It is flexible. New records can be added and viewed easily.
* **Reporting:** We will get detailed printout of ledger which is helpful for us as it shows details of all transactions pertaining to a particular account in a sorted order.
* Shows detailed information about each transaction of a particular account including check number, date of transaction, description and the credit/debit amount.
* Shows trial balance and new balance at the end of each account’s transactions.

**CHAPTER - 3**

**System Design**

View journal

View ledger

Get ledger printout

Post journal to ledger

Add record **to journal:**

Main Menu

User

**Fig 3.1 : System Design**

Following figure 3.1 shows the overall system design of the program. The system has five choices to choose from. They are :

* **View journal:** We can view the journal file which contains monthly transactions that are ultimately to be posted to the ledger file.
* **View ledger:** We can view the ledger file which contains month-by-month summaries of the values associated with each of the bookkeeping accounts.
* **Add record to journal:** In this option we can add new record to the journal file. The records within the journal file are stored using the concept of variable length record.
* **Post journal to ledger:** Once the record transactions are written in the journal, it is posted to the general ledger at the end of the month. This requires us to compare the account numbers of a record of journal with that of a ledger record and if a match is found then that particular record is posted into the ledger file.
* **Get ledger printout:**

1. Immediately after reading a new ledger object we need to print the header line and initialize the balance for the next month from the previous month's balance.
2. For each transaction object that matches, we need to update the account balance.
3. After the last transaction for the account, the balance line should be printed. This is the place, where a new ledger record could be written to create a new ledger file.

Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language. We are using Java as our programming language to code and to define different functional classes to perform different tasks. In the main function we are using the switch case to give choices to the user to perform their tasks of choice.

Few packages that we have used in our project are as follows :

* **Scanner :** Java Scanner class in Java is found in the java.util package. Java provides various ways to read input from the keyboard, the java.util.Scanner class is one of them.

The Java Scanner class breaks the input into tokens using a delimiter which is

white space by default. It provides many methods to read and parse various primitive

values.

The Java Scanner class is widely used to parse text for strings and primitive types

using a regular expression. It is the simplest way to get input in Java. By the help of

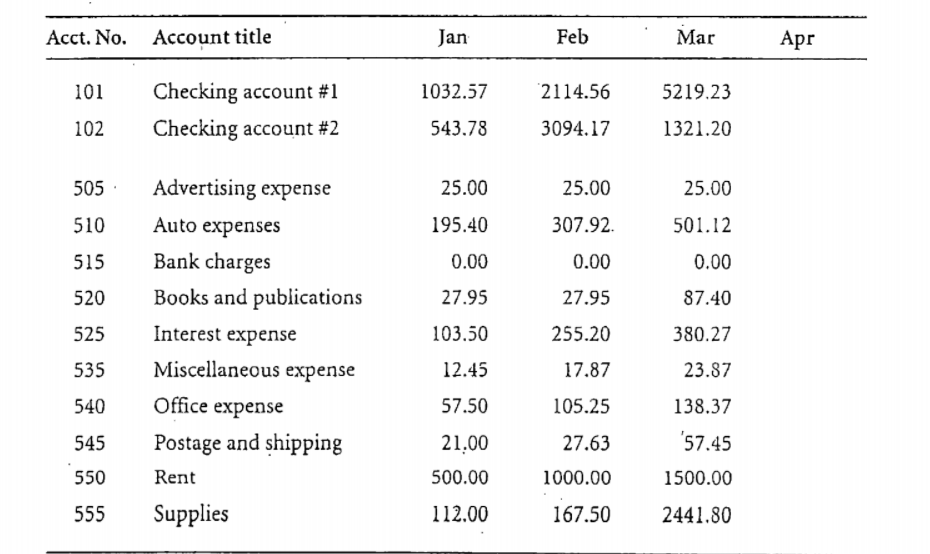
Scanner in Java, we can get input from the user in primitive types such as int, long, double, byte, float, short, etc.

* **BufferedReader :** Java BufferedReader is a class which simplifies reading text from a character input stream. It buffers the characters in order to enable efficient reading of text data.
* **PrintWriter :** Java PrintWriter class is the implementation of Writer class. It is used to print the formatted representation of objects to the text-output stream.
* **BufferedWriter :** BufferedWriter writes text to character output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings.

