**Lab 10** (75 points)

This is it – Almost Done! Note the point value **75 pts.**

This assignment covers all 10 chapters that we have discussed throughout our Java Journey over the semester including:

* Strings
* Conditional Statements
* Loops
* Objects
* Arrays
* 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

1. Create getter and setter headers for each of the instance vars String firstName and String lastName (see UML below)

Citizen

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

Citizen implements CitizenInterface and has two constructors - a default constructor that sets firstName and lastName to null (ie firstName = null) and one that sets firstName and lastName to values passed in.

CarOwner Interface

1. extends Comparable
2. Create getter and setter headers for each of the instance vars String license, int month, and int year (see UML below)
3. Overrides .toString() and prints out firstName, a space, lastName, then license, and then month/year. Ensure that your output lines up nicely (see output.txt in Bb as an example)  
   HINT: Reach back to Ch2 and re-discover 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 or is null, returns -1.

CarOwner

|  |
| --- |
| - license: String  - month: int  - year: int |
| +setLicense(String inFirst): void  +getLicense (): String  +setmonth(int inMonth): void  +getMonth(): int  +setYear(int inMonth): void  +getYear(): int  +compareTo(object o): int  +toString(): String |

***CarOwner*** class ***extends*** Citizen and ***implements*** CarOwnerInterface and has two constructors. The default CarOwner constructor sets first and last name to null, 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.

***RegistrationMethods*** class ***implements*** RegistrationMethodsInterface (provided)

Here is a summary of RegistrationMethods ():

1. 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.
2. flagDinqOwners(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.
3. printArrayToFile(CarOwner[] inArray, String inMsg) - Prints out to a file (1) Message Header based on the String passed in (2) valid entries for the array passed in (ie do not print to file any empty CarOwner objects – for example objects with null as firstName)
4. processText2Array(CarOwner[] inArray) – Updates CarOwner array objects for each line of a csv file for the array passed in and then returns the processed array passed in to the calling method.
5. setFileNames() – Prompts the user to provide the location of the csv file to be processed as well as the location for where the user wants the output file saved.

RegistrationDemo Driver Class

1. Create a RegistrationMethods object
2. Create a CarOwner[] named ltState that is based on the ARRAY\_SIZE named constant in RegistrationMethodsInterface
3. In order to avoid a NullPointerException error, you should initialize all elements in ltState[] to default CarOwner objects (***hint do this in a for loop***)
4. Invoke setFileNames() using the object created above
5. Invoke processTextToArray() passing in the array created above as an argument
6. Invoke printArrayToFile() passing the appropriate arguments (see RegistrationMethods methods summary above), for the array pass in array created above and for the String argument pass the following phrase "Initial Set of Car Owners - Unsorted"
7. Use the Arrays.copyOf() with appropriate arguments to create a copy of ltState that is the exact size of ltState and assign to a CarOwner[] called ltStateCopy
8. Use the appropriate Arrays 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" (hint see step 6 above)
10. Use the appropriate RegistrationMethods method to create a new array of CarOwners whose registration is over 1 yr old called dinqOwners based on the array ltStateCopy. Remember what you did in Lab7 for FindAll()
11. Use the appropriate RegistrationMethods method to print dinqOwners with the String header "Sorted list based on Registration date" (hint see step 6 above).
12. Use the appropriate RegistrationMethods method to create a new array of owners whose registration will expire within the next 3 months or less, but is not over one year old (ie 10-12 months old) called almostDue based on the array ltStateCopy. Remember what you did in Lab7 for FindAll.
13. Use the appropriate RegistrationMethods method to print dinqOwners with the String header "Owners with registration expiring in three months or less"

**Submitting your work**

* Ensure that all methods have **appropriate javadoc code**
* Post the following in a SINGLE zip folder (Citizen.java, CitizenInterface.java, CarOwner.java, CarOwnerInterface.java, RegistrationMethods.java, and RegistrationDemo.java, output file (ie out.txt)) and load into BB