Reflection:

During this class, I learned several valuable lessons. In particular, I learned the importance of properly instantiating arrays and using comments effectively.

In the begging of the class, I did not see the use of comments as heavily as I do now. After receiving feedback from my instructor, I began commenting in increasing amounts, and realized that using comments is essential for keeping track of complex code, particularly when working on larger projects.

Furthermore, in Module 5, I encountered issues with properly instantiating arrays. The instructor feedback advised me that the array list wasn't instantiated properly, which was a confusing comment. After reviewing my code, I realized that I was using basic arrays instead of using ArrayList. I will be checking requirements carefully and verifying they are operating as required in the future.

Additionally, I learned that it is crucial to test my code thoroughly before submitting it. In Module 5, I failed to include the functionality to print the temperature of each day plus the week's average temperature if the user input 'week'. I had only included the functionality to get the temperature for a given day. Thorough testing and checking requirements will help in avoiding this issue in the future.

Overall, this class was an good experience that has helped me improve my skills. I look forward to applying these lessons to future projects and continuing to grow as a programmer.

Module 1: Corrections – added comments for clarification

**package** CSC320\_Pkg;

**public** **class** CriticalThinkingMod1 {

**public** **static** **void** main(String[] args) {

/\*

\* PSUDO CODE:

\*/

/\*Setup string variables for:

\* First name - Michael

\* Last name - Maye

\* Street address - 123 Main Street

\* City - SomewhereUSA

\* Zipcode - 12345

\*/

/\*Setup println statements that print to new lines for variables:

\* First name

\* Last name

\* Street address

\* City

\* Zipcode

\*/

//SOURCE CODE:

//Initialize variables

String firstName = "Michael";

String lastName = "Maye";

String streetAddress = "123 Main Street";

String city = "SomewhereUSA";

String zipCode = "12345";

//Print out information to user

System.***out***.println(firstName);

System.***out***.println(lastName);

System.***out***.println(streetAddress);

System.***out***.println(city);

System.***out***.println(zipCode);

}

}

Module 2: Corrections: Added comments for clarification

**package** CSC320\_Pkg;

**import** java.util.Scanner;

**public** **class** CricitalThinkingMod2 {

/\*

\* Import java scanner and java math

Set variables:

String for Car Manufacturer

String for Car Model

String for Car year

Long for Odometer Start

Long for Odometer End

Double for Gallons Used

Double for Miles PerGallon

Print statement for input and setting in put:

“Enter car manufacturer:”

Set input to string for Car Manufacturer

“Enter Car Model:”

Set input to string for Car Model

“Enter Car Year:”

Set input to string for Car Model

“Enter starting odometer reading:”

Set input to long for Odometer Start

“Enter ending odometer reading:”

Set input to long for Odometer End

“Enter gallons of gas used:”

Set input to double for Gallons Used

Calculate Miles per gallon:

-Calculate miles driven

-Calculate miles per gallon

Print info to user:

\*

\*

\*/

**public** **static** **void** main(String[] args) {

//Initialize scanner and variables

Scanner scan = **new** Scanner(System.***in***);

String carMan;

String carModel;

String carYear;

**double** odStart;

**double** odEnd;

**double** milesDriven;

**double** galUsed;

**double** mpg;

//Request car info from user

System.***out***.println("Enter your car's year, manufacturer, and model:\nExample: 2015 Chevrolet Volt");

carYear = scan.next();

carMan = scan.next();

carModel = scan.next();

//Request odometer readings from user

System.***out***.println("Enter the starting and ending odometer readings:\nExample: 1000.0 1500.0");

odStart = scan.nextDouble();

odEnd = scan.nextDouble();

//Request gallons of gas used from user

System.***out***.println("Enter the gallons used during drive:\nExample: 25.2");

galUsed = scan.nextDouble();

scan.close();

//Calculate miles driven from odometer readings

milesDriven = odEnd - odStart;

//Calculate miles driven from miles driven and gallons of gas used

mpg = milesDriven / galUsed;

//Print car information

System.***out***.printf("Car: %s %s %s\n", carYear, carMan, carModel);

