Week4Lab – 10 pts

Pre-lab questions

1. The best way to get comfortable with loops is to write them! Write just the loops (and any required, dependent variables) to do the following:
   1. Count down from 10 to Blast Off!, printing each number on its own line.

int second = 10;

while(second >= 0){

if (second == 0){

System.out.println("Blast off");

}

else {

System.out.println(second);

}

second--;

}

* 1. Replace the lowercase letter e from each user entered String, replacing it with an underscore and printing the new String. Allow the user to enter as many Strings as they want. Choose a good sentinel value.

boolean done = false;

while (! done){

System.out.print("Enter any word and enter 0 when finished: ");

String str = scan.nextLine();

if (str.equals("0")){

done = true;

}

else{

System.out.println(str.replace('e','\_'));

}

}

Count up by a given number from 0 to another given number. For example the user may enter 5 and 30 and the loop should produce: 5, 10, 15, 20, 25, 30 then stop.

Scanner scan = new Scanner(System.in);

System.out.print("Enter the number: ");

int num = scan.nextInt();

for (int i = num; i <= 30; i += 5){

System.out.println(i);

}

}

* 1. Rewrite the while loop in a. to a for loop.

for (int second = 10; second >= 0; second--){

if (second == 0) {

System.out.println("Blast Off");

}

else {

System.out.println(second);

}

}

* 1. Write a for loop that adds the even numbers from 0 to 100, then prints the final result only.

int sum = 0;

for (int num = 0; num <= 100; num += 2){

sum += num;

}

System.out.println(sum);

Choose one of the following do develop into a program that uses at least one class outside of the driver class that contains the main method. Your solution MUST use an ArrayList of objects created from your custom class. Once chosen, do the following:

Understand the problem (restate in your own words, make any assumptions clear):  
  
UML diagrams of any classes needed, including the one with main:  
  
Pseudocode of each class:  
  
Name of files (.java) submitted:  
  
Screenshots of at least three separate runs of your program with test data:

Garage and Cars

Write a Car class that includes key ways we describe cars. It has to include either price or value (it can include both). Be sure to determine the getters and setters that need to be written, and write the toString() method appropriately so when you ‘print’ a car, it makes sense.

The driver class will contain a garage (ArrayList) that contains several cars. The user must be able to add a car to the garage, print the contents of the garage, get the value of the cars in the garage, remove a car from the garage (like selling it).

We need a car class that has a price a value and other attributes describing cars. The class should behave in such a way so that a car can have either a price, a value, or both. It should have a method to add the car to a garage (an Arraylist) and another method to remove one. All attributes will have a getter and a setter.

**Car**

Plate\_number: String

Name: String

Price: double

Value: int

Year: int

Getters and setters:

Name()->String

Price()->double

Value()->int

Plate()->String

Year()->int

addtoGarage(list: garage)->void

removeFromGarage(list: garage) -> void

toString()->void

Car.java

CarGarage.java

**Car Class Pseudocode:**

class Car:

def Car(String plate, String name, double price, int value ,int year){

this.plate = plate;

this.name = name;

this.price = price;

this.value = value;

this.year = year;

}

// Getters and setters will continue for all attributes

def getName() {

return name;

}

…

def setYear(int year) {

this.year = year;

}

addtoGarage(garage) {

printf("%s added to garage\n", this);

garage.add(this);

}

def removefromGarage(garage) {

System.out.printf("%s removed from garage\n", this);

garage.remove(this);

