CSD214 Lab1

/100

1. [Preface](#_b0gerbgej72s)
2. [Objectives](#_hwd3jhtma1nn)
3. [Deliverables](#_ym7gdpu9vn58)
4. [Assignment](#_szyyosdlkbsr)

# Preface

* **Place a header at the beginning of each program that looks like this**

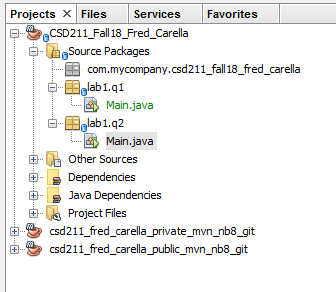
**/\*\***

**Author : Your Name**

**Date : the date**

**Description : describe what your program does.**

**\*\*/**

* **place your programs in packages.   
  put lab1 question 1 in package  
   lab2.q1  
  put lab1 question 2 in package  
   lab2.q2  
  etc…  
  If your program is in a class called Main.java then your packages should look like this…  
  **
* **commit and then push the programs to your git repo.**
* **place comments indicating the INPUT, PROCESS and OUTPUT sections of your program.**

# Objectives

* perform all the procedures outlined in the Assignment below

# Deliverables

* Complete the programming questions outlined below.  
  Then….
* in your IDE , commit your files to your local repository.   
  **Do this every time you modify code.**
* in your IDE , push all your code to your remote git repository (git/push).   
  **Do this frequently, after every coding session.**
* Answer any questions you are asked in a file called **Answers.docx**then upload Answers.docx to your dropbox.  
  **Do this when you complete answering the questions.**
* Upload a file called README to your LMS dropbox with the following information in it (a plain text file will work).   
    
  Name: your name  
  Date: the date  
  Repo: <https://gitlab.com>/<the url of your project. You get this from gitlab.com>|  
    
  **Do this when you’re done and the lab is ready for marking.**
* When you upload the README file to LMS dropbox, consider your lab handed in. Your teacher will be notified that the lab is submitted and that its ready for marking. Teacher marks it, and the mark automagically appears in your LMS Marks.  
    
  **Do not email me a message or a zip file of the project, after all we are in the 21’st century now :)**

# Assignment

For all questions, structure your code using the application skeleton:

Main.java

App.java

Put your code in the run method of the App.java class

In Main.java call "new App.run();"

1. Create a package called lab1 and put all of your code in there.  
   Create a package for each question for example:  
   lab1.q1  
   lab1.q2  
   etc…
2. Make sure you comment your code, comments will be marked.
3. Complete the Person exercise we discussed in class by doing the following
   1. Has the following attributes:  
      firstname,  
      lastname,  
      gender, (use enums for the gender)  
      SIN, (Social Insurance Number)  
      COUNT - a class variable which is incremented every time a Person is created and decremented every time a Person is deleted.
   2. Add constructors so that you can initialize a Person like this  
      // Joe Student is 23 years old and male and SIN=1234  
      Person p1=new Person("Joe", "Student",'M', 1234);  
      Person p3=new Person("Joe", "Student",1234);  
      Person p1=new Person("Joe", "Student");  
      System.out.println(p1);  
      System.out.println(p2);  
      System.out.println(p3);

**/5**

* 1. Add a toString method so it prints Person objects like this…  
     { firstname, lastname, SIN }  
     example;  
     { Joe Student, M, SIN=1234 }

**/5**

* 1. Override the equals and hashCode methods.  
     Two Person objects will be considered equal if all attributes are equal.

**/5**

* 1. Add setters and getters for all attributes

**/5**

1. Override the finalize method so it decrements COUNT. Verify with a Unit test.  
   Person p=new Person();

1. Create a Car class with the following attributes.   
   Note \* use appropriate types.   
   Note \* Use the Person class from q1 for Owner.  
    Make  
    Model  
    Year  
    Owner  
    VIN  
    COUNT - a class variable which is incremented every time a Person is created and  
    decremented every time a Person is deleted.  
     