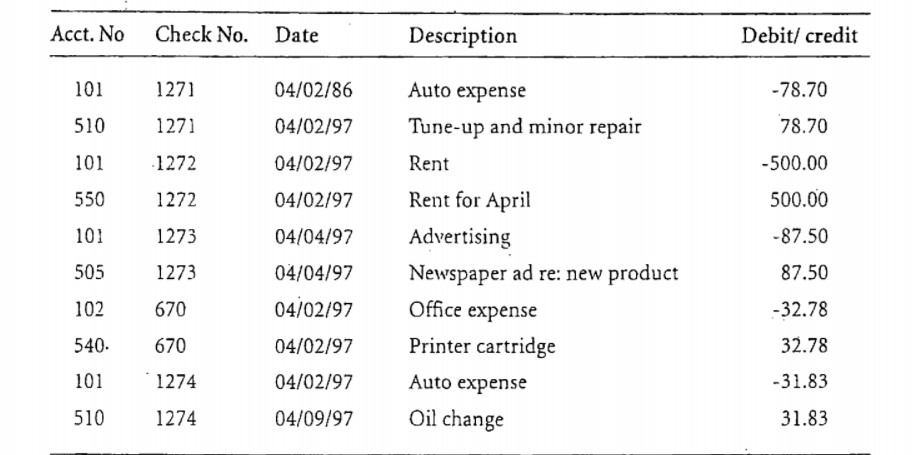
**CHAPTER - 4**

**GENERAL LEDGER TABLE ENTRIES**

**4.1 Journal and Ledger entries**

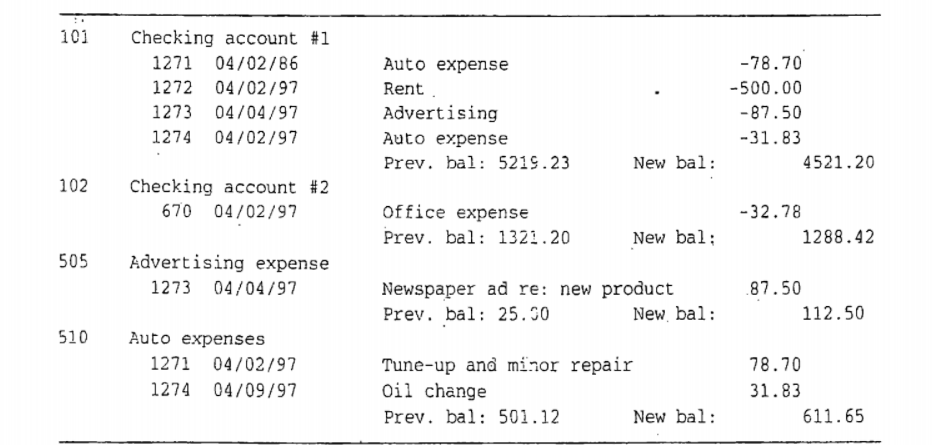
****

**Fig 4.1.1 : Sample Ledger Entries**

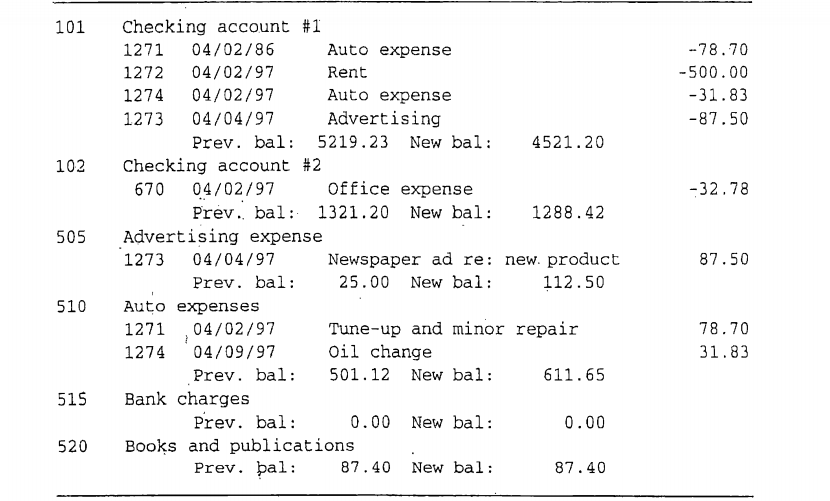


**Fig 4.1.2 : Sample Journal Entries**

Once the journal file is complete for a· given month, meaning that it contains all of the transactions for that month, the journal must be posted to the ledger. Posting involves associating each transaction with its- account. in the ledger. For example, the printed output produced for accounts 101, 102, 505, and 510 during the posting operation.



**Fig 4.1.3 : Sample ledger printout showing the effect of posting from the journal**



**Fig 4.1.4 : Sample Ledger Printout of all six accounts**

In summary, there are three different steps in processing the ledger entries:

1. Immediately after reading a new ledger object we need to print the header line and initialize the balance for the next month from the previous month's balance.
2. For each transaction object that matches, we need to update the account balance.
3. After the last transaction for the· account, the balance line should be printed. This is the place ,where a new ledger record could be written to create a new ledger file.

**CHAPTER - 5**

**IMPLEMENTATION**

**5.1 Java Code for implementation of General Ledger Problem**

import java.io.\*;

import java.util.\*;

class ledger {

public static void main(String[] args) throws IOException {

int choice;

Scanner scan = new Scanner(System.in);

while(true) {

System.out.println("----------------------------------------------");

System.out.println("FILE STRUCTURES LAB WITH MINI PROJECT(18ISL67)");

System.out.println("----------------------------------------------");

System.out.println("WELCOME TO GENERAL LEDGER PROGRAM");

System.out.println("----------------------------------------------");

System.out.println("1. View Journal");

System.out.println("2. View Ledger");

System.out.println("3. Add Record to journal");

System.out.println("4. Post Journal to Ledger");

System.out.println("5. Get Ledger Printout");

System.out.println("6. Exit");

System.out.println("----------------------------------------------");

System.out.println("ENTER YOUR CHOICE");

