**Portfolio Project Option 2**

William Keller

Colorado State University Global

CSC372: Programming 2

Gottipamula, Santosh

9-1-25

**Portfolio Project Option 2**

**Milestones**

I’m not sure what this portion of the assignment is asking me to do. My Milestones have been submitted independently and can alternatively be found in the Github Links.

**Lessons Learned**

This was my second java course. As such I already understood the basic concepts of what an object was and what a class was. There were also some very surface level introductions to error handling in the first class, and I was happy to see an expanded education on that subject.

Our first module covered inheritance. I believe this was already covered in our first java course. Even if it wasn’t I had done some self-learning exercises with codecademy.com which allowed me to learn this before. Also, I work with a program which uses the concept of inheritance to build out digital twins of manufacturing equipment. The template structure and parent child relationships are not new to me.

Polymorphism was a concept I was a little less familiar with. While I understood overloading, which I now understand as Compile-Time Polymorphism, I did not know about Run-Time Polymorphism. At least I didn’t know about it as a concept. The ability to pass different types of objects to a function and have the function handle the object differently is of course very powerful.

The second module was one of my favorites. I have worked through some modules in codecademy.com on HTML, but never knew how to tie a java backend to a front end. Learning about some basic UI functionality here was very exciting to me.

Similar to the second module, the third module used a different library to give us more UI capabilities. Putting the objects of a java program on the screen can be a bit difficult and figuring out how to organize them is annoying. Though I feel I learned quite a bit here, I very much prefer back-end programming. I had to use one of the old HTML tricks to draw out each “container” that I wanted to stack in the frame and figure out in what order to display them.

Module 4 on Abstract classes and Interfaces was completely new to me. I had never considered building an abstract class before, and the concept of an interface never crossed my mind. However in the application I work with I can see parallels between the Abstract classes that we call templates and the interfaces that we call Shapes. In this application you can never use the functionality of a template directly, you can only implement something from it and use that. Similarly you can never use a shape. You add properties and methods to the shape and then attach the shape to any number of things that then gain those properties and methods

Mode 5 covered recursion. I do not like recursion. I had always heard about it as difficult to understand and work through, and I think I took that as ‘well, if it’s difficult to do, but people still do it, it must have a great benefit’. But no. In truth recursion is worse on the system resources than simple loops are. To add to the stack forever until hitting your base case just sounds unreasonable to begin with. I remember back when the google browser was the best because it was the fastest. These days I find it difficult to run a videogame on my PC with 32 gigs of ram if I have a few google tabs open. I’m deciding I’m going to blame recursion until proven otherwise.

Module 6’s Sorting and searching algorithms were good to learn. These have been covered in other courses, but like anything else practice makes perfect and I think I understood them a bit better after this course. I also don’t think I ever had to actually write one myself before, so that was a good exercise.

Module 7 gave a good overview of the different data storage options available, though I know I’m going to have to work with them more extensively to understand which is the right one to use for which reason. I’ve always been interested in how hashing can make my applications more efficient, and honestly I don’t think I’m at that level of understanding yet. But this was a good intro into the concept.

We capped off with module 8, which gave us the confusingly named ‘stacks’, and the easily understood queues. While the concept of a stack is a good one to use for this first in, last out system, I definitely thought they were talking about ‘the stack’ for a while.

And most importantly we got into Exception Handling. The backbone of all good applications. I have worked with error handling, but I have never understood it as I do now. I really had no idea that a thrown error was really the application passing an error object up the chain to find something that would handle it. Just identifying that it is in fact an object with attributes goes a long way in my understanding of the use of them. I find that too often people try teaching in concepts and the concepts lose me entirely, but if they explain to me the minutia of how it really works I can grasp the concepts quite easily.

**Program**

Though the execution only shows valid inputs, I tested this with invalid mpg numbers, strings for mpg numbers, and selecting ‘n’ and bad inputs for all of the Y/N options. One of the few changes I would make for a real program would be to validate the inputs instead of throwing an error from the constructor, but I feel like that is what I was being asked to do here. It is handled appropriately, but is less good from a user interaction standpoint.

