PARKING LOT SYSTEM PROJECT REPORT

By: Dylan Hua

TABLE OF CONTENTS

INTRODUCTION

OBJECTIVE

Implement a system that manages a parking lot using object-oriented principles.

SPECIFICATIONS

- Parking lot has multiple levels. Each level has multiple rows of spots
- Motorcycles, cars, and buses can park. It has motorcycle spot, compact spot, and large spot.
- A motorcycle can park in any spot.
- A car can park in either a single compact spot or a single large spot
- A bus can park in 5 large spots that are consecutive and within the same row. It cannot park in small spot.

ASSUMPTIONS

- If the user enters the number of large spots that fills up the entire level, he or she won't be asked for the number of motorcycle and compact spots, and there will only be large spots for that entire level.
- If the number of large and motorcycle spots fill up the entire level, the parking lot will not have any compact spots.
- After asking the user for the number of large and motorcycle spots, the remaining spots will be filled up with compact spots.
- Parking lot spots are formatted in this order: large, compact, and motorcycle.
- Vehicles will park in or be removed from the 1st available spot(s).

SUMMARY

For this project, I first analyzed the problem to understand the requirements and make assumptions. Next, I designed how the parking lot will be managed based on the requirements and my assumptions. Then, I started programming the parking lot's behavior based on the design. Finally, I did a demonstration of my program in front of the teaching assistant (TA). As a result, I earned full credit for this project.

TECHNOLOGIES USED

- **Programming Language:** Java
- **UML Tool:** StarUML
- **Integrated Development Environment:** Eclipse

PROCESS

ANALYSIS

I read the project specifications several times to ensure that I understand what my project is supposed to do. However, the instructions only said what vehicles there will be, what spots those vehicles can park in, and what the parking lot has. For managing the parking lot, I made assumptions, such as vehicles parking in first available spot(s).

DESIGN

After analyzing the problem and making assumptions, I identified the objects and their attributes, what classes to include in the design, and what responsibilities those classes have. Then, I drew a UML class diagram using StarUML to model the relationships between classes, such as inheritance or aggregation. This gives me a visualization of the parking lot system's structure.

PROGRAMMING

Based on the design, I programmed the parking lot's behavior in Java using object-oriented programming. In fact, I encapsulated (grouped) related private variables and public methods in superclasses and subclasses. If a specific vehicle (subclass) is a type of general vehicle (abstract superclass), there is an inheritance relationship in which the vehicle subclass will extend the vehicle superclass. If a spot (abstract superclass) contains a vehicle (abstract superclass), there is an aggregation relationship in which the vehicle is a private variable in the spot class.

TESTING

When I finished coding a class of an object or the part for getting a specific input, I test that code piece to ensure that piece is behaving as I want it to. To resolve errors, I used Eclipse's debugging feature to toggle breakpoints and run through each of those breakpoints to see how my program behaves. Then, I modify my code based on any errors that I see in Eclipse's debugging window. I did this testing process multiple times until my program behaves the way I wanted it to.

DEMONSTRATION

Before receiving any credit for this project, I must show a sample run of my program and explain whichever part of the code that the TA asks.

RESULT

After doing a sample run of my program and explaining my code that gets the number of levels from the user, the TA gave me 105 points. He gave me 5 extra points because my program can remove vehicles (a bonus feature of my program) from the parking lot.

PARKING LOT SYSTEM

DESCRIPTION

This system manages a parking lot using the 4 object-oriented principles, which are inheritance, polymorphism, abstraction, and encapsulation. In this program, cars, motorcycles, and buses can park in or be removed from compact, small, and large spots. Users can specify what the parking lot will have.

DESIGN

Click here to see the UML class diagram of the parking lot system.

FEATURES

- Users can choose the number of levels, number of spots per level and per row, number of large and motorcycle spots per level, whether to park or remove vehicle, which vehicle to park or remove, and spot type for the chosen vehicle to park in or be removed from.
- Parking lot has multiple levels. Each level has multiple rows of spots. A spot can be small, compact, or large.
- A motorcycle can park in any spot.
- A car can park in a compact or a large spot.
- A bus can only park in 5 consecutive, large spots within the same row.
- Vehicles will park in or be removed from the 1st available spot(s).
- Parking lot spots are formatted in this order: large, compact, and motorcycle.

HOW TO RUN

See README.md in my GitHub repository.

CONCLUSION

CHALLENGES

For me, I spent a considerable amount of time in identifying the objects, classes to include, relationships between classes, and responsibilities that should go for each class. I also put in a lot of effort in fixing logic errors in my program, such as buses not parking in the first 5 available, large spots.

WHAT I LEARNED

Thanks to this project, I have a better understanding of object-oriented design and programming. More specifically, I learned how to identify the objects, classes to include, relationships between classes, and responsibilities that should go for each class. I also learned how to draw UML class diagrams with StarUML to model the relationships between these classes. In addition, I gained experience in using regex expressions for validating user input. Finally, I now know how to practice object-oriented programming in coding and document Java classes and methods properly.