**Object Oriented Programming**

**Assignment 4** **Jacqui Chetty**

**Job Candidates**

Marks available: 100

Percentage of overall marks from this assignment: 20%

Date issued: 2nd December 2020

Deadline: 6th January 2021

## Introduction

For this assignment you will create classes and files to **produce a list of candidates that can be interviewed for a potential job**.

Employment agencies receive hundreds of job applications every day. In order to identify suitable candidates, a computer-aided software approach can be used. Instead of making use of people to examine these CVs, it would be far more efficient and faster to provide an application that can read a file containing all of the CVs; so that the potential candidates can be identified (based on certain criteria) for the interview process.

One way of achieving this can be to make use of a 'dirty' file that contains all potential CVs, and then produce a 'clean' file – one in which the potential candidates are listed in a particular structure. The reason for creating the 'clean' file is to put the CV data into a format that can easily be manipulated. Once you have the 'clean' file, the next step would be to look at each candidate in the 'clean' file and identify (based on certain criteria) whether a candidate is suitable for an interview or not. For example, a suitable candidate may be a computer programmer with 5 years' experience.

Additionally, an application could create a file *only* with very experienced candidates. For example, this may be candidates with $5^+$ years' experience as a computer programmer. The file can also be converted into a .csv file so that it can easily be opened and used as a spreadsheet.

Looking at an example, the following file is the 'dirty' file, aptly names ***dirtycv.txt*** and it contains 5 potential candidates. Each CV starts with a *CV #* (# is replaced with 1 or

2 or 3, etc). Each CV ends with the word *End* (every 2nd CV is highlighted to split the CVs up – for ease of reading – for this document reading only).

**dirtycv.txt**

```
CV 1

Surname:Smith

First Name:John

Address:1 Whatever Street

Qualification:Degree in Computer Science

Position:Data Analyst

Experience:5

Position:Computer programmer

Experience:2

eMail:smithj@gmail.com

End

CV 2

Surname:Harrison

First Name:Jane

Qualification:None

eMail:harrisonj@gmail.com

End

CV 3

Surname:Jones

First Name:Richard

Address:2 Whatever Street

Qualification:Masters in Computer Science

Position:Operator

Experience:10

Position:Computer programmer

Experience:2

eMail:jonesr@gmail.com
```

```
End

CV 4

Surname:Ngomi

First Name:Mary

Address:3 Whatever Street

Qualification:Degree in Computer Science

Position:Programmer

Experience:10

eMail:ngomim@gmail.com

End

CV 5

Surname:Chen

First Name:Chris

Address:4 Whatever Street

Qualification:Masters in Computer Science

Position:Programmer

Experience:7

eMail:chenc@gmail.com

End
```

This file is GIVEN to you. When your application is tested, we will make use of a different data set, i.e. a different dirtycv.txt file.

The first step is for you to create a class with methods and produce a 'clean' CV file, named cleancv.txt - The cleancv.txt should display all of the CVs in the following way (take note that the 'fields' are separated with commas):

**cleancv.txt**

```
Smith0001,Degree in Computer Science,Data Analyst,5,Computer programmer,2,smithj@gmail.com,

Harrison0002,None,harrisonj@gmail.com,

Jones0003,Masters in Computer Science,Operator,10,Computer programmer,2,jonesr@gmail.com,

Ngomi0004,Degree in Computer Science,Programmer,10,ngomim@gmail.com,

Chen0005,Masters in Computer Science,Programmer,7,chenc@gmail.com,
```

Once the cleancv.txt has been created, the next step is for the application to produce a list of potential candidates to be interviewed (based on criteria as stipulated below in the instructions). Given the example above, we would like to choose the candidates that have either a Degree in Computer Science or a Masters in Computer Science. Other criteria relate to the types of positions they hold, i.e. an operator; as well as the number of years' experience they have. This means that the cleancv.txt file should provide a list of candidates placed in a new file called the to-interview.txt file and again, given the above example, it would look as follows:

**to-interview.txt**

```
Smith0001 Degree in Computer Science Data Analyst 5 Computer programmer 2 smithj@gmail.com

Jones0003 Masters in Computer Science Operator 10 Computer programmer 2 jonesr@gmail.com

Ngomi0004 Degree in Computer Science Programmer 10 ngomim@gmail.com

Chen0005 Masters in Computer Science Programmer 7 chenc@gmail.com
```

Only candidates 1, 3, 4 and 5 have been chosen, according to certain criteria. Additionally, it would be standard practice for this file to be a .csv file as part of the assignment you will be expected to produce a .csv file. This allows people to open the file easily using a spreadsheet. The .csv file is to be called to-interview-table-format.csv

The last step is to produce a list of experienced candidates. In this instance, if you were only looking for candidates that have more than 5 years of experience then Jones0003, Ngomi0004 and Chen0005 makes the cut.

