DreamCandies File Tool
Assignment Report

Mylonas Efstratios

Contents

Introduction

As part of my assignment, I developed a tool for DreamCandies company, that gets the extraction files of the company and by using a sample file, exports the subset of the full extraction files only including the data for the customers specified.

I am going to explain how I built the program, how can it be executed and what is the output.

All the implementation is in Java.

1. System Basis

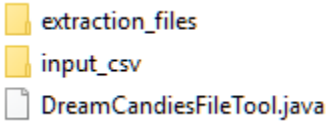
The Operating System I used to make the project is Windows 10.

The programming language used in this project is Java. There is the following version of Java used.

```
C:\Users\Stratos\Java Projects\DreamCandies File Tool>java -version
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

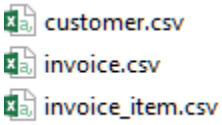
2. Tool Files

Here is the main folder of the tool. Each file format is presented in the tool's functional specification pdf file.



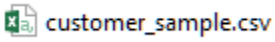
2.1 Extraction Files

The extraction_files folder contains the three files provided by the DreamCandies. They are all in csv format, with ASCII encoding.



2.2 Input CSV File

The input_csv folder contains the customer_sample.csv file that specifies customer codes we want to search and get the subset of the three extraction files.



2.3 DreamCandies File Tool File

The DreamCandiesFileTool.java is eventually the whole code that the tool used to manipulate the extraction files and create the output.

3. Main Code

The Tool contains of four classes:

- Customer
- Invoice_Item
- Invoice
- DreamCandiesFileTool (Main)

3.1 Customer class

The Customer class is a class created to keep a customer's code, as well as his first and last name. There are methods that initialize one Customer object, set a new one, and get some private data used later. Also there is a print option to see one customer's data.

```
class Customer
{
    private char[] customer_code;
    private char[] firstname;
    private char[] lastname;

    public Customer()
    {
        this.customer_code = new char[30];
        this.firstname = new char[30];
        this.lastname = new char[30];
    }

    public void setCustomer(String code, String name, String surname)
    {
        char[] c1 = code.replace("\"", "").toCharArray();
        char[] c2 = name.replace("\"", "").toCharArray();
        char[] c3 = surname.replace("\"", "").toCharArray();

        this.customer_code = c1;
        this.firstname = c2;
        this.lastname = c3;
    }

    public String getCode()
    {
        return new String(this.customer_code);
    }

    public String getFirstName()
    {
        return new String(this.firstname);
    }

    public String getLastName()
    {
        return new String(lastname);
    }

    public void printCustomer()
```

```
    {
        System.out.println(new String(this.customer_code) + " " + new
String(this.firstname) + " " + new String(this.lastname));
    }

}
```

3.2 Invoice_Item class

The Invoice_Item class is the class to store each invoice item's data to an object. There is a need to store the invoice code that the item is included, it's code, the amount of the items and its quantity. There are methods that initialize and set new objects of them, get some data needed later by the tool and, of course, printing option.

```
class Invoice_Item
{
      private char[] invoice_code;
      private char[] item_code;
      private float amount;
      private int quantity;

      public Invoice_Item()
      {
            invoice_code = new char[30];
            item_code = new char[100];
      }

      public void setInvoiceItem(String ccode, String icode, String am,
String qt)
      {
            char[] c1 = ccode.replace("\"", "").toCharArray();
            char[] c2 = icode.replace("\"", "").toCharArray();

            this.invoice_code = c1;
            this.item_code = c2;

            float f = Float.parseFloat(am.replace("\"", ""));
            this.amount = f;

            int i = Integer.parseInt(qt.replace("\"", ""));
            this.quantity = i;
      }

      public String getCode()
      {
            return new String(this.invoice_code);
      }

      public String getItemCode()
      {
            return new String(this.item_code);
      }

      public String getAmount()
      {
```

```
            return Float.toString(this.amount);
        }

        public String getQuantity()
        {
            return Integer.toString(this.quantity);
        }

        public void printInvoiceItem()
        {
            System.out.println(new String(this.invoice_code) + " " +
new String(this.item_code) + " " + Float.toString(amount) + " " +
Integer.toString(quantity));
        }
}
```

3.3 Invoice class

The Invoice class is the class that keeps the data of an invoice in an object. It keeps the customer as a Customer object, as the list of its items in a list (ArrayList of Invoice_Item). The class stores the the invoices' code, the amount money spent and the date it created. There are methods to initialize and set new class objects, get specific data needed and printing.