//Print miles driven, gallons used, and mpg

System.***out***.printf("Miles Driven: %.2f\nGallons Used: %.2f\nMPG: %.2f", milesDriven, galUsed, mpg);

}

}

Module 3: Correction – Added comments for clarification

**package** CSC320\_Pkg;

**import** java.util.Scanner;

**public** **class** CriticalThinkingMod3 {

**public** **static** **void** main(String[] args) {

/\*

\* Declare variable for weekly income, weekly tax withholding and tax rate:

\* Double weekIncome;

\* Double taxRate;

\* Double weekTaxWithhold;

\*

\* If else statement to set the tax level:

\* if weekIncome is less than $500: taxRate equals 0.10

\* else if weekIncome is less than $1500: taxRate equals 0.15

\* else if weekIncome is less than $2500: taxRate equals 0.20

\* else if weekIncome is greater than or equal to $2500: taxRate equals 0.30

\*

\* Calculate weekly tax withholding:

\* weekTaxWithhold equals weekIncome \* taxRate

\*

\* Display weekTaxWithold:

\* Print weekTaxWithold

\*/

//Initialized required variables

**double** weekIncome;

**double** taxRate = 0;

**double** weekTaxWithhold;

**boolean** input = **false**;

//Initialized scanner as scnr

Scanner scnr = **new** Scanner(System.***in***);

//Print out request to user for weekly income

System.***out***.println("Enter weekly income");

//\*\*\*Added do/while loop to catch invalid input from user\*\*\*

**do** {

**if** (scnr.hasNextDouble()) {

input = **true**;

}

**else** {

System.***out***.println("Invalid Input");

scnr.next();

}

}**while** (!input);

//Set weekIncome to input from user

weekIncome = scnr.nextDouble();

//If statement to use user input to set proper tax rate

**if** (weekIncome < 500.00) {

taxRate = 0.10;

} **else** **if** (weekIncome < 1500.00) {

taxRate = 0.15;

} **else** **if** (weekIncome < 2500.00) {

taxRate = 0.20;

} **else** **if** (weekIncome >= 2500.00) {

taxRate = 0.30;

}

// \*\*\*Closed scnr to avoid memory leaks\*\*\*

scnr.close();

//Set weekTaxWithold to weekly income times tax rate

weekTaxWithhold = weekIncome \* taxRate;

//Print out weekly tax rate to user

System.***out***.printf("Your weekly tax withholding is %.2f\n", weekTaxWithhold);

}

}

Module 4: Correction – added comments for clarification

package CSC320\_Pkg;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class CriticalThinkingMod4 {

public static void main(String[] args) {

/\*

\* Write a program that utilizes a while-loop to read a set of five floating-point values from user input.

\* Include code to prevent an endless loop. Ask the user to enter the values, then print the following data:

\*

\* Create variables for:

\* List userInput

\* Int variable

\* Double total

\* Double average

\* Double maximum

\* Double minimum

\* Double interestTot

\*

\* While loop that asks user for input to add floating-point values to list:

\*

\* while user input is not 'done':

\* add user input to userInput

\* ask user for another input

\* add 1 to variable

\*

\*

\* While loop that takes all user input and calculates required outputs:

\*

\* set variable to zero

\*

\* while variable is less than userInput length:

\* add float at userInput[variable] to total

\* if float at userInput[variable] is larger than maximum:

\* maximum equals userInput[variable]

\* if float at userInput[variable is less than minimum:

\* minimum equals userInput[variable]

\*

\* Calculate average and interest total at 20%:

\*

\* average equals total divided by length of user input

\*

\* interestTot equals total plus total times .20

\*

\* Print required output

\*/

// create scanner and list

Scanner scnr = new Scanner(System.in);

List<Double> userInputList = new ArrayList<Double>();

// Create variables

int i;

Double input;

String check = null;

Double total = 0.0;

Double average = 0.0;

Double maximum = 0.0;

Double minimum;

Double interestTot = 0.0;

// Initial request for input

System.out.println("Enter the ammounts:");

// Check if input is of Double type

scnr.hasNextDouble();