}

def toString(){

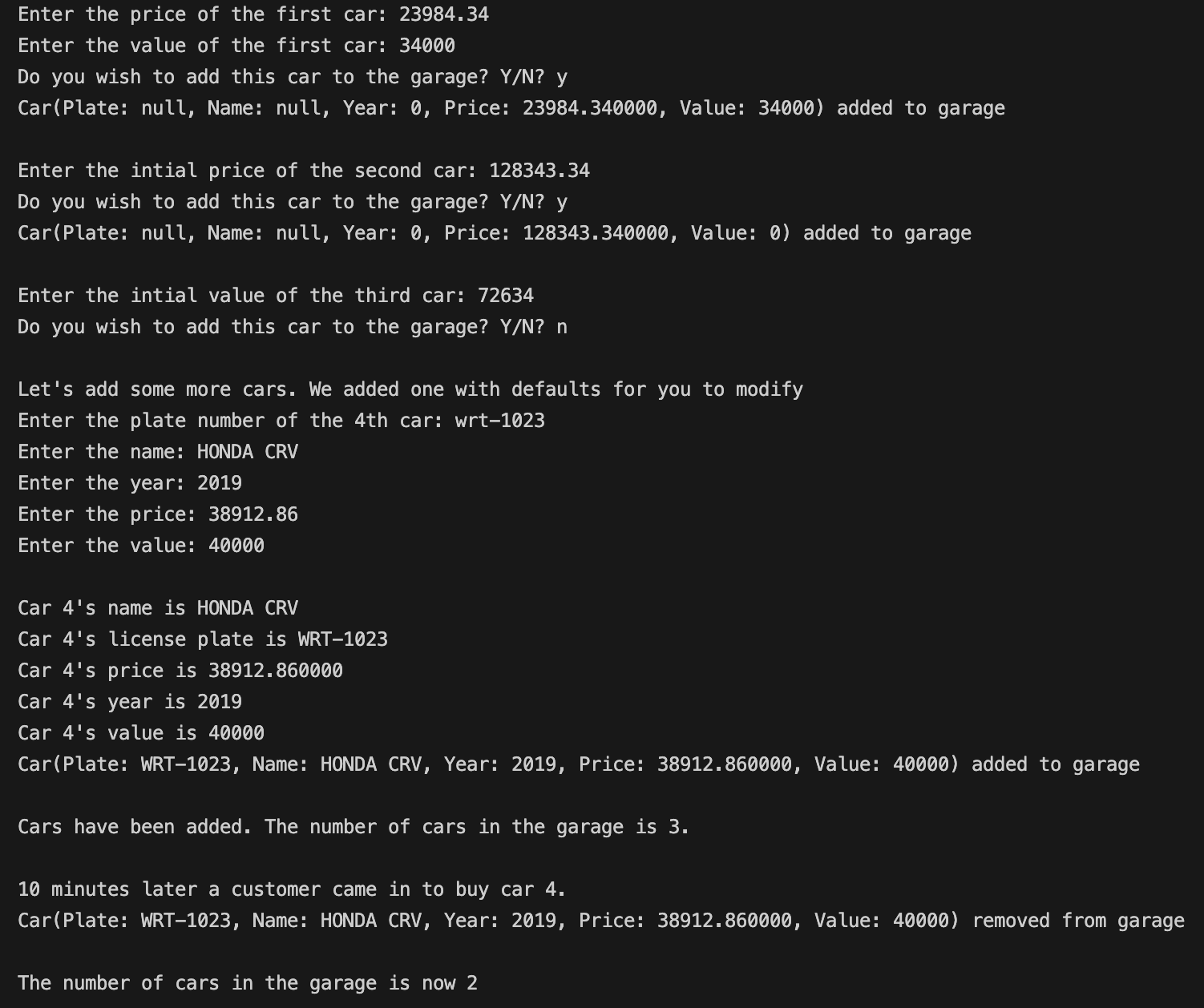
String info = f"Car(Plate: %s, Name: %s, Year: %d, Price: %f, Value: %d)",

