**Portfolio Project**

Jacob W. Palmer

CSU Global

CSC320: Programming 1

Herbert Pensado

June 9, 2024

This has been my most challenging course yet and has showed a lot of places where I can improve. The challenges I have faced come in the form of personal and technical challenges, ranging from managing time, learning a new language, to finding the best way to learn a new language. I did really like that the class was structured in a way that allowed me to reference my Basic Programming Material. I’m not proficient with Python, but the ability to look back at those programs and connect the dots between how both languages work was extremely helpful in this course.

Not directly related to learning Java, but a hurdle, nonetheless, was working full time, being an Air Force reservist, and being a full-time student. It’s not bad doing all three while I’m able to be home, but when I’m travelling it’s harder because I must work off a laptop. It also really helped that I could get ahead of my coursework and set myself up for success.

Reflecting on learning Java, it wasn’t too bad, but I could have used more practice work. I utilized Codacedemy and got through their basic course quickly, but it didn’t give me enough experience to make the class easy. I think back to how I used to take home a sheet of math problems, that would go over a fundamental concept. I need more of that and the option to create smaller programs as practice would help. It is nice to know that the language is similar to Python and the parallels between the two made it easier to pick up.

Another way of making the study of Java, and programming in general, is the ability to pick up the patterns with the methods and functions that are predefined. I spent a lot of time trying to find different methods that would do what I need. I could have saved a lot of time if I had flashcards to study or a pattern with the methods available, that could also translate to other languages. I now know how I can do better, by memorizing more of these methods as it’ll be crucial to making larger and more complex programs.

As I asked peers, who have programming experience, whether it be Java, Python, Typescript or something else, I found that each always had a hardened opinion on the language I was learning. I haven’t taken on that kind of opinion yet; however, I do see the benefit in learning Java. I think speaking to or pointing out the connection between Java and Python could be helpful. An example would be how initializing a list in Python is easy, however in Java you have to setup the parameters yourself. Java gives you more control over how it’s set up, but it’s all stuff that Python does for the user automatically. I think making those connections helps students understand how each language learned can be broken down into smaller pieces or languages, that can lead them to the root of all languages and how they’re used.

Even though I don’t have a great understanding of Java, I have the skills to make some very basic programs now. To make this better for me, I need more repetition with methods and principles that make a good program. I’m happy about the structure of the course and the choice of the language as it gives me more insight into how these programming languages are learned. On a personal level, I’m also happy that we’re learning Java because it’s heavily used for the program I’m working on and allows me to understand more of the code base and make me more valuable to my team.

A tool that helped me here, at the very end of the course and this assignment, is the Live Share module for VS Code. I was able to interface with peers, get feedback and work through problems in real time with others. It was my most time efficient way of learning. I hope that this is common amongst other IDEs as I think this would be great for classmates to help each other and for the online tutors as well. Thank you for taking the time to lead this course.

**SOURCE CODE**

// Changed from having four methods, to have a single method controlling the vehicle information.

// Create a method for handling how the inventory will be displayed.

// This method will also encompass the header for the inventory, making it more visually appealing.

// Utilize a while loop to keep the program running until the user decides to exit the program

// Add an exit menu button and logic

import java.util.Scanner;

import java.util.ArrayList;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

// import VehicleTwo;

class Main {

public static void displayInventory (ArrayList<VehicleTwo> vehicleInventory) {

try {

String HeaderString = String.join( " ", "MAKE", "MODEL", "COLOR", "YEAR", "MILEAGE");

System.out.println(HeaderString);

for (int i = 0; i < vehicleInventory.size(); i++) {

System.out.println(vehicleInventory.get(i));

}

}

catch (Exception e) {

System.out.println("An error has occurred" + e.getMessage());

}

}

public static void main (String[] args) {

//Scanners for user inputs

Scanner UserInput = new Scanner(System.in);