System.out.println("----------------------------------------------");

choice = scan.nextInt();

switch(choice) {

case 1 : System.out.println("View Journal Part");

display\_journal();

break;

case 2 : System.out.println("View Ledger Part");

display\_ledger();

break;

case 3 : System.out.println("Add record to Journal Part");

add\_record();

break;

case 4 : post();

System.out.println("Journal Posted to Ledger!");

break;

case 5 : System.out.println("--------------------------------------------------------------------");

System.out.println("\t\t\t\tLEDGER PRINTOUT");

print\_ledger();

break;

case 6 : System.exit(0);

}

}

}

//Method for displaying the journal

public static void display\_journal() throws IOException {

String acctno = "",check = "",date = "",desc = "",cred = "",s;

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("Acct. No\tCheck. No\tDate\t\tDescription\t\tCredit/Debit");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

BufferedReader b = new BufferedReader(new FileReader("journal.txt"));

while((s = b.readLine())!=null)

{

String result[] = s.split("\\|");

acctno = result[0];

check = result[1];

date = result[2];

desc = result[3];

cred = result[4];

System.out.println(acctno + "\t\t" + check + "\t\t" + date + "\t" + desc + "\t\t" + cred);

}

b.close();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

//Method for displaying the ledger

public static void display\_ledger() throws IOException{

String s, acctno = "", acctitle = "", jan = "", feb = "", mar = "", apr = "", may = "";

String jun = "", jul = "", aug = "", sep = "", oct = "", nov = "", dec = "";

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("Acct. No\tAccount Title\t\tJan\tFeb\tMar\tApr\tMay\tJun\tJul\tAug\tSep\tOct\tNov\tDec");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

BufferedReader b = new BufferedReader(new FileReader("ledger.txt"));

while((s = b.readLine())!=null)

{

String result[] = s.split("\\|");

acctno = result[0];

acctitle = result[1];

jan = result[2];

feb = result[3];

mar = result[4];

apr = result[5];

may = result[6];

jun = result[7];

jul = result[8];

aug = result[9];

sep = result[10];

oct = result[11];

nov = result[12];

dec = result[13];

System.out.println(acctno + "\t\t" + acctitle + "\t\t" + jan + "\t" + feb + "\t" + mar + "\t" + apr + "\t" + may + "\t" + jun + "\t" + jul + "\t" + aug + "\t" + sep + "\t" + oct + "\t" + nov + "\t" + dec);

}

b.close();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

//Method for adding record into the journal

public static void add\_record() throws IOException, FileNotFoundException{

String acctno,check,date,desc,cred;

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Details of the Transaction: ");

System.out.print("Account Number: ");

acctno = scan.nextLine();

System.out.print("Check Number: ");

check = scan.nextLine();

System.out.print("Date: ");

date = scan.nextLine();

System.out.print("Description: ");

desc = scan.nextLine();

System.out.print("Credit/Debit: ");

cred = scan.nextLine();

pack(acctno,check,date,desc,cred);

}

public static void pack(String acctno,String check,String date,String desc,String cred) throws IOException,FileNotFoundException{

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("journal.txt",true)));

String b = acctno + "|" + check + "|" + date + "|" + desc + "|" + cred + "|";

pw.println(b);

pw.flush();

pw.close();

}

//Method for posting the journal file to the ledger file

public static void post() throws IOException{

//get sorted journal and ledger list and create third list to store merged text.

sort\_journal();

String acctno = "",check = "",date = "",desc = "",cred = "",ledgerno = "",acctitle = "";

String olds = null;

BufferedReader br1 = new BufferedReader(new FileReader("journal.txt"));

BufferedReader br2 = new BufferedReader(new FileReader("ledger.txt"));

PrintWriter pw = new PrintWriter("finalledger.txt");

//if(ledgeritem1 == journalitem2) print(item2) movetonext(item2)

//if(ledgeritem1 > journalitem2) means no match for ledger. movetonext(journalitem2)

//if(ledgeritem1 < journalitem2) means item1's account details are done. movetonext(ledgeritem1)

String s = br1.readLine();

String t = br2.readLine();

while((s!=null)||(t!=null)) {

String result1[] = s.split("\\|");

String result2[] = t.split("\\|");

acctno = result1[0];

check = result1[1];

date = result1[2];

desc = result1[3];

cred = result1[4];

ledgerno = result2[0];

acctitle = result2[1];

if(ledgerno == acctno) {

String b = acctno + "|" + acctitle + "|" + check + "|" + date + "|" + desc + "|" + cred + "|";

pw.println(b);

s = br1.readLine();

}

else if(Integer.parseInt(ledgerno) < Integer.parseInt(acctno)) {

if((olds == ledgerno) || (olds!=null)) {

t = br2.readLine();

}

else {

String a = acctno + "|" + acctitle + "|0|0|0|0|";

pw.println(a);

t = br2.readLine();

}

}

else { //ledgerno > acctno

String b = acctno + "|" + acctitle + "|" + check + "|" + date + "|" + desc + "|" + cred + "|";

pw.println(b);

s = br1.readLine();

}

if(s == null) {

while(t!=null) {

t = br2.readLine();

if(t == null) {

break;

}

String result3[] = t.split("\\|");

ledgerno = result3[0];

acctitle = result3[1];

String a = ledgerno + "|" + acctitle + "|0|0|0|0|";

pw.println(a);

