Final Project (50 points)

This assignment covers all chapters to date throughout our Java Journey over the semester including:

- Strings
- Conditional Statements
- Loops
- Objects
- Arrays, ArrayList
- Inheritance, Polymorphism, Interfaces
- Exceptions
- File I/O

You have come a long way and should be very proud of what you accomplished over the semester.

In this lab, you will be creating a license registration tracking system for the Country of Warner Brothers for the State of Looney Tunes. You will create four classes: Citizen, CarOwner, RegistrationMethods, and RegistrationDemo. You will build a CitizenInterface and CarOwnerInterface and implement CitizenInterface and CarOwnerInterface for Citizen and CarOwner classes respectively. You will create RegistrationMethods class that implements RegistrationMethodsInterface(provided).

Citizen Interface and class

1. Create getter and setter headers for each of the instance vars, String firstName and String lastName (see UML below)
2. toString() returns a String with firstName, a space, and lastName (Note the csv file has these reversed)

Citizen

- firstName: String - lastName: String
+setFirstName(String inFirst): void +getFirstName(): String +setLastName(String inLast): void +getLastName(): String +toString(): String

Citizen implements CitizenInterface and Serializable and has two constructors - a default constructor that sets firstName and lastName to “No Name” (ie firstName = “No Name”) and one that sets firstName and lastName to values passed in. NOTE – Serializable must be implemented since we will be writing objects to binary files and CarOwner is extending Citizen. You will need to import java.io.Serializable.

CONTINUED ON NEXT PAGE

CarOwner

```
- license: String
- month: int
- year: int
---------------------------------------
+setLicense(String inLicense): void
+getLicense (): String
+setmonth(int inMonth): void
+getMonth(): int
+setYear(int inYear): void
+getYear(): int
+compareTo(Object o): int
+toString(): String
```

CarOwner class ***extends*** Citizen and ***implements*** CarOwnerInterface, Serializable and has two constructors. The default CarOwner constructor sets first and last name to “No Name”, license to “Not Assigned”, month to 0, and year to 0. Another constructor sets first and last name, license, month, and year to the values passed in. NOTE – Serializable must be used since we will be writing objects to binary files.

CarOwnerInterface.java + CarOwner.java Method specifics

1) CarOwnerInterface.java extends Comparable. NOTE - Since Comparable is an interface, CarOwnerInterface extends Comparable. This requires CarOwnerInterface to have compareTo() and therefore compareTo() must be written out in CarOwner. Doing this provides a mechanism to sort CarOwners.
2) Create getter and setter headers for each of the instance vars String license, int month, and int year (see UML above)
3) Overrides .toString() and returns a String with the following data: firstName, a space, lastName, then license, and then month/year. Since CarOwner extends Citizen, take advantage of super.toSting( ) for some of CarOwner’s toString( ). Ensure that your output lines up nicely (see output.txt in Canvas) HINT: Remember what “\t” does.
4) compareTo(Object o) - compares CarOwner objects and if the calling object is earlier in chronological time in comparison to the argument returns -1, if the calling object is later in chronological time in comparison to the argument returns 1, if the calling object and argument are the same in chronological time returns 0. NOTE: if the argument passed in is not a CarOwner object (use instanceof or getClass to determine this) or is null, compareTo( ) returns -1. Since we have not officially covered compareTo( ) and this is a later advanced topic (Chapter 15), below is the compareTo( ) that you can cut and paste into BlueJ.

CONTINUED ON NEXT PAGE

```
/**
     * overrides compareTo to sort CarOwners based on chronological
     * of month and year car was last.  If param is null or not CarOwner
class, returns -1
     * if object total months < param total months, temp is -1, object total
months > param total months, temp is 1
     * otherwise the same and temp is 0
     * @return temp
     */
    public int compareTo(Object o){
        int temp = -1;
        if(o != null && getClass() == o.getClass()){
            CarOwner copy = (CarOwner)o;
            if((year*12+month) < (copy.year*12+copy.month))
                temp = -1;
            else if ((year*12+month) > (copy.year*12+copy.month))
                temp = 1;
            else //the same
                temp = 0;
        }
        return temp;
    }
```

RegistrationMethods class ***implements*** RegistrationMethodsInterface (provided). RegistrationMethods has three String instance vars, inputFileName, outputFileName, and binFileName. NOTE – RegistrationMethods requires importing java.util.ArrayList, java.util.Scanner (or java.util.*) along with java.io.*.
Here is a summary of RegistrationMethods ():