```
class Invoice
{
    private Customer customer;
    private char[] invoice_code;
    private ArrayList<Invoice_Item> invoice_items;
    private float amount;
    private Date date;

    public Invoice()
    {
        customer = new Customer();
        invoice_code = new char[30];
        invoice_items = new ArrayList<Invoice_Item>();
        this.date = new Date();
    }

    public void setInvoice(Customer cust, ArrayList<Invoice_Item>
inv_itm, String am, String dt)
    {
        this.customer = cust;
        this.invoice_items = inv_itm;

        char[] c1 = inv_itm.get(0).getCode().toCharArray();
        this.invoice_code = c1;

        float f = Float.parseFloat(am.replace("\"", ""));
        this.amount = f;

        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMM-
yyyy");
```

```
        try
        {
            Date d = dateFormat.parse(dt.replace("\"", ""));
            this.date = d;
        } catch (ParseException e)
        {
            e.printStackTrace();
        }
    }

    public String getCustomerCode()
    {
        return this.customer.getCode();
    }

    public String getInvoiceCode()
    {
        return new String(this.invoice_code);
    }

    public String getAmount()
    {
        return Float.toString(this.amount);
    }

    public String getDate()
    {
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMM-
yyyy");
        return dateFormat.format(date);
    }

    public Customer getCustomer()
    {
        return this.customer;
    }

    public ArrayList<Invoice_Item> getInvoiceItems()
    {
        return this.invoice_items;
    }

    public void printInvoice()
    {
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMM-
yyyy");
        System.out.println(customer.getCode() + " " +
this.invoice_items.get(0).getCode() + " " + Float.toString(amount) + "
" + dateFormat.format(date));
    }
}
```

3.4 DreamCandiesFileTool class

This is the main class of the tool. It has the main method that the tool runs to operate. It consists of the main algorithm that the tool gets the data from the files, it stores them in

ArrayLists, makes the search to subset the lists to new output lists, so that it creates the output files needed.

```
public class DreamCandiesFileTool{

	public static void main(String[] args)
	{
		//Initialise Lists
		ArrayList<Customer> customer_list = new
ArrayList<Customer>();
		ArrayList<Invoice> invoice_list = new ArrayList<Invoice>();
		ArrayList<Invoice_Item> invoice_item_list = new
ArrayList<Invoice_Item>();
		ArrayList<String> customer_sample_list = new
ArrayList<String>();

		//Set filepaths
		String CustomerCSV = "./extraction_files/customer.csv";
		String InvoiceCSV = "./extraction_files/invoice.csv";
		String InvoiceItemCSV =
"./extraction_files/invoice_item.csv";
		String CustomerSampleCSV =
"./input_csv/customer_sample.csv";

		//Set type of csv
		String line = "";
		String cvsSplitBy = ",";


		//Get data from the 3 files
		//1. Get customer list from file CustomerCSV
		try (BufferedReader br = new BufferedReader(new
FileReader(CustomerCSV)))
		{
			while ((line = br.readLine()) != null)
			{
				Customer customer = new Customer();
				String[] index = line.split(cvsSplitBy);
				customer.setCustomer(index[0], index[1],
index[2]);
				customer_list.add(customer);
			}

		} catch (IOException e)
		{
			e.printStackTrace();
		}

		customer_list.remove(0); //remove the header

		//2. Get invoice item list from file InvoiceItemCSV
		try (BufferedReader br = new BufferedReader(new
FileReader(InvoiceItemCSV)))
		{
			int i=0;
			while ((line = br.readLine()) != null)
```

```
                {
                        if (i == 0) //get rid of header
                        {
                                i++;
                                continue;
                        }
                        Invoice_Item invoice_item = new Invoice_Item();
                        String[] index = line.split(cvsSplitBy);
                        invoice_item.setInvoiceItem(index[0], index[1],
index[2], index[3]);
                        invoice_item_list.add(invoice_item);
                }