// While loop, stops if 'check' variable is equal to "done"

while (check != "done") {

// check if user has entered input (used to manage if statement during while

// loop)

if (scnr.hasNext()) {

// if user entered Double then add input to end of user input list and ask for

// next input

if (scnr.hasNextDouble()) {

input = scnr.nextDouble();

userInputList.add(input);

System.out.println("Enter next ammount, or 'done' if no more entries.");

scnr.nextLine();

}

// if user entered "done" set check to "done" and add another line

else if (scnr.hasNext("done")) {

check = "done";

System.out.print("");

}

// if is not a double or "done", print invalid statement and move to next input

// line

else if (scnr.hasNext()) {

System.out.println("Invalid input, please try again");

scnr.nextLine();

continue;

}

}

}

// \*\*\*Closed scnr to avoid memory leaks\*\*\*

scnr.close();

// set variable for while loop

i = 0;

// set initial minimum value to handle while loop minimum calculation

minimum = userInputList.get(0);

// while loop stops if i is greater than or equal to user input list length

while (i < userInputList.size()) {

total += userInputList.get(i);

// if user input at 'i' is greater than current maximum, replace

if (userInputList.get(i) > maximum) {

maximum = userInputList.get(i);

}

// if user input at 'i' is less than current minimum, replace

if (userInputList.get(i) < minimum) {

minimum = userInputList.get(i);

}

// increment i

i++;

}

// calculate total interest at %20

interestTot = total + (total \* 0.20);

// calculate average by dividing total by user input list length

average = total / userInputList.size();

// print required outputs

System.out.printf("Your total is %.2f\n", total);

System.out.printf("The average is %.2f\n", average);

System.out.printf("The minimum is %.2f\n", minimum);

System.out.printf("The maximum is %.2f\n", maximum);

System.out.printf("The Total with 20 percent interest is %.2f\n", interestTot);

}

}

Module 5: Corrections – added functionality for week input to print each day and weekly avg, added comments for clarification

**package** CSC320\_Pkg;

**import** java.util.Scanner;

**public** **class** CriticalThinkingMod5 {

**public** **static** **void** main(String[] args) {

/\*

\* Import Scanner

\*

\* Create Variables

\*

\* Prompt user for day of week

\*

\* Get index of day in Days array

\*

\* Return temp at day index

\*

\* Print day and temp to user

\*/

// Setup Variables

String[] days = **new** String[] { "sunday", "monday", "tuesday", "wednesday", "thursday", "friday", "saturday" };

**double**[] week\_temp = **new** **double**[] { 87.2, 74.5, 81.4, 76.5, 86.5, 83.9, 79.2 };

**int** k = 0;

**int** i = 0;

**boolean** input = **false**;

String userDay;

**double** weeklyAvg;

**double** total = 0;

// Create scanner object

Scanner scnr = **new** Scanner(System.***in***);

// Ask for user input

System.***out***.println("Enter a day of the week");

userDay = scnr.next();

scnr.close();

// For loop to iterate over array

**for** (i = 0; i <= days.length - 1; i++) {

total += week\_temp[i];

// Check if user input matches a day, set input to true if match is found, set

// day to index of match

**if** (userDay.toLowerCase().compareTo(days[i].toLowerCase()) == 0) {

input = **true**;

k = i;

}

}

weeklyAvg = 610;

weeklyAvg = total / 7;

//Check if user entered 'week'

**if** (userDay.compareToIgnoreCase("week") == 0) {

//Print weeks temps and week average

System.***out***.printf("Sun: %.2f, Mon: %.2f, Tues: %.2f, Wed: %.2f, Thurs: %.2f, Fri: %.2f, Sat: %.2f\n",

week\_temp[0], week\_temp[1], week\_temp[2], week\_temp[3], week\_temp[4], week\_temp[5], week\_temp[6]);

System.***out***.printf("Weekly average: %.2f", weeklyAvg);

}

// Check if user input is valid with 'input' bool

**else** **if** (input) {

// Print out day and temp, print Invalid Input if no match from user entry

System.***out***.printf("The temperature on %s was %.2f\n", days[k], week\_temp[k]);

