```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;
import java.io.FileWriter;
import java.io.PrintWriter;

public class ProcessCustomerTransactions {

	public static void main(String[] args) {

		List<File> all = new ArrayList<File>();

	    String pathname = System.getenv("$TRANSACTION_PROCESSING");
	    System.out.println(System.getenv("$TRANSACTION_PROCESSING"));
	    addTree(new File(pathname + "\\pending"), all);
	    Collections.sort(all);
	    Object[] filenames = all.toArray();
		try {
			for(int i = 0; i < filenames.length; i++){
				BufferedReader reader = new BufferedReader(new
FileReader((File)filenames[i]));
				List<String[]> entries = new ArrayList<String[]>();
				int skipped = 0;
				double credits = 0;
				double debits = 0;
				String line = "";
				String[] temp = new String[2];
				List<Long> accountNums = new ArrayList<Long>();
				while ((line = reader.readLine()) != null)
			{
					int low = 0;
					int high = accountNums.size() - 1;
					temp = line.split(",\\s|,");
					long clientNumber = -1;
					try{
						clientNumber = Long.parseLong(temp[0]);
					}catch(NumberFormatException e){
						skipped++;
						continue;
					}
					entries.add(temp);
					if(accountNums.size() == 0){
						accountNums.add(clientNumber);
						continue;
					}
					boolean found = false;
					while(high >= low){
						int middle = (low + high) / 2;
						if(accountNums.get(middle) == clientNumber){
							found = true;
```

```
                                        break;
                                    }
                                    if(accountNums.get(middle) < clientNumber)
                                        low = middle + 1;
                                    if(accountNums.get(middle) > clientNumber)
                                        high = middle - 1;
                                }
                                if(!found){
                                    if(high == -1)
                                        high = 0;
                                    Long tempLong = accountNums.get(high);
                                    if(tempLong.compareTo(clientNumber) < 0){
                                        if(clientNumber != accountNums.size()-
1)
                                            accountNums.add(high + 1,
clientNumber);
                                        else
                                            accountNums.add(clientNumber);
                                    }
                                    else{
                                        accountNums.add(high, clientNumber);
                                    }
                                }
                            }
                            for(int n = 0; n < entries.size(); n++){
                                double tempFunds =
Double.parseDouble(entries.get(n)[1]);
                                if(tempFunds >= 0){
                                    credits += tempFunds;
                                }
                                else{ //Double.parseDouble(entries.get(n)[1]) < 0
                                    debits -= tempFunds;
                                }
                            }

                            reader.close();
                            ((File)filenames[i]).renameTo(new File(pathname +
"\\processed\\" + ((File)filenames[i]).getName()));
                            FileWriter write = new FileWriter(new File(pathname +
"\\reports\\" +
                                    ((File)filenames[i]).getName().substring(0,
((File)filenames[i]).getName().length()-4) + ".txt"), false);
                            PrintWriter pw = new PrintWriter(write);
                            pw.println("File Processed: " +
((File)filenames[i]).getName());
                            pw.println("Total Accounts: " + accountNums.size());
                            pw.println("Total Credits: $" + credits);
                            pw.println("Total Debits: $" + debits);
                            pw.println("Skipped Transactions: " + skipped);
                            pw.close();
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
            }
```

```
	static void addTree(File file, Collection<File> all) { //Builds a list of
files of the given directory
	    File[] children = file.listFiles();
	    if (children != null) {
	        for (File child : children) {
	            all.add(child);
	            addTree(child, all);
	        }
	    }
	}
}
```