}

}

olds = acctno;

}

pw.flush();

pw.close();

br1.close();

br2.close();

}

//Method for sorting the Journal

public static void sort\_journal() throws IOException {

BufferedReader br = new BufferedReader(new FileReader("journal.txt"));

ArrayList<String> str = new ArrayList<>();

String line = "";

while((line = br.readLine())!=null){

str.add(line);

}

br.close();

Collections.sort(str);

FileWriter writer = new FileWriter("journal.txt");

for(String s: str){

writer.write(s);

writer.write("\n");

}

writer.close();

}

//Method for getting Ledger Printout

public static void print\_ledger() throws IOException {

String acctno = "",acctitle = "",check = "",date = "",desc = "",cred = "";

int old = 0,prevbal = 0,newbal = 0,count = 0 ;

BufferedReader br = new BufferedReader(new FileReader("finalledger.txt"));

String s,l;

BufferedReader br1 = new BufferedReader(new FileReader("ledger.txt"));

while((s=br.readLine())!=null) {

String result[] = s.split("\\|");

acctno = result[0];

acctitle = result[1];

check = result[2];

date = result[3];

desc = result[4];

cred = result[5];

if(Integer.parseInt(acctno) > old) {

if(old!=0) {

l = br1.readLine();

String result1[] = l.split("\\|");

int i = 2;

while(i<14) {

prevbal += Integer.parseInt(result1[i]);

i++;

}

newbal = prevbal + count;

System.out.println("\n\t\t\t\tPrev Balance : " + prevbal + "\tNew Balance : " + newbal);

System.out.println("--------------------------------------------------------------------------");

prevbal = 0;

newbal = 0;

}

else

System.out.println("--------------------------------------------------------------------------");

System.out.println(acctno + "\t" + acctitle);

System.out.println("--------------------------------------------------------------------------");

System.out.println("\t\t\t" + check + "\t" + date + "\t" + desc + "\t" + cred);

count = Integer.parseInt(cred);

}

else {

System.out.println("\t\t\t" + check + "\t" + date + "\t" + desc + "\t" + cred);

count += Integer.parseInt(cred);

}

old = Integer.parseInt(acctno);