     
   1. Create constructors so you can create cars like this:  
        
       Person shane=new Person("Shane", "Mollari");  
       Car ford=new Car("Ford", "Mustang", 2012);  
       Car chev=new Car("Chevrolet", "Camaro", 2012, shane);  
       Car ford=new Car("Ford", "Focus", 2013, 1234);  
        
      Where 1234 == the VIN of the car

**/5**

* 1. Add an appropriate toString method so you can print out each car. Make sure the full details of each owner get printed as well.

**/5**

* 1. Override the equals and hashCode methods. Two cars will be equal if their all attributes are equal.

**/5**

1. Write unit tests that will test that the “equals” methods for both Person and Car work.

Assume:  
Person p1 =new Person(“First”, “Last”, Person.Gender.M, 1234);   
Person p2 =new Person(“First”, “Last”, Person.Gender.M, 1234);   
Person p3 =new Person(“First”, “Last”, Person.Gender.M, 12345);   
Person p4=p1;  
  
Car c1 = new Car("Chevrolet", "Camaro", 2012, 1234, p1);  
Car c2 = new Car("Chevrolet", "Camaro", 2012, 1234, p1);  
Car c3 = new Car("Chevrolet", "Camaro", 2012, 12345, p1);  
Car c3 = c1;  
  
Where 1234 == the VIN of the car

* 1. Test that   
     p1.equals(p2) is TRUE,  
     p4.equals(p1) is TRUE,  
     p1.equals(p3) is FALSE,  
       
     c1.equals(c2) is TRUE,  
     c4.equals(c1) is TRUE,  
     c1.equals(c3) is FALSE,
  2. What would you expect p1 == p2 to evaluate to? They are the same but what is the output of the following code?  
     if(p1 == p2)  
      System.out.println(“equal”);  
     else  
      System.out.println(“not equal”);

**/15**

1. Write a "used car lot" application that is implemented with the following features.

**/50**

Your app will consist of the following classes  
  
l1.q4.Car.java - Use the Car class from Q4  
l1.q3.Person.java - Use the Person class from Q3  
l1.q6.App.java  
  
In App.java, include the following instance variables  
----------------  
 private final int numCars = 100;

private Car[] unsoldCars = new Car[numCars];

private Car[] soldCars = new Car[numCars];

private int unsoldCurrentIndex = 0;

private int soldCurrentIndex = 0;

private Scanner input = new Scanner(System.in);

private String menu = ""

+ "----------------\n"

+ " 1. Add a Car\n"

+ " 2. Edit a car\n"

+ " 3. Delete an unsold Car\n"

+ " 4. Delete a sold Car\n"

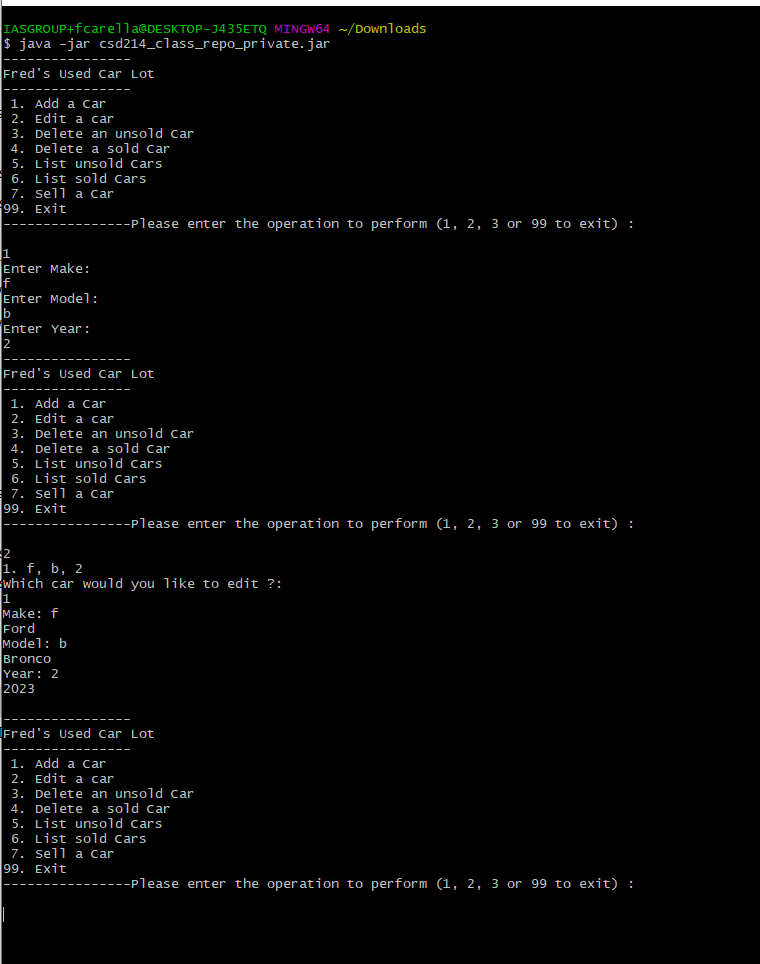
+ " 5. List unsold Cars\n"