**to-interview-experience.txt**

```
Jones0003 10

Ngomi0004 10

Chen0005 7
```

### Submission instructions

Submit your work by the deadline above as a zip archive. Submit only .java source files as well as .txt and .csv files in your archive unless told otherwise.

**Make absolutely sure you are not submitting .class files instead of .java files.**

Each Java file must have a package declaration at the topic of it (these declarations must be on the very first line of the file). The final part of this declaration varies according to the assignment and exercise.

**The package declaration for this assignment is:**

```
package com.bham.pij.assignments.candidates;
```

Do not copy and paste this package declaration from this pdf! You will get unexpected characters in your files that will render them untestable. Type it in.

Your package declaration must be on the very first line of your file. Do not put anything above the package declaration, not even comments.

**Do not forget to add the semicolon** at the end of the package declaration. All characters in a package name should be lower case.

In each of these questions, you are not restricted to only creating the required methods, but you must create at least the required methods. Please abide by the rules of the question.

All of the required methods must be public.

None of the required methods should be declared as static.

Do not use non-ASCII characters anywhere in your files, even as comments.

All strings can be treated as case insensitive unless otherwise stated.

**Introduction**

For this assignment you will create classes and methods that will be used to develop an application for an employment agency. The end goal is to identify potential candidate interviewees for a job.

*Each question involves the creation of a .txt file as output.* You need to create all of the classes yourself as well as most of the .txt files.

**Please do NOT specify any path names. For example, you would make use of the following relative path when creating/using a file:**

```
Path fileIn = Paths.get("dirtycv.txt");
```

The only validation that is required is stated in the exercises below.

As you do each question, you are strongly recommended to check that the functionality you added in the previous exercise still works. This is called regression testing.

**Creating an application to produce a file that lists potential candidates for a job interview.**

As discussed, employment agencies find it difficult to examine hundreds of CVs to determine who needs to be interviewed. It is therefore very difficult for agencies to properly extract information from CVs, and potentially good candidates may not be highlighted.

You have been tasked with writing a Java application to produce an automated list of potential candidates to be interviewed, stored in a series of files. Each question produces a file. It is important that a file is produced as this is what will be marked.

**Question 1: [30]**

Write a Java class called **CleaningUp** with methods that will take an existing file (the **dirtycv.txt** that has been given to you) and clean the file up. The cleaned-up version must be written to a new file called **cleancv.txt** file.

For question 1 create a method **public void cleanUpFile()**. You have been given the dirtycv.txt file (to be used as input / read). The method must re-structure the dirtycv.txt file so that each candidate can be viewed as a record, taking up exactly 1 line in the new file (the cleancv.txt). For example, CV 1 is John Smith in the dirtycv.txt file and appears as follows:

```
CV 1

Surname:Smith

First Name:John

Address:1 Whatever Street

Qualification:Degree in Computer Science

Position:Data Analyst

Experience:5

Position:Computer programmer

Experience:2

eMail:smithj@gmail.com
```

```
End
```

Once this CV is cleaned up it should appear as follows in the cleancv.txt file:

```
Smith0001,Degree in Computer Science,Data Analyst,5,Computer programmer,2,smithj@gmail.com,
```

Take note that there are 5 CVs in the dirtycv.txt file and that means there needs to be 5 lines (records) in the cleancv.txt file (see above for the complete example).

The dirtycv.txt file needs to be cleaned up so that each line is formatted accordingly:

*Identifier,qualification,position,experience,position,experience,eMail,*

Identifier – Smith0001

Qualification – Degree in Computer Science

Position – Data Analyst

Experience – 5 (this would be 5 years)

Position – Computer programmer

Experience – 2

eMail – smithj@gmail.com

Remember that this is an example illustrating what the output could look like. This is not to say that it will always be John Smith.

**Take note of the following:**

- The identifier needs to be generated, i.e. Smith0001 (it is mandatory that there is a 000 placed in front of the 1. So, CV 1 correlates to Smith0001;
- This is a comma separated file;
- There are no spaces between commas;
- The last character in the line is a comma;
- There are variations between CVs in the dirtycv.txt file, for example CV 2 has no address.