//Vehicle list instantiation

ArrayList<VehicleTwo> vehicleInventory = new ArrayList<VehicleTwo>();

while (true) {

System.out.println("\nBAD NEWS CAR SALES\n");

System.out.println("Select from the following options:");

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

System.out.println("1. ADD VEHICLE");

System.out.println("2. UPDATE VEHICLE");

System.out.println("3. REMOVE VEHICLE");

System.out.println("4. DISPLAY INVENTORY");

System.out.println("5. EXPORT INVENTORY");

System.out.println("6. EXIT");

//Method for applying user inputs

int menuNumber = UserInput.nextInt();

//Checking for appropriate user menu choice

if (menuNumber <= 0 || menuNumber >= 7) {

System.out.printf("Not a menu option. Please choose a number between 1-6.");

}

//logic for adding a vehicle

else

if (menuNumber == 1) {

try {

//Tried requesting all vehicle information at once, but didn't output the way I wanted it to

//Changed to reflect requesting each piece of information separately and then triggering the scanner

System.out.printf("\nEnter make: ");

String make = UserInput.next();

System.out.printf("Enter model: ");

String model = UserInput.next();

System.out.printf("Enter color: ");

String color = UserInput.next();

System.out.printf("Enter year: ");

int year = UserInput.nextInt();

System.out.printf("Enter mileage: ");

int mileage = UserInput.nextInt();

VehicleTwo newVehicle = new VehicleTwo(make, model, color, year, mileage);

vehicleInventory.add(newVehicle);

System.out.println("\nVehicle has been added:");

System.out.println(make + " " + model + " " + color + " " + year + " " + mileage + " ");

}

catch (Exception e) {

System.out.println("An error has occurred" + e.getMessage());

}

}

//logic for updating a vehicle in the list

else if (menuNumber == 2) {

if (vehicleInventory.size() <= 0) {

System.out.printf("\n Inventory is empty \n");

}

else

System.out.printf("Enter the index number of the vehicle to be updated: ");

int vehicleIndex = UserInput.nextInt();

if (vehicleIndex <= 0 || vehicleIndex > vehicleInventory.size()) {

System.out.printf("\n Invalid Index Number \n");

}

else

try {

System.out.printf("\nEnter make: ");

String make = UserInput.next();

System.out.printf("Enter model: ");

String model = UserInput.next();

System.out.printf("Enter color: ");

String color = UserInput.next();

System.out.printf("Enter year: ");

int year = UserInput.nextInt();

System.out.printf("Enter mileage: ");

int mileage = UserInput.nextInt();

//Setting VehicleTwo object to variable newVehicle

VehicleTwo newVehicle = new VehicleTwo(make, model, color, year, mileage);

//Changing information at vehicleIndex with newVehicle information

vehicleInventory.set(vehicleIndex, newVehicle);

System.out.printf("\nVehicle has been updated:\n");

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

displayInventory(vehicleInventory);

}

catch (Exception e) {

System.out.println("An error has occurred" + e.getMessage());

}

}

//logic for removing a vehicle from the list

else if (menuNumber == 3) {

if (vehicleInventory.size() <= 0) {

System.out.printf("\n Inventory is empty \n");

}

else

System.out.printf("Enter the index number of the vehicle to be removed: ");

int IndexToRemove = UserInput.nextInt();

if (IndexToRemove < 0 || IndexToRemove > vehicleInventory.size()) {

System.out.printf("\n Invalid Index Number \n");

}

else

try {

vehicleInventory.remove(IndexToRemove);

System.out.printf("\nVehicle has been removed. \n");

}

catch (Exception e) {

System.out.println("An error has occurred" + e.getMessage());

}

}

//logic for displaying the list (vehicle inventory)

else if (menuNumber == 4) {

if (vehicleInventory.size() <= 0) {

System.out.printf("\nInventory is empty");

}

else

displayInventory(vehicleInventory);

}

//logic for converting the list to a .txt file

else if (menuNumber == 5) {

if (vehicleInventory.size() <= 0) {

System.out.printf("\nInventory is empty");

