For this exam, you will be working with a real life problem… Currently, Canvas sends professors a report of all grades that have been modified. Grade changes are made by the professor, TA’s, and Graders of a course. The problem is the email is formatted ridiculously. The system sends a comma separated file format – a simple list that is very difficult to read. So, your job is to write a Java program that will open a file containing this raw data (see example below) and produce a well formatted printout of the data which is easier to view. The file’s data is comma separated and has NO header row. File name: Exam\_01-C.txt Copy/download this file to your project folder. Create a workspace folder on your desktop and start a new project.

Below is an actual output of 3 records sent to me (names have been changed to protect the innocent)

***Transaction ID:*** *3122b63e-5f6c-4ab2-8f94-ff2880fe2b1c,****First name:****Tom,****Last name:****Ryan,****Previous grade:****80,****New grade:****81,****Grade******desc:****Exam\_01,****Source system:****Canvas,****Grade updated by:****nelsonmp,****Grade updated date:****09/20/2018 04:55:06 PM*

*Transaction ID: 64c3c3e9-b379-45c2-b0fa-2e5ba277e6e4,First name:Sally,Last name:Powers,Previous grade:79,New grade:80,Grade desc:Exam\_01,Source system:Canvas,Grade updated by:nelsonmp,Grade updated date:09/20/2018 04:55:06 PM*

*Transaction ID: 124b3451-d4a5-43d0-ac33-8e2f03556b3e,First name:Jack,Last name:Ryan,Previous grade:81,New grade:82,Grade desc:Exam\_01,Source system:Canvas,Grade updated by:nelsonmp,Grade updated date:09/20/2018 04:55:05 PM*

As you can see, the data is not in useable form. It is just too difficult to see quickly the activity by the grader. You are to open the file and read in each line into a class called GradeLine where each field is filled in with the correct data. Note, this is like what we did on Tuesday where each line also includes the field name (e.g., “First name:”, “Last name:”, etc) so you will need to parse the full line by commas as well as removing the field names. For example, take the first line from above:

**First name:Tom, Last name:Ryan, Previous grade:80, New grade:81, Grade desc:Exam\_01, Source system:Canvas, Grade updated by:nelsonmp, Grade updated date/time:09/20/2018 04:55:06 PM**

This will be the line you read from the file. You should covert this to raw data by removing the field names so it looks like:

**Tom,Ryan,80,81,Exam\_01,Canvas,nelsonmp,09/20/2018 04:55:06 PM**

Once you have it in this form split the string by the commas to build your GradeLine object and then add it to a HashMap object where you will be able to look up a record by the TransactionID.

Your GradeLine class should have the following:

1. Constructors: (1) Workhorse, (2) Copy, (3) one that accepts a RandomAccessFile, (4) one that accepts a Scanner, (5) one that accepts an array of Strings, (6) one that accepts a String (this would be a string of data from the file)
2. Private properties / Public getters & setters
3. Methods: (1) equals, (2) clone, (3) compareTo, (4) sortString(), (5) toString, (6) save(RandomAccessFile raf), (7) save(PrintWriter pw), (8) getDiff() that returns the difference between the fromGrade to the toGrade (i.e., return toGrade – fromGrade;)

* Write the code for the compareTo() method so that the class is sorted by (hint: in order to include the updatedDateTime as a sortable data you will need to reformat it to the following form: 2018/09/20):
  1. UpdatedByUserID - lastName - firstName - gradeDescription – updatedDateTime
* Write a Tester class that will open and import all records from the file. Do not allow for duplicate TransactionID records to be read in but, rather, print them to the screen
* Select any TransactionID from the file and do a search on that ID and print the GradeLine object for that transaction
* Loop through all records and print out the following format:

lstName firstName gradeDescription fromGrade toGrade difference UpdatedByUserID updatedDateTime