1. setFileNames() – Prompt the user to provide the location of the csv file to be processed (registration.csv). The method should create a File object to test if a file exists at the path provided. If it does not, hold the user until the correct path is entered. Set inputFileName with this file/path. Prompt the user for file / path for the desired location of the output file (output.txt). Set outputFileName with this file/path. Prompt the user for file / path for desired the location of the binary output file (binFile.dat). Set binFileName with this file/path. NOTE – For Windows, you might want to see if you have a c:/tmp folder, if not create one and put registration.csv file in here (ie c:/tmp/registration.csv), save the output.txt file here (ie c:/tmp/output.txt), and the binFile.dat here (c:/tmp/binFile.dat). For Mac OS, you should have a /tmp folder already, but can create one if necessary and put registration.csv file in here (ie /tmp/registration.csv), save the output.txt file here (ie /tmp/output.txt), and the binFile.dat here (/tmp/binFile.dat).
2. processTextToArrayList(ArrayList<CarOwner> inList) – Adds new CarOwner objects to the inList CarOwner collection passed in. For each line of the registration.csv file, a CarOwner object is added to inList. Pitfall – Make sure you clear the first line of the csv (String input = inputStream.nextLine()). Then you can use split based on "," to create a String array. Once you split and have the components how can you use these piece parts to create an object for inList - think new, what does new call? You will need to try, catch this.
3. printArrayListToFile(ArrayList<CarOwner> inList, String inMsg) - Prints inList to a text file. Message Header is based on inMsg passed in. You will need to try, catch this.
4. writeListToBinary(ArrayList<CarOwner> inList) – Writes a ArrayList<CarOwner> collection to a binary file (ie, binFile.dat). You will need to try, catch this.

5. readListFromBinary() – Reads an ArrayList<CarOwner> collection from a binary file (binFile.dat). Then, each ArrayList object item is then written to a newly created CarOwner[ ] called temp. temp is returned to the calling method. *In this method you will need to pull the CarOwner objects from the ArrayList and put into a CarOwner array.* You will need to try, catch this.
6. printArrayToFile(CarOwner[ ] inArray, String inMsg) - Prints inArray to a text file. Message Header is based inMsg passed in. You will need to try, catch this.
7. flagOverdueOwners(CarOwner[] inArray) - generates and returns an array for vehicles whose registration have expired. This is defined as registrations that are over 12 months old based on current REG_MONTH and REG_YEAR. HINT - A helpful way to do this method and almostDue is how you completed findAll() in Lab7. Do one pass to get the number of occurrences. Create an array based on this size, and do another pass and add applicable owners to the new array. NOTE: REG_MONTH and REG_YEAR is set in RegistrationMethodsInterface.
8. flagAlmostDueOwners(CarOwner[] inArray) - Method that generates and returns an array for vehicles whose registration will expire in three months or less (months 10-12). The state of Looney Tunes sends a reminder three months out to the car owner.

RegDemo Class details

1. Create a RegistrationMethods object called `dmv`
2. Invoke setFileNames( ) using the object created above
3. Create an ArrayList<CarOwner> collection called `ltState`.
4. Invoke processTextToArrayList( ) passing in `ltState` as an argument
5. Invoke printArrayListToFile () passing the appropriate arguments (see RegistrationMethods methods summary above). For the String argument pass the following phrase "Initial Set of Car Owners - Unsorted"
6. Invoke writeListToBinary() using `ltState` as an argument
7. Invoke readListFromBinary() assigning the result to CarOwner[ ] `ltStateCopy`.
8. Use Arrays.sort method to sort `ltStateCopy`
9. Use the appropriate RegistrationMethods method to print a copy of `ltStateCopy` with the String header "Sorted list based on Registration date"
10. Use flagOverdueOwners( ) to create a new CarOwner[ ] whose registration is over 1 yr old that is assigned to CarOwner[ ] `overdue` based on passing in `ltStateCopy`.
11. Use the appropriate RegistrationMethods method to print `overdue` with the String header "Owners with Expired Registration"
12. Use flagAlmostDueOwners( ) to create a new CarOwner[ ] whose registration will expire within the next 3 months or less, but is not over one year old (registrations that are 10-12 months old) that is assigned to CarOwner[] `almostDue` based on the array `ltStateCopy`.
13. Use the appropriate RegistrationMethods method to print `almostDue` with the String header "Owners with registration expiring in three months or less"

CONTINUED ON NEXT PAGE

Rubric

Citizen Interface – 2pt
Citizen.java –5pts
CarOwner Interface – 3pts
CarOwner.java – 10pts
RegistrationMethods.java – 20pts
RegistrationDemo.java –5pts
Correct Output including alignment – 2pt
Javadocs – 3pts

Submitting your work

For all labs you will need to provide a copy of all .java files. **No need to provide .class files. I cannot read these.** ***NOTE – For Replit, please update Main.java to another name such as TempProb.java, ProChall3.java, etc.*** In addition to your .java files, you will need to provide output files of your console. The name of the output file should match the class name and have the .txt extension such as TempProbOut.txt, ProChall3Output.txt. For GUIs such as JOptionPane, you will instead need to create screenshots. For Windows users, Snipping Tool is a great way to do this. Chromebook - Shift+Ctrl+Show Windows. Mac OS users, you can see how to take screenshots using the following url - https://support.apple.com/en-us/HT201361.