            } catch (IOException e)
            {
                e.printStackTrace();
            }

            //3. Get invoice list from file InvoiceCSV
            try (BufferedReader br = new BufferedReader(new
FileReader(InvoiceCSV)))
            {
                int first=1;
                while ((line = br.readLine()) != null)
                {
                        if (first == 1) //get rid of header
                        {
                                first = 0;
                                continue;
                        }
                        Invoice invoice = new Invoice();
                        String[] index = line.split(cvsSplitBy);

                        //search and set corresponding customer
                        Customer cust = new Customer();
                        for (int i=0; i<customer_list.size(); i++)
                        {
                                if
(customer_list.get(i).getCode().equals(index[0].replace("\"", "")))
                                {
                                        cust = customer_list.get(i);
                                }
                        }

                        //search and set corresponding invoice items
list
                        ArrayList<Invoice_Item> inv_itm = new
ArrayList<Invoice_Item>();
                        for (int i=0; i<invoice_item_list.size(); i++)
                        {
                                if
(invoice_item_list.get(i).getCode().equals(index[1].replace("\"", "")))
                                {

      inv_itm.add(invoice_item_list.get(i));
                                }
                        }
```

```
                        invoice.setInvoice(cust, inv_itm, index[2],
index[3]);
                        invoice_list.add(invoice);
                  }

            } catch (IOException e)
            {
                  e.printStackTrace();
            }



            //Get customer sample customers list from file
CustomerSampleCSV
            try (BufferedReader br = new BufferedReader(new
FileReader(CustomerSampleCSV)))
            {
                  int i=0;
                  while ((line = br.readLine()) != null)
                  {
                        if (i == 0)
                        {
                              i++;
                              continue;
                        }
                        String customer_sample;
                        customer_sample = line.replace("\"", "");
                        customer_sample_list.add(customer_sample);
                  }

            } catch (IOException e)
            {
                  e.printStackTrace();
            }

            //Make output empty lists for the output of the tool
            ArrayList<Customer> output_customer_list = new
ArrayList<Customer>();
            ArrayList<Invoice> output_invoice_list = new
ArrayList<Invoice>();
            ArrayList<Invoice_Item> output_invoice_item_list = new
ArrayList<Invoice_Item>();

            //Fill the lists by searching the customers we've got in
the customer sample file
            //Once we found the customer we get the rest of data in
each list
            for (int i=0; i<customer_sample_list.size(); i++)
            {
                  String search = customer_sample_list.get(i); //set
the customer we want to search
                  for (int j=0; j<invoice_list.size(); j++)
                  {
                        if
(search.equals(invoice_list.get(j).getCustomerCode()))
                        {
```

```
      output_invoice_list.add(invoice_list.get(j));
                              int found=0;
                              for (int k=0;
k<output_customer_list.size(); k++) //check for duplicates
                              {
                                    if
(invoice_list.get(j).getCustomerCode().equals(output_customer_list.get(
k).getCode()))
                                    {
                                          found = 1;
                                    }
                              }
                              if (found == 0)
                              {

      output_customer_list.add(invoice_list.get(j).getCustomer());
                              }
                              ArrayList<Invoice_Item>
temp_invoice_item_list = new ArrayList<Invoice_Item>();
                              temp_invoice_item_list =
invoice_list.get(j).getInvoiceItems();
                              for (int k=0;
k<temp_invoice_item_list.size(); k++)
                              {

      output_invoice_item_list.add(temp_invoice_item_list.get(k));
                              }
                        }
                  }
            }

            //Make output files
            //But First, create output folder if it doesn't exists
            String outputPath = "./output_csv/";
            File directory = new File(outputPath);
            if (!directory.exists())
            {
                  directory.mkdir();
            }

            //Then we create and fill the output customer file in ASCII
encoding
            try {
                  File fout = new File("./output_csv/customer.csv");
                  FileOutputStream fos = new FileOutputStream(fout);
                  BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(fos, "US-ASCII"));

      bw.write("\"CUSTOMER_CODE\",\"FIRSTNAME\",\"LASTNAME\"");
                  bw.newLine();
                  for (int i=0; i<output_customer_list.size(); i++)
                  {
                        String s =
                              "\"" +
                              output_customer_list.get(i).getCode() +
                              "\",\"" +
```