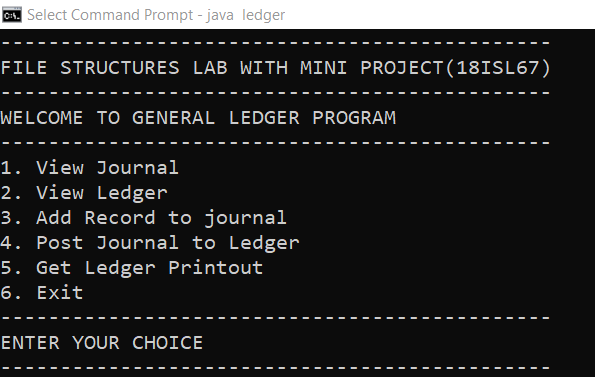
}

}

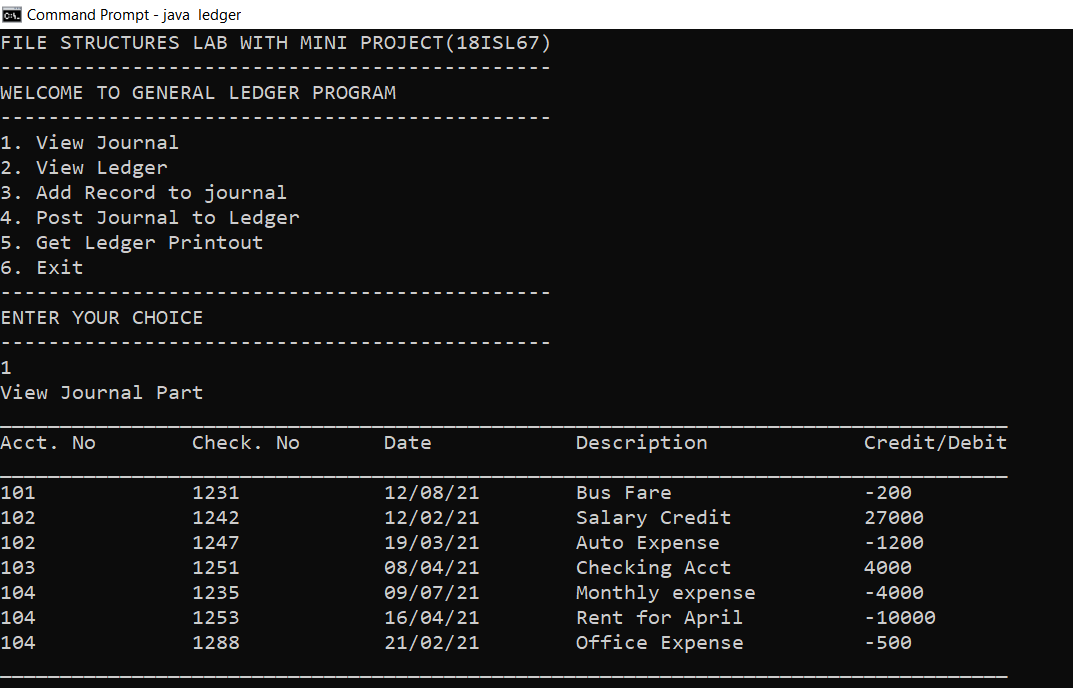
}

**CHAPTER - 6**

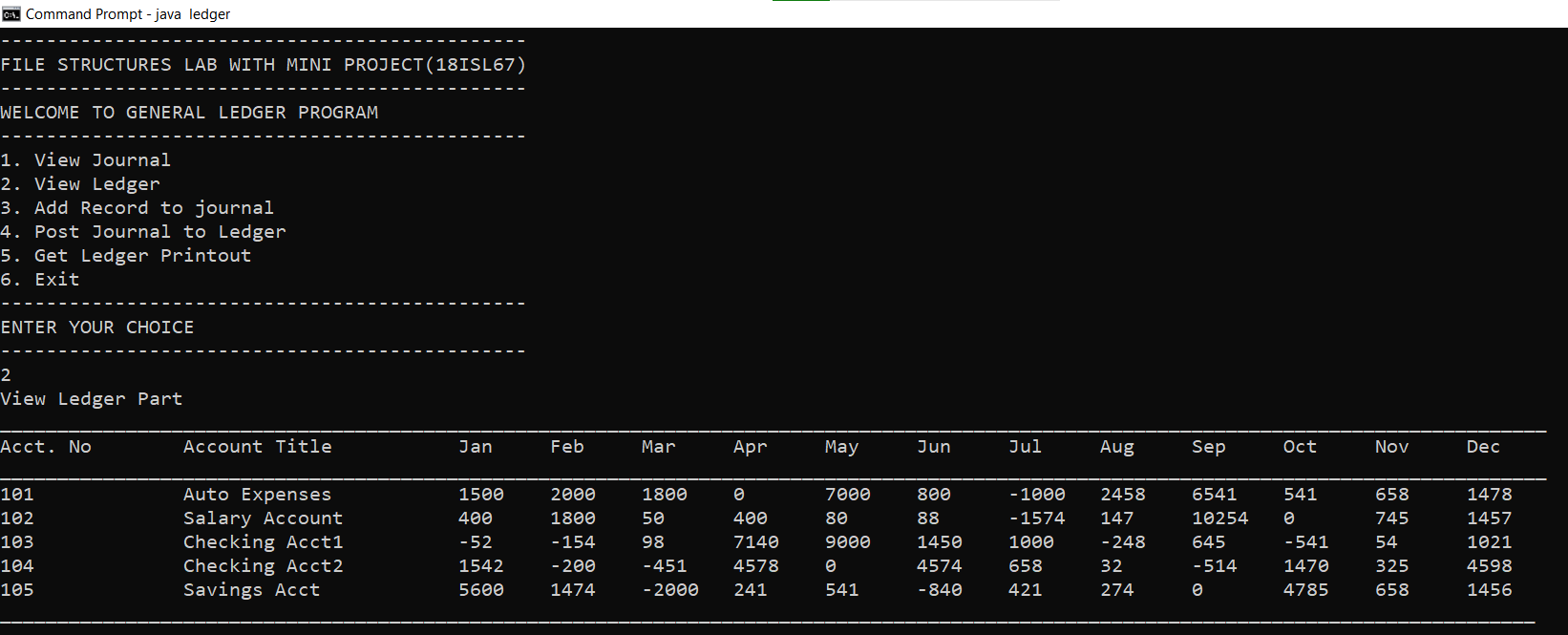
**RESULTS**

****

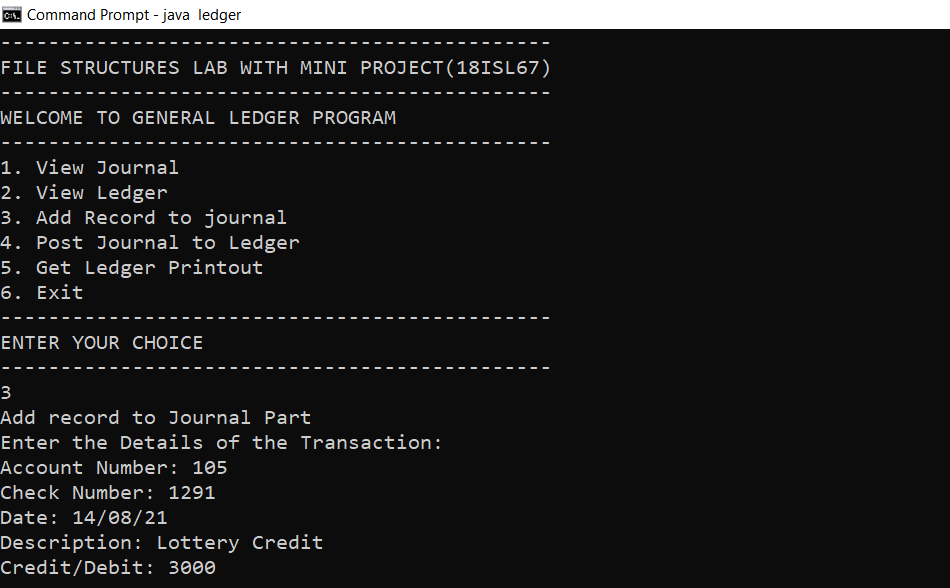
**Fig 6.1 : Menu for the General Ledger Program**