} **else** {

//Print invalid input error

System.***out***.print("Invalid Input");

}

}

}

Module 6:

**package** CSC320\_Pkg;

**import** java.util.\*;

**public** **class** CriticalThinkingMod6 {

/\* Create new scan object

\*

\* Call method to get input and print to screen

\* Print "Success" or "Failure" returned from method

\*

\*

\*

\* Create method to get user input:

\*

\* public static String method (scanner object)

\*

\* Create array to store user inputs as strings

\* Create integer i to track while loop iteration

\*

\* Create string to hold user input line

\*

\* While loop to iterate over the length of input line split with split(" ")

\*

\* Try catch to enter split user data in array at array[i]

\*

\* Try stores user input split into array and increments i

\*

\* Catch advises user to only enter three inputs and returns "Failure" to calling method

\*

\* Print statement to print user inputs

\* return "Success" to calling method

\*

\*/

**public** **static** **void** main(String[] args) {

//Create new scanner object

Scanner scnr = **new** Scanner(System.***in***);

//call method and print what it returns

System.***out***.println(*getInput*(scnr));

}

**public** **static** String getInput(Scanner scanObj) {

//create new string array to hold user input and integer for while loop

String[] userInput = **new** String[3];

**int** i = 0;

//request input from user

System.***out***.println("Please enter three separate inputs. i.e. 'word' '1234' '#@#$%'");

String input = scanObj.nextLine();

//while loop to iterate over line entered by user, place at least 3 (at most the amount entered by user) into array

**while** (i < 3 || i < input.split(" ").length) {

//try catch to enter user input strings into array. will catch if user entered less than or more than 3 input strings

**try** {

userInput[i] = input.split(" ")[i];

i++;

}

**catch** (Exception e) {

System.***out***.println("Invalid, please enter three valid inputs");

//returns "Failure" to calling method

**return** "Failure";

}

}

//Prints user input strings and returns "Success" to calling method for printing to console

System.***out***.printf("Your input was %s, %s, and %s.\n", userInput[0], userInput[1], userInput[2]);

**return** "Success";

}

}

Portfolio Project:

Source Code:

Automobile Class:

**package** CSC320\_Pkg;

**public** **class** AutoInventory {

String make;

String model;

String year;

String color;

String cond;

String vin;

**double** mileage;

**double** msrp;

**double** dealerPur;

**double** price;

// default constructor

**public** AutoInventory() {

make = "";

model = "";

year = "";

color = "";

cond = "";

vin = "";

mileage = 0.0;

msrp = 0.0;

dealerPur = 0.0;

price = 0.0;

}

// parameterized constructor

**public** AutoInventory(String make, String model, String year, String color, String cond, String vin, **double** mileage,

**double** msrp, **double** dealerPur, **double** price) {

**this**.make = make;

**this**.model = model;

**this**.year = year;

**this**.color = color;

**this**.cond = cond;

**this**.vin = vin;

**this**.mileage = mileage;

**this**.msrp = msrp;

**this**.dealerPur = dealerPur;

**this**.price = price;

}

// method to add a new vehicle

**public** String addNewVehicle(String make, String model, String year, String color, String cond, String vin,

**double** mileage, **double** msrp, **double** dealerPur, **double** price) {

**try** {

**this**.make = make;

**this**.model = model;

**this**.year = year;

**this**.color = color;

**this**.cond = cond;

**this**.vin = vin;

**this**.mileage = mileage;

**this**.msrp = msrp;

**this**.dealerPur = dealerPur;

**this**.price = price;

**return** "Vehicle added successfully";

} **catch** (Exception e) {

**return** "Failed to add vehicle";

}

}

// method to list vehicle information

**public** String[] listVehicleInformation() {

**try** {

String[] vehicleInfo = **new** String[10];

vehicleInfo[0] = **this**.make;

vehicleInfo[1] = **this**.model;

vehicleInfo[2] = **this**.year;

vehicleInfo[3] = **this**.color;

vehicleInfo[4] = **this**.cond;

vehicleInfo[5] = String.*valueOf*(**this**.vin);

vehicleInfo[6] = String.*valueOf*(**this**.mileage);

vehicleInfo[7] = String.*valueOf*(**this**.msrp);

vehicleInfo[8] = String.*valueOf*(**this**.dealerPur);

vehicleInfo[9] = String.*valueOf*(**this**.price);

