**Main.java**

```
/**Solomiya Pobutska
* Assignment #1
* CISC 3130 Spring 2020
*
*
*
* TechStore Company asked for an Accounts Receivable department
* report with customers data of orders, payments, previous and due balances
* */

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class Main {

  public static void main(String[] args) {
      ArrayList<Customer> customers = new ArrayList<>();

      try {
          File mfile = new File("MasterFile.txt");//reading from masterfile.txt stored in ArrayList
          try (FileReader fileReader = new FileReader(mfile)) {
              BufferedReader bufferedReader = new BufferedReader(fileReader);
              String line;
              String regex = "(\\s)+";
              while ((line = bufferedReader.readLine()) != null) {
                  String[] columns = line.trim().split(regex);// splits  each line
                  String cname = columns[1].trim();
                  int cnum = Integer.parseInt(columns[0].trim());
                  double balancedue = Double.parseDouble(columns[2].trim());
                  Customer customer = new Customer(cname , cnum , balancedue);
                  customers.add(customer);

                  //for every added customer reading its related transactions
                  File tfile = new File("Transactions.txt");
                  FileReader tfileReader = new FileReader(tfile);
                  BufferedReader tbufferedReader = new BufferedReader(tfileReader);
                  String tline;
```

```
            //loop till end of transaction file
            while ((tline = tbufferedReader.readLine()) != null) {
                String[] splited = tline.trim().split(regex);// splitting each line
                char code = splited[0].trim().toUpperCase().charAt(0);
                int customerid = Integer.parseInt(splited[1].trim());
                if (customerid != customer.getCnum())
                    continue;
                //using switch case for order or payment
                switch (code) {
                    case 'P': {
                        int transnum = Integer.parseInt(splited[2].trim());
                        double amount = Double.parseDouble(splited[3].trim());
                        Transaction trans = new Payment(transnum , amount);
                        customer.addTranscation(trans);
                        break;
                    }

                    case 'O': {

                        int transnum = Integer.parseInt(splited[2].trim());
                        String item = splited[3].trim();
                        int quantity = Integer.parseInt(splited[4].trim());
                        double cost = Double.parseDouble(splited[5].trim());

                        Transaction trans = new Order(transnum , item , quantity , cost);
                        customer.addTranscation(trans);

                        break;
                    }

                }
            }

            tbufferedReader.close();
        }
    }
} catch (IOException e) {// EXCEPTION !!!
    e.printStackTrace();
}

 //Creating and writing into the file
    try {
            FileWriter fstream = new FileWriter("output.txt");
            BufferedWriter out = new BufferedWriter(fstream);

            for (Customer cust : customers) {
                out.write(cust.Print());
            }
            out.close();

        }
        catch (Exception e) { // Catching exception if any
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

**Transaction.java**

```
/**
* Transaction is an abstract class which gets transaction number
* information of the customer
* */

public abstract class Transaction {
    int transactionnumber; //transaction number

    //constructor
    public Transaction(int transactionnumber) {
        super();
        this.transactionnumber = transactionnumber;
    }

    //getter for transaction number
    public int getTransactionnumber() {
        return transactionnumber;
    }

    //tostring
    @Override
    public String toString() {
        return "TRANSACTION "+getTransactionnumber();
    }
}
```

**Order.java**

```
/**
* Order class extends Transaction and sets up all the Order information
* */

public class Order extends Transaction {
   private String item;
   private int quantity;
   private double itemcost;

   //Constructor using properties
   public Order(int transnum, String item, int quantity, double itemcost) {
      super(transnum);
      this.item = item;
      this.quantity = quantity;
      this.itemcost = itemcost;
   }

   //getters and for properties
   public double getItemcost() {
      return itemcost;
   }

   public double getTotalcost() {
      return itemcost*quantity;
   }

   public String getItem() {
      return item;
   }
   public int getQuantity() {
      return quantity;
   }

   //overriding tostring
   @Override
   public String toString() {
      return super.toString()+"\t"+getItem()+" ORDERED\t$"+getTotalcost()+"\n";
   }
}
```

**Payment.java**

```
/**
* Payment class extends Transaction and gets the payment information of the customer
* */

public class Payment extends Transaction {
   private double amount; // amount of payment transaction

   // constructor
   public Payment(int transnum, double amount) {
      super(transnum);
      this.amount = amount;
   }

   // getter for amount
   public double getAmount() {
      return amount;
   }

   // overriding toString
   @Override
   public String toString() {
      return super.toString() + "\tPAYMENT\t$" + getAmount() +"\n";
   }
}
```

**Customer.java**

```
/**
* Customer class includes all data about the customer;
* it also updates customer's balance information
* */

import java.util.ArrayList;

public class Customer {
   private String cname;
   private int cnum;
   private double balancedue;
   private ArrayList<Transaction> transactions;

   //constructor used while comparisons
   public Customer (int cnum) {
       this.cnum = cnum;
   }