****

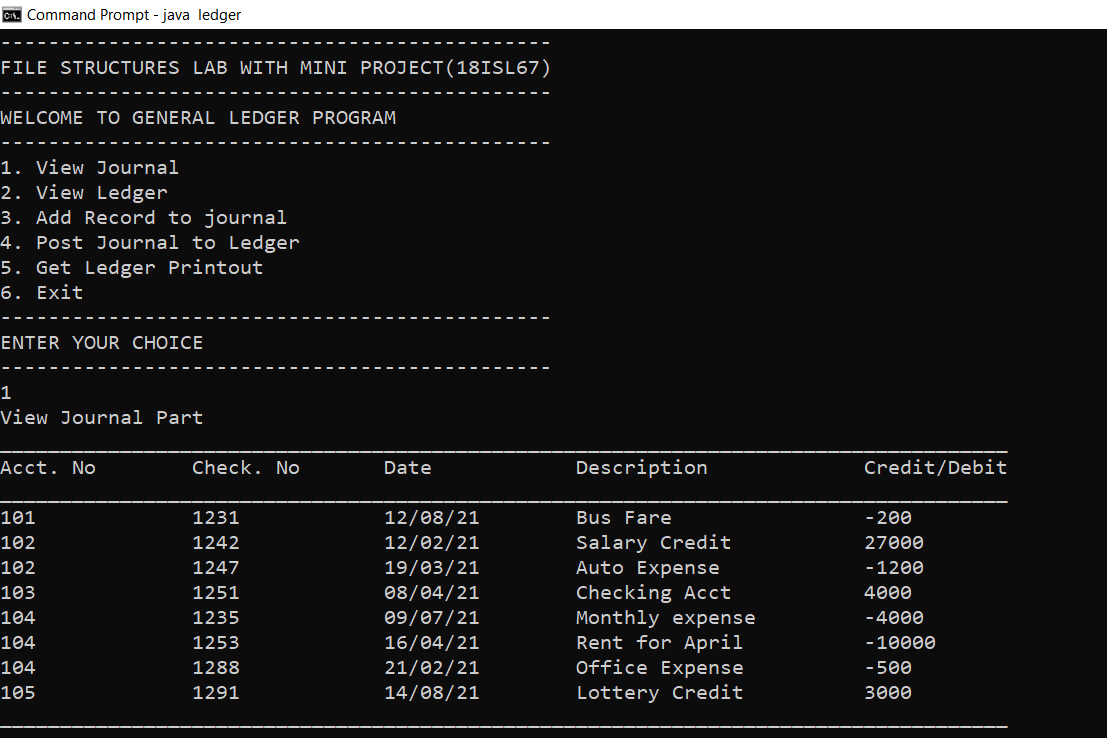
**Fig 6.2 : View Journal Option**

****

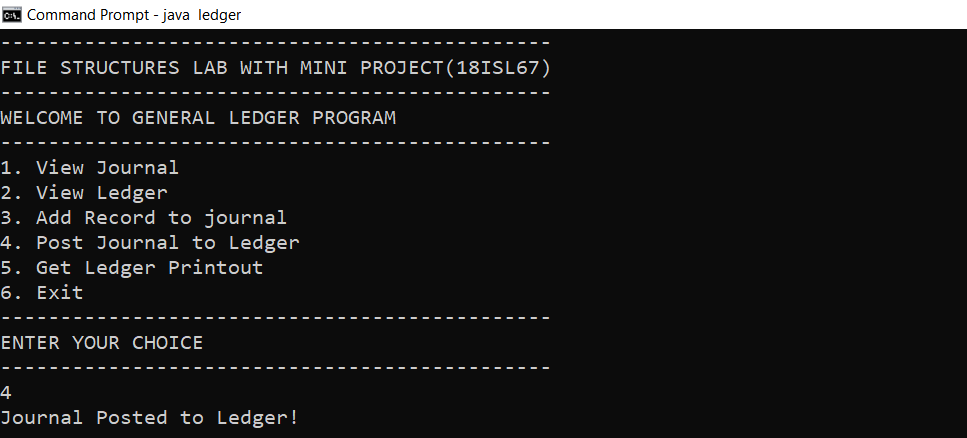
**Fig 6.3 : View Ledger Option**

****