}

else

System.out.printf("Do you want to print the information to a file? (Y/N) ");

String YesNo = UserInput.next();

System.out.println(YesNo);

if (YesNo.contains("Y")) {

System.out.println("YES");

String filePath = "C:\\Users\\Jake\\OneDrive\\Desktop\\Folder\\School\\CSU Global\\CSC320.txt";

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

for (VehicleTwo vehicleTwo : vehicleInventory) {

writer.write(vehicleTwo.toString());

writer.newLine();

}

System.out.printf("\nVehicle inventory has been printed! \n");

}

catch (IOException e) {

System.out.println("\nAn error has occurred" + e.getMessage());

}

}

else

System.out.printf("\nExiting the menu. Goodbye!");

}

//logic for exiting the program, otherwise it will loop back to the main menu

else if (menuNumber == 6) {

try {

System.out.printf("\nExiting the system. Goodbye!");

UserInput.close();

break;

}

catch (Exception e) {

System.out.println("\nAn error has occurred" + e.getMessage());

}

}

}

}

}

public class VehicleTwo {

//private variables for vehicle description

private String make;

private String model;

private String color;

private int year;

private int mileage;

//vehicle object

public VehicleTwo(String make, String model, String color, int year, int mileage) {

try {

this.make = make;

this.model = model;

this.color = color;

this.year = year;

this.mileage = mileage;

}

catch (Exception e){

System.out.printf("Something went wrong!" + e.getMessage());

}

}

@Override

public String toString() {

try {

return make + " " + model + " " + color + " " + year + " " + mileage + " ";

}

catch (Exception e) {

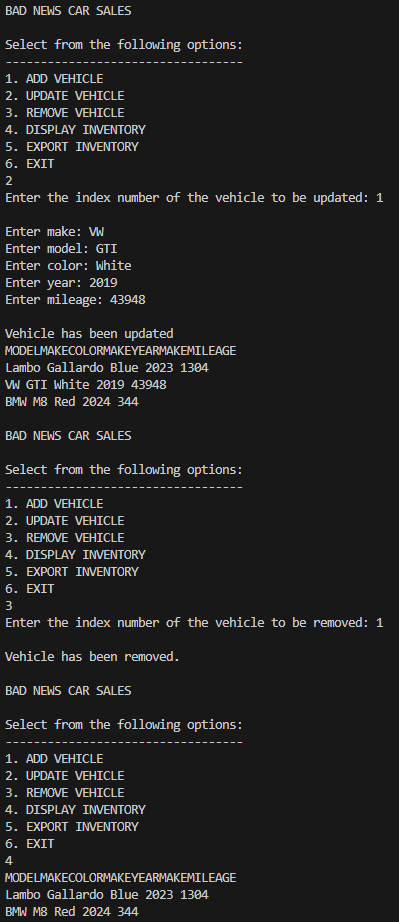
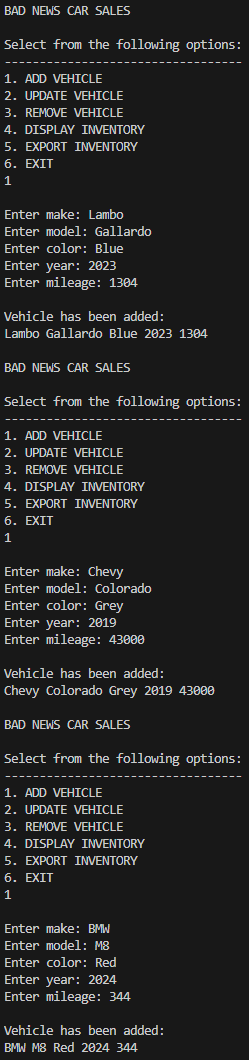
System.out.println("An error has occurred" + e.getMessage());

return "";

}

}

}

A screenshot of a computer program

Description automatically generated