```
      output_customer_list.get(i).getFirstName() +
                              "\",\"" +
                              output_customer_list.get(i).getLastName()
+
                              "\"";

                        bw.write(s);
                        bw.newLine();
                  }

                  bw.close();
            } catch (IOException e)
            {
                  e.printStackTrace();
            }

            //Next the invoice item file
            try {
                  File fout = new
File("./output_csv/invoice_item.csv");
                  FileOutputStream fos = new FileOutputStream(fout);
                  BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(fos, "US-ASCII"));

      bw.write("\"INVOICE_CODE\",\"ITEM_CODE\",\"AMMOUNT\",\"QUANTITY\"
");
                  bw.newLine();
                  for (int i=0; i<output_invoice_item_list.size(); i++)
                  {
                        String s =
                              "\"" +
                              output_invoice_item_list.get(i).getCode()
+
                              "\",\"" +

      output_invoice_item_list.get(i).getItemCode() +
                              "\",\"" +

      output_invoice_item_list.get(i).getAmount() +
                              "\",\"" +

      output_invoice_item_list.get(i).getQuantity() +
                              "\"";

                        bw.write(s);
                        bw.newLine();
                  }

                  bw.close();
            } catch (IOException e)
            {
                  e.printStackTrace();
            }

            //And lastly the invoice file
            try {
```

```
                File fout = new File("./output_csv/invoice.csv");
                FileOutputStream fos = new FileOutputStream(fout);
                BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(fos, "US-ASCII"));

      bw.write("\"CUSTOMER_CODE\",\"INVOICE_CODE\",\"AMMOUNT\",\"DATE\"
");
                bw.newLine();
                for (int i=0; i<output_invoice_list.size(); i++)
                {
                      String s =
                            "\"" +

      output_invoice_list.get(i).getCustomerCode() +
                            "\",\"" +

      output_invoice_list.get(i).getInvoiceCode() +
                            "\",\"" +
                            output_invoice_list.get(i).getAmount() +
                            "\",\"" +
                            output_invoice_list.get(i).getDate() +
                            "\"";

                      bw.write(s);
                      bw.newLine();
                }

                bw.close();
            } catch (IOException e)
            {
                e.printStackTrace();
            }

      }
}
```

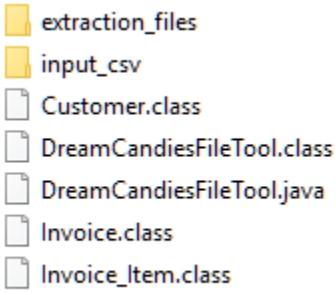
4. Compile and Run

The program compiles without any problem and there are not any errors or warnings.

```
C:\Users\Stratos\Java Projects\DreamCandies File Tool>javac DreamCandiesFileTool.java

C:\Users\Stratos\Java Projects\DreamCandies File Tool>
```

After the compilation, there are class files created as usual. They are the four classes.

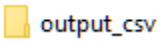


Then the program runs.

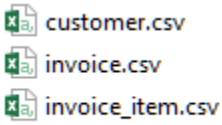
```
C:\Users\Stratos\Java Projects\DreamCandies File Tool>java DreamCandiesFileTool

C:\Users\Stratos\Java Projects\DreamCandies File Tool>
```

As the tool worked properly, there is a new folder named output_csv created.

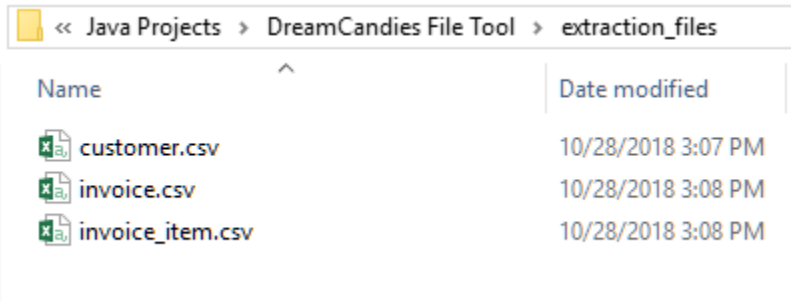


It contains the three output files.



5. Tool Example

Let's say we have the three main extraction files as follows:



- Customer.csv

customer.csv | invoice.csv | invoice_item.csv