Create a main() method in its own class (***JobCandidatesMain***) and test the program before continuing.

**To summarise**, at the end of this question you should produce a file called the **cleancv.txt** file. The contents of the file should appear as follows:

```
Smith0001,Degree in Computer Science,Data Analyst,5,Computer programmer,2,smithj@gmail.com,
Harrison0002,None,harrisonj@gmail.com,
Jones0003,Masters in Computer Science,Operator,10,Computer programmer,2,jonesr@gmail.com,
Ngomi0004,Degree in Computer Science,Programmer,10,ngomim@gmail.com,
Chen0005,Masters in Computer Science,Programmer,7,chenc@gmail.com,
```

## Question 2: [40]

You will need to make use of the **cleancv.txt** file that you created in Question 1, to complete Question 2.

Now that you have a clean file; the next step is to extract *only* those candidates that the agency would like to interview. In other words, *the output of this question should produce a new file that only holds the candidates to be interviewed.* This file should be called **to-interview.txt**

Create a class called **CandidatesToInterview** and ensure that you have relative path names for the files:

**The criteria for the candidates to become potential interviewees are as follows:**

The potential candidates are going to be candidates that have

- suitable qualifications
- experience

Create a method called **public void findCandidates()** and make use of the following arrays as keywords to determine which candidates get chosen for interviews.

```
String [] keywordsDegree = {"Degree in Computer Science", "Masters in
Computer Science"};

String [] keywordsExperience = {"Data Analyst", "Programmer", "Computer
programmer", "Operator"};
```

For this assignment, these are the keywords used for the degrees as well as the years of experience. In the real world this would not be so. To clarify, the keywords array *keywordsDegree* means that the candidates must *either* have a Degree in Computer Science **or** a Masters in Computer Science. Similarly, the keywords array *keywordsExperience* are candidates that must have experience as *either* data analysts **or** Programmer **or** Computer programmer **or** Operator.

For this class, the program must determine whether a candidate in the **cleancv.txt** file should be chosen to be interviewed. If so, the candidate details are to be written to the **to-interview.txt** file based on the keywords as described above. The end result for this example is as follows:

**to-interview.txt**

```
Smith0001 Degree in Computer Science Data Analyst 5 Computer programmer 2 smithj@gmail.com

Jones0003 Masters in Computer Science Operator 10 Computer programmer 2 jonesr@gmail.com

Ngomi0004 Degree in Computer Science Programmer 10 ngomim@gmail.com

Chen0005 Masters in Computer Science Programmer 7 chenc@gmail.com
```

The following form part of each line in the to-interview.txt file:

- There is a space between each field (a field being Smith0001 or Degree in Computer Science, etc)
- The following data is captured as part of the record
    - Identifier
    - Qualification
    - Position (most current)
    - Years of experience (most current)
    - Position
    - Years of experience

Take note that Harrison did not make the list as they do not have a qualification. Although Ngomi0004 and Chen0005 have the experience related to being a programmer *only*, they still qualify. Smith0001 and Jones0003 qualify as they have experience related to 2 roles. All the qualifiers have either a Degree in Computer Science or a Masters in Computer Science.

**To summarise**, at the end of this question you should produce a file called the **to-interview.txt** file.

Go to class ***JobCandidatesMain*** and test the program before continuing.

**Question 3: [10]**

Following on from question 2 and staying within the class **CandidatesToInterview,** create another method **public void candidatesWithExperience()** that will produce a file only showing candidates that have *more than 5 years of experience* in their current role. The current role is the first role listed, such as data analyst for Smith0001, operator for Jones0002, programmer for Ngomi0004 and programmer for Chen0005.

You will make use of the **to-interview.txt** file that you created in question 2, to read in the candidates. Extract only those candidates that qualify, placing them in a new file, the **to-interview-experience.txt** file.

**to-interview-experience.txt**

```
Jones0003 10

Ngomi0004 10

Chen0005 7
```

**Hint:** The use of a multi-dimensional array (or any other data structure) to store potential candidates (in question 2) may make it easier to extract the data from the to-interview.txt file so that you get the to-interview-experience.txt file (only those with more than 5 years' experience).

**To summarise**, at the end of this question you should produce a file called the **to-interview-experience.txt** file.

Go to class ***JobCandidatesMain*** and test the program before continuing.