   // constructor
   public Customer (String cname , int cnum , double balancedue) {
       super();
       this.cname = cname;
       this.cnum = cnum;
       this.balancedue = balancedue;
       this.transactions = new ArrayList<>();
   }

   //adding each transaction into arraylist
   public void addTranscation (Transaction trans) {
       transactions.add(trans);
   }

   //updating balance due with all transactions
   public void updateBalanceDue () {
       for (Transaction trans : transactions) {
           if (trans instanceof Order) { //checking if this is order
               Order order = (Order) trans;
               balancedue = balancedue + order.getTotalcost(); //adding due
           } else if (trans instanceof Payment) { //checking if this is payment
               Payment payment = (Payment) trans;
               balancedue = balancedue - payment.getAmount(); //deducing due
           }
       }
   }

   @Override
   public boolean equals(Object obj){
       if(obj != null && obj instanceof Customer){
           Customer cust = (Customer) obj;
           return getCnum()== cust.getCnum();
       }
       return false;
   }

   //getters for each property of customer
   public String getCname() {
       return cname;
   }
```

```
    public int getCnum() {
        return cnum;
    }

    public double getBalancedue() {
        return balancedue;
    }

    public ArrayList<Transaction> getTransactions() {
        return transactions;
    }

    //Overriding tostring
    @Override
    public String toString() {
        return "Customer [cname=" + cname + ", cnum=" + cnum + ", balancedue=" + balancedue +
"]";
    }

    //prints balance sheet
    public String Print() {
        String output = "";
        output = output + getCname()+"\t"+getCnum()+"\n\n\t\t\t"+"PREVIOUS BALANCE
\t$"+getBalancedue()+"\n\n";

        for(Transaction trans:transactions){
            output = output+trans.toString();
        }
        updateBalanceDue();
        output = output + "\n\n\t\t\t"+"BALANCE DUE\t$"+getBalancedue()+"\n";
        return output;
    }
}
```

**MasterFile.txt**

```
1000 Victor 100.00
1001 Veronica 10.00
1002 Sam 250.00
1003 Adam 600.00
1004 Nick 50.00
```

**Transactions.txt**

```
P   1000   1234   50.00
O   1000   2345   Laptop   1   999.00
P   1000   3456   12.50
O   1000   4567   CPU      1   200.00
P   1000   5678   1000.00


O   1001   5671   Laptop   1   1500.00
P   1001   6789   260.00
O   1001   7890   Mice     1   35.99
P   1001   8910   4.00
O   1001   9101   Pens     1   5.99


P   1002   2123   50.00
P   1002   5654   100.00
O   1002   5467   Router   1   90.00
O   1002   8985   Case     1   39.99
P   1002   2439   150.00


P   1003   2039   200.00
O   1003   3948   Screen   1   300.00
O   1003   4920   Wires    2   50.00
P   1003   5839   25.00
O   1003   6868   Light    1   75.00


P   1004   5545   30.00
P   1004   5963   10.00
O   1004   2331   PowerUnit   1  100.00
P   1004   3212   60.00
P   1004   9532   10.00
```

OUTPUT:

```
Victor          1000
                      PREVIOUS BALANCE    $100.00

TRANSACTION   1234        PAYMENT   $50.00
TRANSACTION   2345        Laptop   $999.00
TRANSACTION   3456        PAYMENT   $12.50
TRANSACTION   4567        CPU   $200.00
TRANSACTION   5678        PAYMENT   $1000.00

                      BALANCE DUE    $236.00



Veronica         1001
                      PREVIOUS BALANCE      $10.00

TRANSACTION   5671       Laptop  $1500.00
TRANSACTION   6789       PAYMENT   $260.00
TRANSACTION   7890       Mice   $35.99
TRANSACTION   8910       PAYMENT$   $4.00
TRANSACTION   9101       Pens  $5.99

                      BALANCE DUE    $1287.98



Sam          1002
                      PREVIOUS BALANCE      $250.00

TRANSACTION   2123       PAYMENT    $50.00
TRANSACTION   5654       PAYMENT   $100.00
TRANSACTION   5467       Router   $90.00
TRANSACTION   8985       Case   $39.99
TRANSACTION   2439       PAYMENT   $150.00

                      BALANCE DUE    $79.99



Adam          1003
                      PREVIOUS BALANCE      $600.00
```

```
TRANSACTION   2039       PAYMENT    $200.00
TRANSACTION   3948       Screen    $300.00
TRANSACTION   4920       Wires   $50.00
TRANSACTION   5839       PAYMENT   $25.00
TRANSACTION   6868       Light   $75.00

                         BALANCE DUE    $800.00



Nick          1004
                         PREVIOUS BALANCE      $50.00

TRANSACTION   5545       PAYMENT    $30.00
TRANSACTION   5963       PAYMENT    $10.00
TRANSACTION   2331       PowerUnit    $100.00
TRANSACTION   3212       PAYMENT   $60.00
TRANSACTION   9510       PAYMENT   $10.00

                         BALANCE DUE    $40.00
```