+ " 6. List sold Cars\n"

+ " 7. Sell a Car\n"

+ "99. Exit\n"

* 1. ----------------  
     Main (the driver)  
     ----------------  
     tbd  
     ----------------  
     1. Your app will implement a CRUD (Create, Update, Delete) menu for adding, editing and deleting cars from the used car lot.
        + When a car is sold, you will indicate the new owner of the vehicle by adding a person attribute to the car.
        + Cars will be stored in an array data structure.
        + Add cars to the end of the array.
        + When a car is deleted, the array should be `"re-packed", meaning, for example, if the first car is deleted, all the other cars will be shifted to to the left so to speak. There will be no "empty" car spots between cars.
        + Use "null" to indicate an empty car spot (end of the cars).
     2. The app should run in a loop. See MenuLoop example of how to implement loops for this application. I will go over this in the lecture…  
          
        Download and run this app for an example of how it should run (note, its not complete, the sold functions arent implemented, Also, your app and be designed differently as long as the functions are implemented):  
        [Used car lot app...](https://drive.google.com/file/d/1WrTPFMNHG8R6eRIQt2iYgstSiYvKfZbX/view?usp=sharing)   
          
        Download it, open a shell and then run it like this:  
          
        java -jar csd214\_class\_repo\_private.jar



[If you’re curious how I made the app, read this…](https://www.jetbrains.com/idea/guide/tutorials/hello-world/packaging-the-application/)

1. ~~Add a GUI to your app. I’ll discuss this further in class.~~

See notes below…

NOTES\*\*

Example code to edit a car: accepts user input, if user hits enter, it uses the default. See getInput(String s) and getInput(int i)...

private void edit() {

list(1);

System.out.println("Which car would you like to edit ?:");

int choice = input.nextInt();

input = new Scanner(System.in); // reset the scanner

if ((choice < currentIndex + 1) && choice > 0) {

Car c = cars[choice - 1];

System.out.println("Make: " + c.getMake());

c.setMake(getInput(c.getMake()));

System.out.println("Model: " + c.getModel());

c.setModel(getInput(c.getModel()));

System.out.println("Year: " + c.getYear());

c.setYear(getInput(c.getYear()));

} else {

System.out.println("Choice out of bounds");

}

System.out.println("");

}

private String getInput(String s) {

String ss = input.nextLine();

if (ss.trim().isEmpty()) {

return s;

}

Scanner in2 = new Scanner(ss);

return in2.nextLine();

}

private int getInput(int i) {

String s = input.nextLine();

if (s.trim().isEmpty()) {

return i;

}

Scanner in2 = new Scanner(s);

return in2.nextInt();

}

private double getInput(double i) {

String s = input.nextLine();

if (s.trim().isEmpty()) {

return i;

}

Scanner in2 = new Scanner(s);

return in2.nextDouble();

}