**return** vehicleInfo;

} **catch** (Exception e) {

**return** **null**;

}

}

// method to remove a vehicle

**public** String removeVehicle() {

**try** {

**this**.make = "";

**this**.model = "";

**this**.year = "";

**this**.color = "";

**this**.cond = "";

**this**.vin = "";

**this**.mileage = 0.0;

**this**.msrp = 0.0;

**this**.dealerPur = 0.0;

**this**.price = 0.0;

**return** "Vehicle removed successfully";

} **catch** (Exception e) {

**return** "Failed to remove vehicle";

}

}

// method to update vehicle attributes

**public** String updateVehicleAttributes(String make, String model, String year, String color, String cond, String vin,

**double** mileage, **double** msrp, **double** dealerPur, **double** price) {

**try** {

**this**.make = make;

**this**.model = model;

**this**.year = year;

**this**.color = color;

**this**.cond = cond;

**this**.vin = vin;

**this**.mileage = mileage;

**this**.msrp = msrp;

**this**.dealerPur = dealerPur;

**this**.price = price;

**return** "Vehicle attributes updated successfully";

} **catch** (Exception e) {

**return** "Failed to update vehicle attributes";

}

}

//Get methods for car properties

**public** String getMake() {

**return** **this**.make;

}

**public** String getModel() {

**return** **this**.model;

}

**public** String getYear() {

**return** **this**.year;

}

**public** String getVin() {

**return** **this**.vin;

}

**public** String getColor() {

**return** **this**.color;

}

**public** String getCond() {

**return** **this**.cond;

}

**public** **double** getMileage() {

**return** **this**.mileage;

}

**public** **double** getMsrp() {

**return** **this**.msrp;

}

**public** **double** getDealerPur() {

**return** **this**.dealerPur;

}

**public** **double** getPrice() {

**return** **this**.price;

}

//Set methods for car properties

**public** **void** setMake(String make) {

**this**.make = make;

}

**public** **void** setModel(String model) {

**this**.model = model;

}

**public** **void** setYear(String year) {

**this**.year = year;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **void** setCond(String cond) {

**this**.cond = cond;

}

**public** **void** setVin(String vin) {

**this**.vin = vin;

}

**public** **void** setMileage(**double** mileage) {

**this**.mileage = mileage;

}

**public** **void** setMsrp(**double** msrp) {

**this**.msrp = msrp;

}

**public** **void** setDealerPur(**double** dealerPur) {

**this**.dealerPur = dealerPur;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

}

Dealership Inventory Class:

**package** CSC320\_Pkg;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** DealershipSystem {

// Set static array list to contain cars

**static** ArrayList<AutoInventory> *inventory* = **new** ArrayList<AutoInventory>();

**static** Scanner *scnr* = **new** Scanner(System.***in***);

**static** **boolean** *exit* = **false**;

**public** **static** **void** main(String[] args) {

System.***out***.println("Welcome to the dealer inventory system.");

**while** (!*exit*) {

System.***out***.println("Add, Update, Remove, Inventory, Print or quit?");

**if** (*scnr*.hasNext()) {

String userChoice = *scnr*.next();

*scnr*.nextLine(); // consume newline character in the buffer

**if** (userChoice.toLowerCase().compareTo("add") == 0) {

*addNewVehicle*();

} **else** **if** (userChoice.toLowerCase().compareTo("inventory") == 0) {

*printInventory*();

} **else** **if** (userChoice.toLowerCase().compareTo("quit") == 0) {

System.***out***.println("Goodbye");

*exit* = **true**;

} **else** **if** (userChoice.toLowerCase().compareTo("update") == 0) {

*updateVehicle*();

}

**else** **if** (userChoice.toLowerCase().compareTo("print") == 0) {

*printInventoryToFile*();

}

**else** **if** (userChoice.toLowerCase().compareTo("remove") == 0) {

*removeVehicle*();