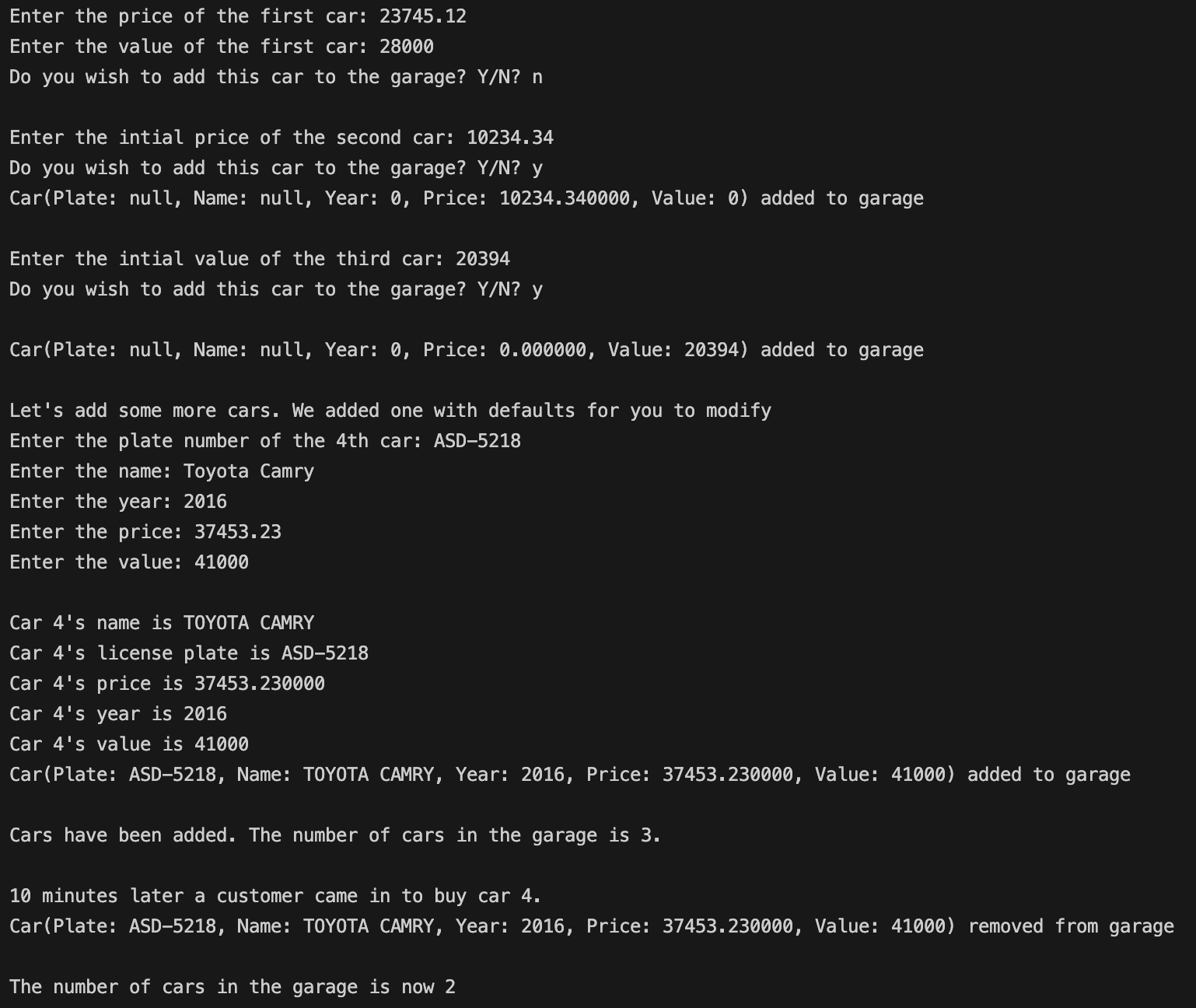
this.plate, this.name, this.year, this.price, this.value);