**Vehicle.java**

package ct8;

public class Vehicle {

private String make;

private String model;

private double milesPerGallon;

public Vehicle() {

this.make = "No make Entered";

this.model = "No model Entered";

this.milesPerGallon = -1;

}

public Vehicle(String make, String model, double milesPerGallon) {

if (milesPerGallon <= 0) {

throw new IllegalArgumentException("Miles per Gallon Cannot be 0 or less than 0");

}

this.make = make;

this.model = model;

this.milesPerGallon = milesPerGallon;

}

public String getMake() {

return make;

}

public void setMake(String make) {

this.make = make;

}

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}

public double getMilesPerGallon() {

return milesPerGallon;

}

public void setMilesPerGallon(double milesPerGallon) {

if (milesPerGallon <= 0) {

throw new IllegalArgumentException("Miles per Gallon Cannot be 0 or less than 0");

}

this.milesPerGallon = milesPerGallon;

}

*@Override*

public String toString() {

return "Make: " + make + "\n" + "Model: " + model + " \n" + "MPG: " + milesPerGallon;

}

}

A screen shot of a computer program

AI-generated content may be incorrect.

**Driver.java**

package ct8;

import java.util.LinkedList;

import java.util.Scanner;

import java.util.InputMismatchException;

import java.io.IOException;

import java.util.Comparator;

import java.io.BufferedWriter;

import java.nio.charset.StandardCharsets;

import java.nio.file.\*;

public class Driver {

private static LinkedList<Vehicle> *VList*= new LinkedList<>();

private static Boolean *vehToAdd* = true;

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

public static void main(String[] args) {

// **TODO** Auto-generated method stub

System.***out***.println("Welcome to your vehicle inventory system!");

while (*vehToAdd*) {

String make;

String model;

double mpg = -1;

char cont;

while (true) {

System.***out***.println("Please enter your vehicle's make.");

make = *input*.nextLine();

System.***out***.println("Please enter your vehicle's model.");

model = *input*.nextLine();

while(true) {

System.***out***.println("Please enter the approximate miles per gallon of fuel consumption for this vehicle.");

try {

mpg = *input*.nextDouble();

} catch(InputMismatchException e) {

System.***out***.println("Value must be a number, try again.");

}

*input*.nextLine();

break;

}

//validate and add the vehicle to the List

System.***out***.println("You would like to add a " + make + " " + model + " with " + mpg + " miles to the gallon. Is this correct? (Y/N)");

cont = *input*.nextLine().toUpperCase().charAt(0);

if (cont != 'Y' && cont != 'N') {

System.***out***.println("Please enter either Y for yes, or N for no to continue");

continue;

} else if (cont == 'Y') {

try{

*VList*.add(new Vehicle(make, model, mpg));

System.***out***.println("Vehicle successfully added to List.");

break;

} catch (IllegalArgumentException e) {

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

System.***out***.println("Your vehicle was not able to be added, please try again;");

continue;

}

} else {

System.***out***.println("Ok, let's try again.");

continue;

}

}

while (true) {

System.***out***.println("Would you like to add another vehicle? (Y/N)");

cont = *input*.nextLine().toUpperCase().charAt(0);

if (cont != 'Y' && cont != 'N') {

System.***out***.println("Please enter either Y for yes, or N for no to continue");

continue;

} else if (cont == 'Y') {

break;

} else {

*vehToAdd* = false;

break;

}

}

}// end adding vehicles

*VList*.sort(Comparator.*comparing*(Vehicle::getMilesPerGallon));

for (Vehicle v: *VList*) {

System.***out***.print(v.toString());

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

}

try {

*saveToFile*(*VList*,Paths.*get*(System.*getProperty*("user.dir"), "vehicles.txt"));

} catch(IOException e) {

System.***out***.println("An error has occured.");

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

}

}

public static void saveToFile(LinkedList<Vehicle> Vlist, Path file) throws IOException {

try (BufferedWriter writer = Files.*newBufferedWriter*(

file, StandardCharsets.***UTF\_8***,

*StandardOpenOption*.***CREATE***,

*StandardOpenOption*.***TRUNCATE\_EXISTING***)) {

for (Vehicle v : Vlist) {

writer.write(v.getMake() + "\n"

+ v.getModel() + "\n"

+ v.getMilesPerGallon()

+ "\n\n");

writer.newLine();

}

}

}

}

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Output file

A screenshot of a computer

AI-generated content may be incorrect.

Github

<https://github.com/Strungul/CSC372>