**When finished zip up your two files (Tester.java and GradeLine.java) and upload to Canvas**

1. Create a workspace on the desktop of your computer and start a Java program using this folder
2. If you don’t already have it copy the Exam\_01-D.txt to your project folder.
3. Look at the data file and create a class called Animal to represent the data. Use proper coding practices:
   1. Properties
      1. private
      2. public getters / setters
   2. Constructors
      1. Workhorse
      2. Copy
      3. Special Case
         1. Accepts an array of Strings
         2. Accepts a single String that represents a row from the file
         3. Accepts a RandomAccessFile
         4. Accepts a Scanner
   3. Methods
      1. clone()
      2. equals()
      3. save(RandomAccessFile raf)
      4. save(PrintWriter pw)
      5. getFormalName() -> returns the owner’s title and last name (note the owner’s data is both first and last name so you’ll need to extract the last name for this method)
      6. toString() that returns a nicely formatted string with some data of the animal
      7. compareTo() so that your Animal can be sorted by the animal and then the owner’s name
      8. sortString() method that returns the proper sort string used to sort the Animal
4. Read in file
   1. Read in each row into your app and convert it to an object of class Animal and add it to a HashMap object called animals where you will be able to look up a record by the animal’s ID
   2. Do not allow for duplicate ID’s to be imported (print them to the console as duplicate record)
5. File Import Rules:
   1. If the animal has a comma, put the string after the comma in front of the data – do this logic in the setter only!
      1. e.g., “Tern, white-winged black” should change to: “white-winged black tern” (note: all lower case)
   2. If animal has, as part of its description, “(unidentified)” you are to remove it – do this logic in the setter only!
      1. e.g., “Woodrat (unidentified)” should change to: “Woodrat”
   3. When reading in the cost of the animal that the owner paid convert it to double. It will come in as “$5623.33” and you will need to convert that to: 5623.33 in order to store it as a double.
6. Output:
   1. While reading in the data print the animals that had duplicate animal ID’s but don’t import them
   2. Select any recordID from the file and do a search on that ID and print the Animal object for that record ID
   3. Print all owners that are doctors (based on the “title” field being “Dr”) with animals that are from France
   4. Print the total amount of money spent on animals from China

**When finished zip up your two files (Tester.java and Animal.java) and upload to Canvas**

1. Using your desktop as your workspace create a new java project (name doesn’t matter)
2. Copy the Exam\_01-E.txt to your project folder. This file is located on your desktop.
3. Look at the data file and create a class called Athlete to represent the data. Use proper coding practices:

*(fields:* ***athleteID athleteName athleteBornIn age teamName yearsOnTeam salary currentlyInjured****)*

* 1. Properties
     1. private
     2. public getters / setters
  2. Constructors
     1. Workhorse
     2. Copy
     3. Special Case
        1. Accepts an array of Strings
        2. Accepts a single String that represents a row from the file
        3. Accepts a RandomAccessFile
        4. Accepts a Scanner
  3. Methods
     1. clone()
     2. equals()
     3. save(RandomAccessFile raf)
     4. save(PrintWriter pw)
     5. getFormalName() -> returns the athelte’s name but in the form “last name, first name” – note that the athelte’s name is one String value so you’ll need to split the name and reorder it for this method)
     6. toString() that returns a nicely formatted string with some data of the animal
     7. compareTo() so that your Athlete can be sorted by their salary
     8. sortString() method that returns the string that is used in the compareTo() method. Sort by:  
        teamName – athelteBornIn - athelteName

1. Read in file
   1. Read in each row into your app and convert it to an object of class Athlete and add it to a HashMap object called athletes where you will be able to look up a record by the athleteID
   2. Do not allow for duplicate ID’s to be imported (print them to the console using the toString()) for example:

Duplicate Record: 25f55edc-972c-4f58-880b-20cdaf813c24 Roxana Duce Denver Broncos

1. File Import Rules:
   1. If the athlete is under 20 years old print them out as “under age”
      1. e.g., **Under Aged Athlete Found: Samuel Fortescue 19 years old**
   2. Keep a count (and print at the end) any athlete that is currently injured
      1. e.g., **There were 1,331 injured athletes**
   3. When reading in the cost of the animal that the owner paid convert it to double. It will come in as “$5623.33” and you will need to convert that to: 5623.33 in order to store it as a double.
2. Output:
   1. While reading in the data print the athletes that have duplicate athlete ID’s but don’t import them in
   2. Select any athleteID from the file and do a search on that ID and print the Athlete object for that athleteID
   3. Print all athletes that played for the “Arizona Cardinals” and were born in Ohio
   4. Print the total salary for all athletes that played for the “Cincinnati Bengals” and are in their 30’s

**When finished zip up your two files (Tester.java and Athlete.java) and upload to Canvas**