```
"CUSTOMER_CODE","FIRSTNAME","LASTNAME"
"CUST0000010231","Maria","Alba"
"CUST0000010235","George","Lucas"
"CUST0000010237","Stratos","Mylonas"
```

- Invoice.csv

customer.csv | invoice.csv | invoice_item.csv

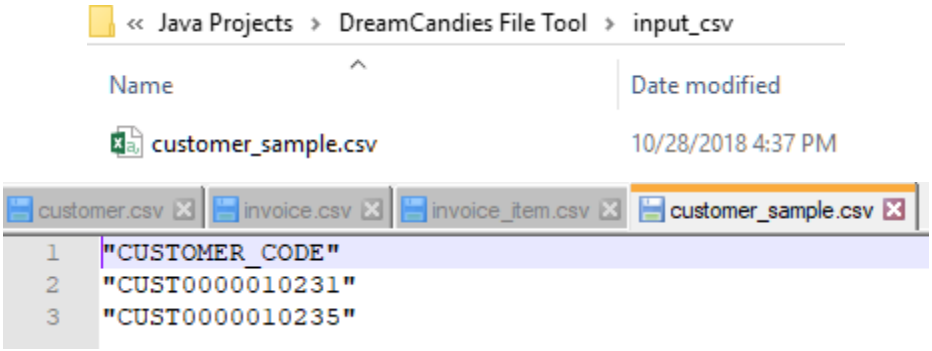
```
"CUSTOMER_CODE","INVOICE_CODE","AMOUNT","DATE"
"CUST0000010231","IN0000001","105.50","01-Jan-2016"
"CUST0000010235","IN0000002","186.53","01-Jan-2016"
"CUST0000010231","IN0000003","114.14","01-Feb-2016"
"CUST0000010237","IN0000005","256.12","05-Feb-2017"
```

- Invoice_item.csv

customer.csv | invoice.csv | invoice_item.csv

```
"INVOICE_CODE","ITEM_CODE","AMOUNT","QUANTITY"
"IN0000001","MEIJI","75.60","100"
"IN0000001","POCKY","10.40","250"
"IN0000001","PUCCHO","19.50","40"
"IN0000002","MEIJI","113.40","150"
"IN0000002","PUCCHO","73.13","150"
"IN0000003","POCKY","16.64","400"
"IN0000003","PUCCHO","97.50","200"
"IN0000005","POCKY","16.64","400"
"IN0000005","PUCCHO","97.50","200"
"IN0000005","POCKY","16.64","400"
"IN0000005","PUCCHO","97.50","200"
```

There is also the input_csv folder containing the sample file.

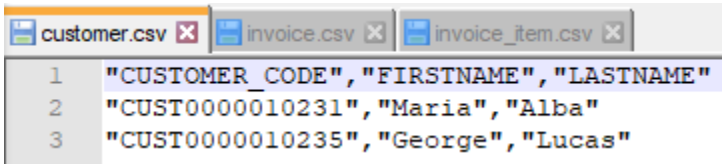


So, the tool has to export the data for customers with code CUST0000010231 and CUST0000010235.

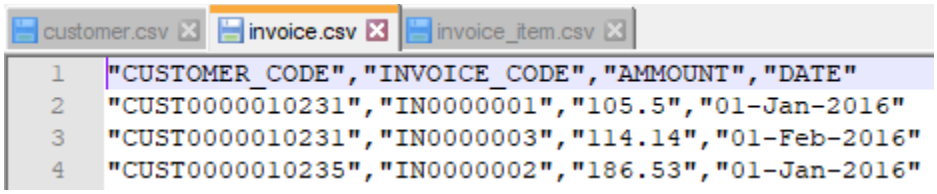
Let's see.

After the running process we correctly get the output folder (output_csv), including the subset files for the specific two customers.

- Customer.csv



- Invoice.csv



- Invoice_Item.csv