At the end of question 3 you should have the following files as output:

From Question 1

- cleancv.txt -> all CVs arranged in a particular format read from dirtycv.txt

From Question 2

- to-interview.txt -> only the candidates that have qualified

From Question 3

- to-interview-experience.txt -> only the candidates that have more than 5 years; experience

**Question 4: [2 x 10, write to .csv file and read from .csv file]**

The last step is to place all the candidates to be interviewed (use the to-interview.txt file) into a **.csv** file so that the employment agency within the organisation can make use of the file in a spreadsheet. Make another method in the

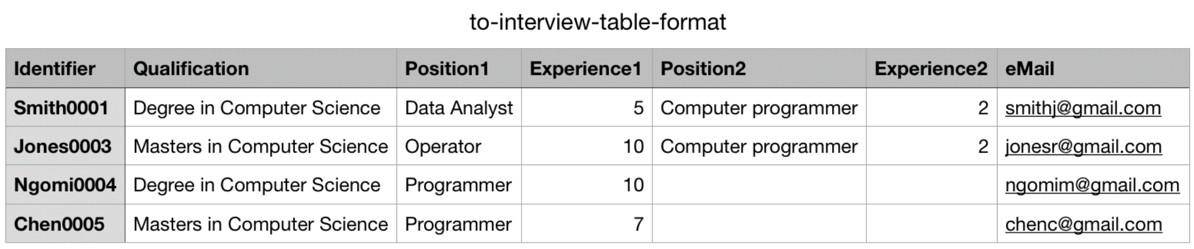
**CandidatesToInterview** class called **public void createCSVFile()** and read from the to-interview.txt file to ensure that the candidates to be interviewed are placed into a to-interview-table-format.**csv** file. You will need to do some research around how to create a .csv file.

**Hint:** You can make use of the multi-dimensional array created earlier.

An example of what your .csv file should look like is shown here. Take note of the following:

- the table has a heading in the first row of the spreadsheet;
- the eMail address is positioned correctly when there a candidate has only **one** current position and experience.

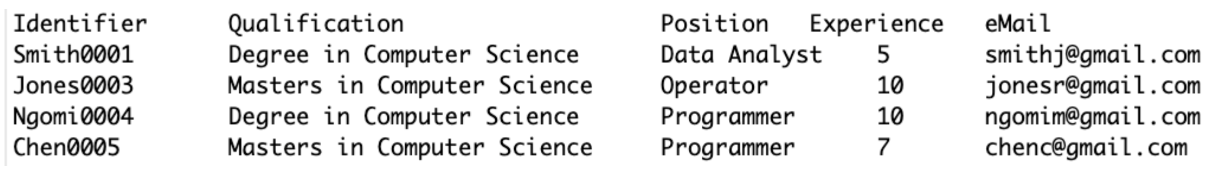
**to-interview-table-format.csv**



to-interview-table-format

| Identifier | Qualification | Position1 | Experience1 | Position2 | Experience2 | eMail |
|---|---|---|---|---|---|---|
| **Smith0001** | Degree in Computer Science | Data Analyst | 5 | Computer programmer | 2 | smithj@gmail.com |
| **Jones0003** | Masters in Computer Science | Operator | 10 | Computer programmer | 2 | jonesr@gmail.com |
| **Ngomi0004** | Degree in Computer Science | Programmer | 10 | | | ngomim@gmail.com |
| **Chen0005** | Masters in Computer Science | Programmer | 7 | | | chenc@gmail.com |

Make another method (just for fun and for marks of course) called **public void createReport()** that reads the records from the to-interview-table-format.csv file and prints the contents to the console in the following format:



```
Identifier      Qualification                   Position   Experience   eMail
Smith0001       Degree in Computer Science      Data Analyst    5       smithj@gmail.com
Jones0003       Masters in Computer Science     Operator        10      jonesr@gmail.com
Ngomi0004       Degree in Computer Science      Programmer      10      ngomim@gmail.com
Chen0005        Masters in Computer Science     Programmer      7       chenc@gmail.com
```

Take note that for this report only the current role is outputted. This is just a report that is sent to the console and is not related to a file. This means that you need to take a screenshot of the output and place it into a text editor file such as Word or TextEdit. The file is to be called **screenshot-candidates**

Go to class **JobCandidatesMain** and test the program before submitting.

The following files are to be handed in as part of your submission:

- .java source file(s)
- cleancv.txt
- to-interview.txt
- to-interview-experience.txt

- to-interview-table-format.csv
- screenshot-candidates (a text doc)