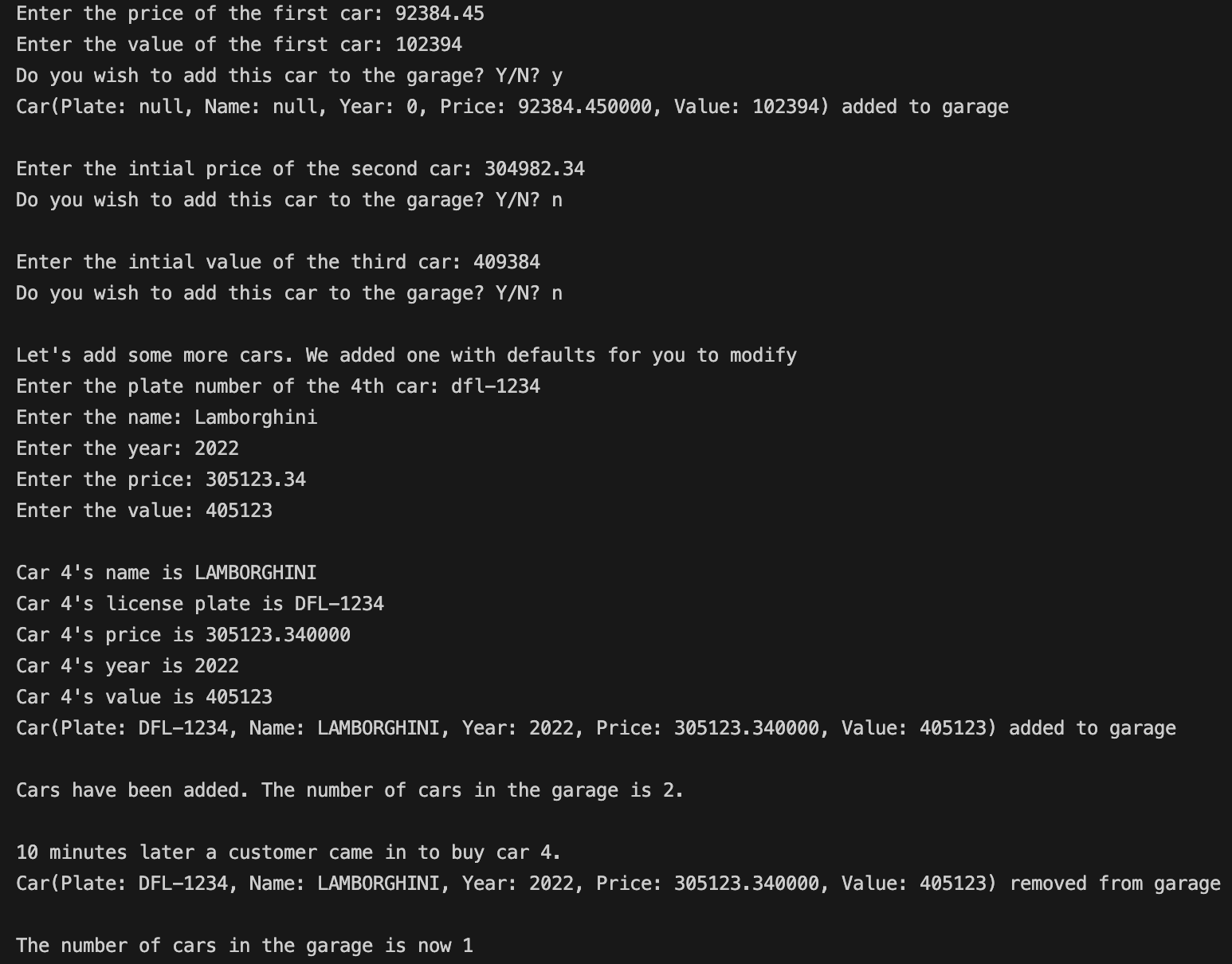
return info;

}

}









Airplane Seats

Write a Seat class that contains information on a given seat on an airplane. The seat is available or sold. If it is sold, it knows the confirmation number. It always knows its value (or price) and its type. You get to name the types, but make sure that “desk” is one of them! Determine the getters and setters and any other methods you need, but be sure to write toString() because you will be printing out the seats.

The driver class will contain an ArrayList of all the seats. Use a loop or loops to generate at least 100 seats in total. The user must be able to see all the seats that are available, and all that are available of a certain type (like “desk”), choose one getting a confirmation number, then be able to view all seats with that confirmation number.

Cooking and Calories

Write an ingredient class so that a given ingredient knows its unit and calories, such as 100 calories and 1 cup, as well as its name and type, such as protein, carb or fat.

The driver class for this will allow the user to create ingredients from a recipe, see them as a list, one ingredient per line, formatted as expected for a recipe. They will also be able to get the total calories for the whole recipe and be able to enter the number of servings and get the calories per serving. The user should also be able to get a list of ingredients by type, ex. printing only the protein ingredients.