}

**else** {

System.***out***.println("Invalid Input");

}

}

}

}

**public** **static** **void** addNewVehicle() {

String make;

String model;

String year;

String color;

String cond;

String vin;

**double** mileage;

**double** msrp;

**double** dealerPur;

**double** price;

System.***out***.println("Please enter vehicle VIN, Make, Model, and Year:");

String[] userInput = **new** String[4];

**int** i = 0;

String input = *scnr*.nextLine();

**while** (i < 4 || i < input.split(" ").length) {

**try** {

userInput[i] = input.split(" ")[i];

i++;

} **catch** (Exception e) {

System.***out***.println("Invalid input");

**return**;

}

}

make = userInput[1];

model = userInput[2];

year = userInput[3];

vin = userInput[0];

System.***out***.println("Please enter vehicle color:");

color = *scnr*.nextLine();

System.***out***.println("Please enter vehicle condition:");

cond = *scnr*.nextLine();

System.***out***.println("Please enter vehicle mileage:");

mileage = *scnr*.nextDouble();

System.***out***.println("Please enter vehicle MSRP:");

msrp = *scnr*.nextDouble();

System.***out***.println("Please enter vehicle dealer purchase price:");

dealerPur = *scnr*.nextDouble();

System.***out***.println("Please enter vehicle price:");

price = *scnr*.nextDouble();

AutoInventory car = **new** AutoInventory(make, model, year, color, cond, vin, mileage, msrp, dealerPur, price);

*inventory*.add(car);

System.***out***.println("Vehicle added:");

System.***out***.printf(

"Make: %s, Model: %s, Year: %s, Color: %s, Condition: %s, VIN: %s, Mileage: %.2f, MSRP: %.2f, Dealer Purchase Price: %.2f, Price: %.2f\n",

car.getMake(), car.getModel(), car.getYear(), car.getColor(), car.getCond(), car.getVin(),

car.getMileage(), car.getMsrp(), car.getDealerPur(), car.getPrice());

}

**public** **static** **void** removeVehicle() {

System.***out***.println("Enter the VIN of the vehicle you want to update: ");

String vin = *scnr*.nextLine();

AutoInventory car = **null**;

**for** (AutoInventory a : *inventory*) {

**if** (a.getVin().equals(vin)) {

*inventory*.indexOf(a);

*inventory*.remove(a);

**break**;

}

}

**if** (car == **null**) {

System.***out***.println("Vehicle not found");

**return**;

}

}

**public** **static** **void** updateVehicle() {

System.***out***.println("Enter the VIN of the vehicle you want to update: ");

String vin = *scnr*.nextLine();

AutoInventory car = **null**;

**for** (AutoInventory a : *inventory*) {

**if** (a.getVin().equals(vin)) {

car = a;

**break**;

}

}

**if** (car == **null**) {

System.***out***.println("Vehicle not found");

**return**;

}

System.***out***.println("Enter new Vin (leave blank to keep current value):");

String newVin = *scnr*.nextLine();

**if** (!newVin.isEmpty()) {

car.setVin(newVin);

}

System.***out***.println("Enter new make (leave blank to keep current value):");

String make = *scnr*.nextLine();

**if** (!make.isEmpty()) {

car.setMake(make);

}

System.***out***.println("Enter new model (leave blank to keep current value):");

String model = *scnr*.nextLine();

**if** (!model.isEmpty()) {

car.setModel(model);

}

System.***out***.println("Enter new year (leave blank to keep current value):");

String year = *scnr*.nextLine();

**if** (!year.isEmpty()) {

car.setYear(year);

}

System.***out***.println("Enter new color (leave blank to keep current value):");

String color = *scnr*.nextLine();

**if** (!color.isEmpty()) {

car.setColor(color);

}

System.***out***.println("Enter new condition (leave blank to keep current value):");

String cond = *scnr*.nextLine();

**if** (!cond.isEmpty()) {

car.setCond(cond);

}

System.***out***.println("Enter new mileage (leave blank to keep current value):");

String mileageStr = *scnr*.nextLine();