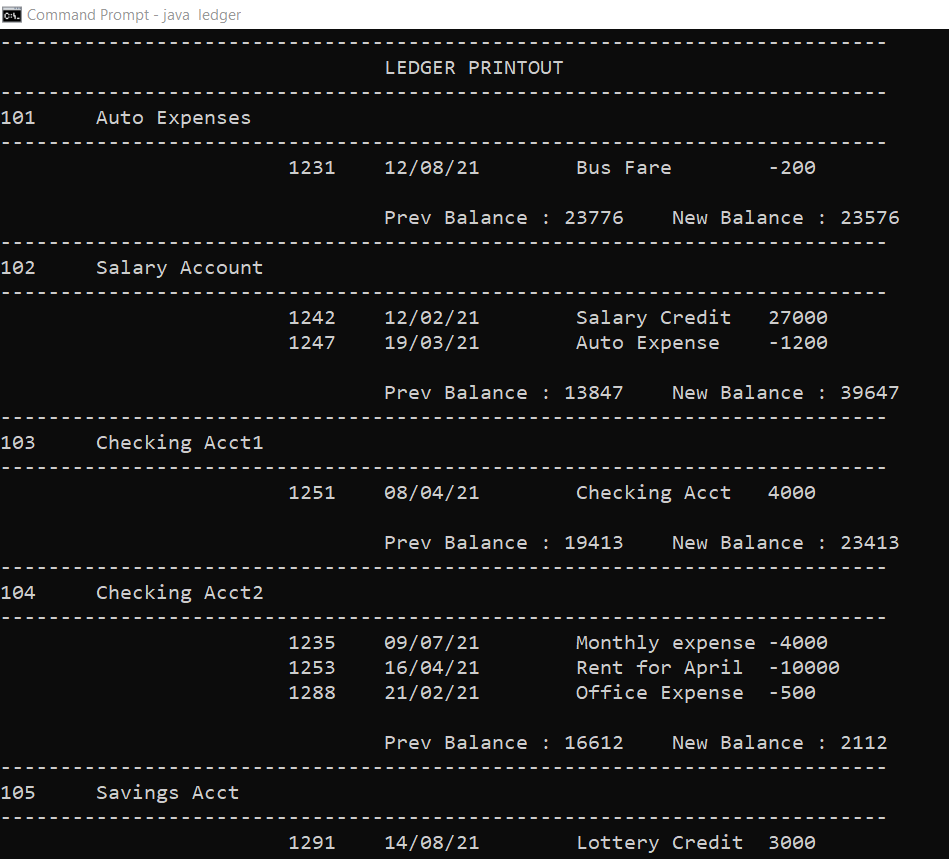
**Fig 6.4 : Add Record to Journal Option**

****

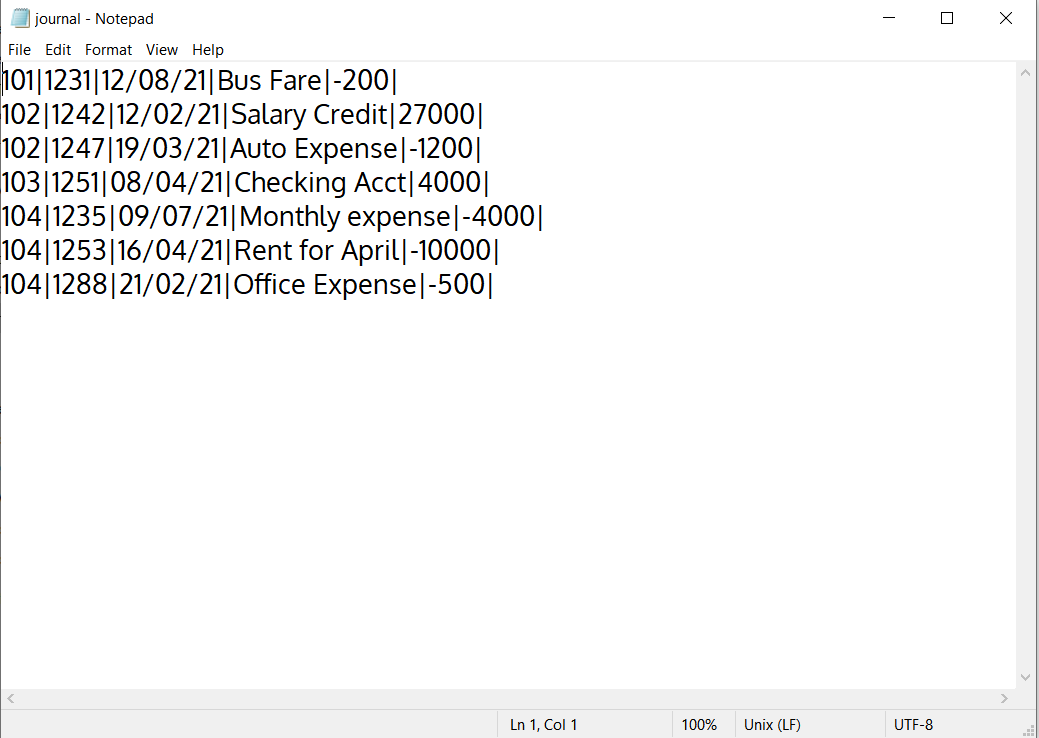
**Fig 6.5 : Updated Journal once new record is added**

****

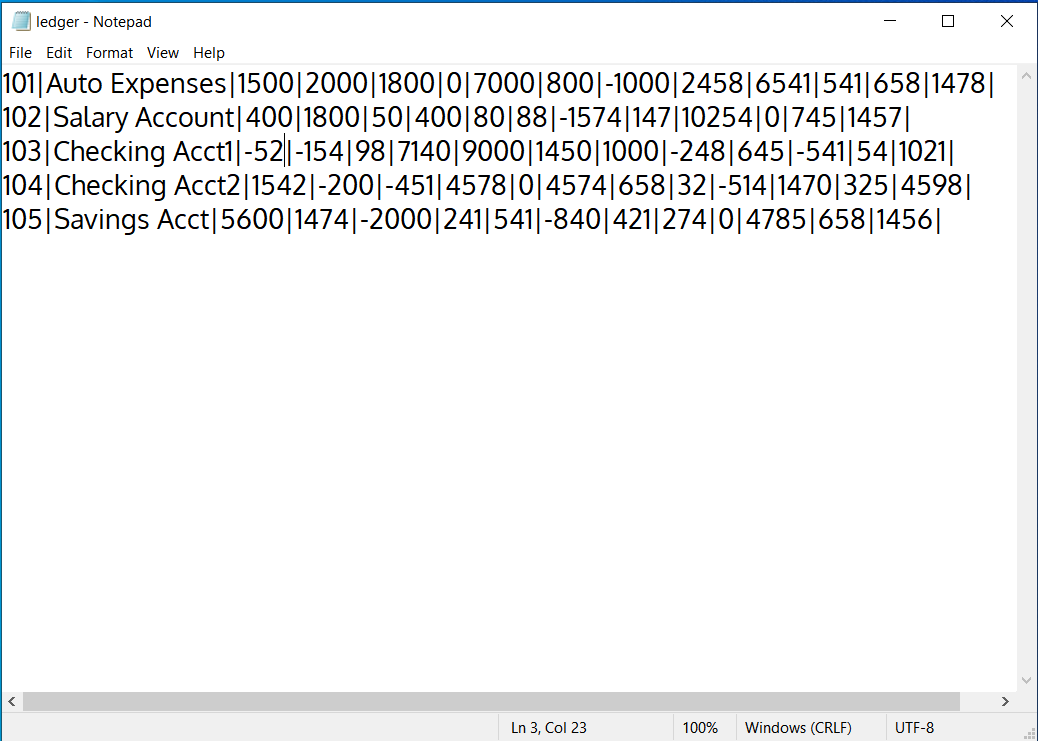
**Fig 6.6 : Post Journal to Ledger Option**

****

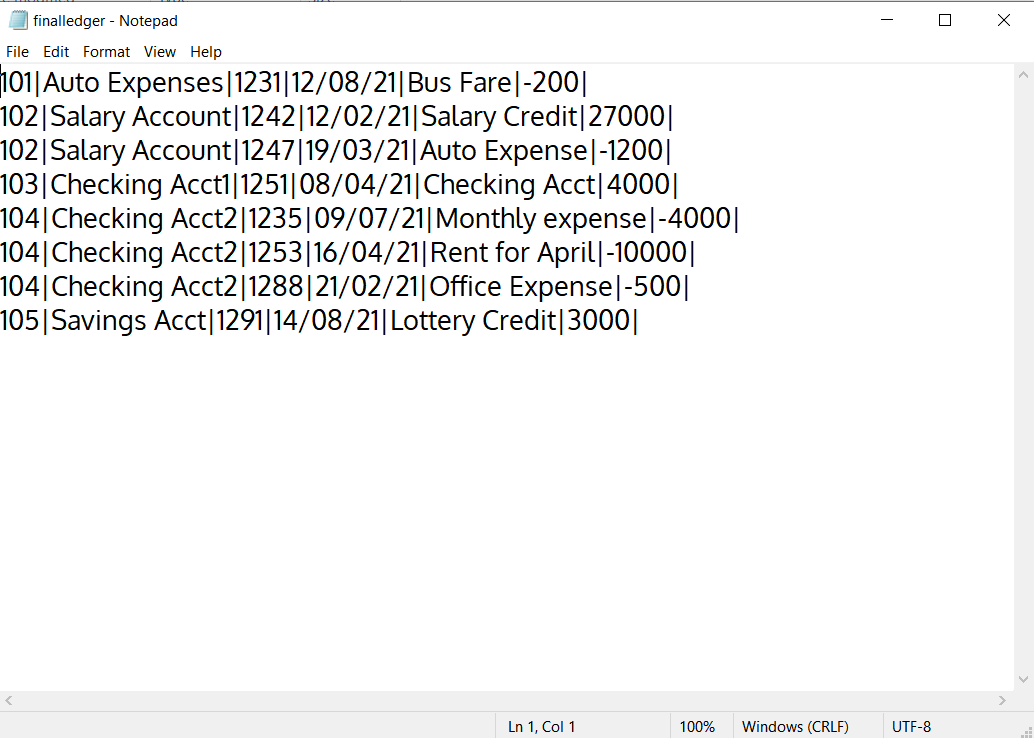
**Fig 6.7 : Ledger Printout Option**

****

**Fig 6.8 : Contents of journal.txt**

****

**Fig 6.9 : Contents of ledger.txt**

****

**Fig 6.10 : Contents of finalledger.txt**

**CHAPTER - 7**

**CONCLUSION**

We have used consequential processing in our project as it can be applied to large files and since the ledger file is a large file this technique is preferable. This technique is preferred when operating with two or more sequential lists. As this problem includes two files namely, the ledger file and the journal file, consequential processing is used. Another advantage of using this technique is that it requires less seeks as the files are sequential. The project can be replaced with the traditional ledger to bring digitalization and make account handling easier and faster.

**CHAPTER - 8**

**REFERENCES**

1. Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3rd Edition, Pearson Education, 1998.
2. https://www.site.uottawa.ca/~lucia/courses/2131-01/notes/lect14.pdf