**if** (!mileageStr.isEmpty()) {

**try** {

**double** mileage = Double.*parseDouble*(mileageStr);

car.setMileage(mileage);

} **catch** (NumberFormatException e) {

System.***out***.println("Invalid input for mileage");

}

}

System.***out***.println("Enter new MSRP (leave blank to keep current value):");

String msrpStr = *scnr*.nextLine();

**if** (!msrpStr.isEmpty()) {

**try** {

**double** msrp = Double.*parseDouble*(msrpStr);

car.setMsrp(msrp);

} **catch** (NumberFormatException e) {

System.***out***.println("Invalid input for MSRP");

}

}

System.***out***.println("Enter new dealer purchase price (leave blank to keep current value):");

String dealerPurStr = *scnr*.nextLine();

**if** (!dealerPurStr.isEmpty()) {

**try** {

**double** dealerPur = Double.*parseDouble*(dealerPurStr);

car.setDealerPur(dealerPur);

} **catch** (NumberFormatException e) {

System.***out***.println("Invalid input for dealer purchase price");

}

}

System.***out***.println("Enter new price (leave blank to keep current value):");

String priceStr = *scnr*.nextLine();

**if** (!priceStr.isEmpty()) {

**try** {

**double** price = Double.*parseDouble*(priceStr);

car.setPrice(price);

} **catch** (NumberFormatException e) {

System.***out***.println("Invalid input for price");

}

}

System.***out***.println("Vehicle updated:");

**for** (AutoInventory a : *inventory*) {

**if** (a.getVin().equals(newVin)) {

car = a;

**break**;

}

}

System.***out***.printf(

"Make: %s, Model: %s, Year: %s, Color: %s, Condition: %s, VIN: %s, Mileage: %.2f, MSRP: %.2f, Dealer Purchase Price: %.2f, Price: %.2f\n",

car.getMake(), car.getModel(), car.getYear(), car.getColor(), car.getCond(), car.getVin(),

car.getMileage(), car.getMsrp(), car.getDealerPur(), car.getPrice());

*printFilePrompt*(car);

}

**public** **static** **void** printInventory() {

System.***out***.println("Current inventory:\n");

System.***out***.printf("%-10s %-10s %-10s %-17s %-10s %-10s %-10s%n", "Make", "Model", "Year", "VIN", "Condition",

"Mileage", "Price");

System.***out***.println("------------------------------------------------------------------");

**for** (AutoInventory car : *inventory*) {

System.***out***.printf("%-10s %-10s %-10s %-17s %-10s %-10.1f $%-10.2f%n", car.getMake(), car.getModel(),

car.getYear(), car.getVin(), car.getCond(), car.getMileage(), car.getPrice());

}

}

**public** **static** **void** printFilePrompt(AutoInventory car) {

System.***out***.println("Do you want to print the information to a file? (y/n)");

String printToFile = *scnr*.nextLine();

**if** (printToFile.equalsIgnoreCase("y")) {

System.***out***.println("Do you want to print just the added car or the whole inventory? (a/i)");

String printOption = *scnr*.nextLine();

**if** (printOption.equalsIgnoreCase("a")) {

*printToFile*(car);

} **else** **if** (printOption.equalsIgnoreCase("i")) {

*printInventoryToFile*();

}

}

}

**public** **static** **void** printToFile(AutoInventory car) {

**try** {

FileWriter writer = **new** FileWriter("car.txt");

writer.write(car.toString());

writer.close();

System.***out***.println("Car information has been printed to file.");

} **catch** (IOException e) {

System.***out***.println("An error occurred while printing the car information to file.");

e.printStackTrace();

}

}

// Method to print inventory to file

**public** **static** **void** printInventoryToFile() {

**try** {

FileWriter writer = **new** FileWriter("inventory.txt");

**for** (AutoInventory car : *inventory*) {

writer.write(car.toString());

}

writer.close();

System.***out***.println("Inventory has been printed to file.");

} **catch** (IOException e) {

System.***out***.println("An error occurred while printing the inventory to file.");

e.printStackTrace();

}

}

}

Output:

Graphical user interface, text, application, email

Description automatically generated Text

Description automatically generated 