ParkADuke

Introduction

JMU Parking Services has contracted with the Computer Science department to develop a parking management system for the new Mason street parking garage. They have provided the problem specification below.

Specification

When a car enters the garage the automated license plate scanner will provide the plate number, the arrival time, and indicate whether the car is compact or full-sized. After the car is scanned a large LED display should assign the car to the nearest available space. The display should notify the customer if no spaces are available.

The building has four floors with exactly 100 spaces per floor. The spaces on the first floor are labeled "A 0" through "A 99", the spaces on the second floor are "B 0" - "B 99" etc. The first 10 spots on each floor are reserved for compact cars. All spaces are numbered according to their proximity to the entrance.

An exit scanner will scan the license plates of departing cars. Cars should be charged $5.00 for the first 30 minutes, and $1 for each additional hour.

Design

1. Have one member of your team read the specification out loud. Discuss the specification and ask your instructor if you have any questions about the requirements.

2. Circle all nouns and noun phrases in the specification.

3. Cross out duplicate appearances of any term you circled in step 2 (include synonyms).

4. The engineering department has already handled the hardware aspects of the system. The attached UML illustrates the Java classes they have provided for interfacing with their devices. (Private fields are not shown.) Cross out any nouns that are already represented by one of the provided classes.

5. Cross out any remaining terms that don't need to be represented in your design.

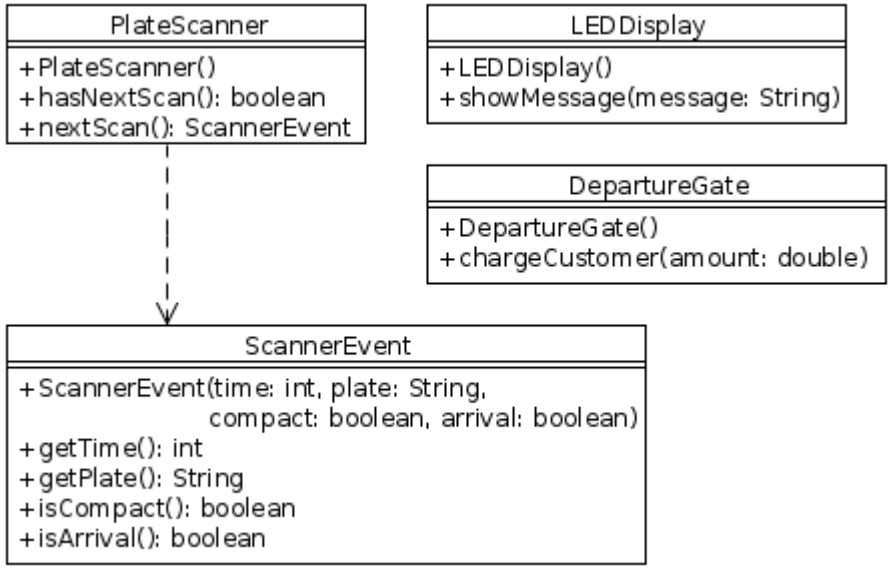
6. List each of the remaining terms below. Each of these terms need to be represented somehow in your final design. Some should be programmer-defined classes, while some may be represented by primitive variables or strings. Circle the terms that seem like good candidates for classes.

7. Develop UML diagrams for each of the classes you selected in the previous step. For each class consider:

- What is this class responsible for *knowing*? This should give you an idea of what the fields for the class should be.
- What is this class responsible for *doing*? This should give you an idea of what methods are required.

8. Once you have finished a draft of your design, show it to an instructor or TA.

9. Stub-out your classes on the back of your design document.


PlateScanner
+PlateScanner()
+hasNextScan(): boolean
+nextScan(): ScannerEvent
LEDDisplay
+LEDDisplay()
+showMessage(message: String)
DepartureGate
+DepartureGate()
+chargeCustomer(amount: double)
ScannerEvent
+ScannerEvent(time: int, plate: String,
compact: boolean, arrival: boolean)
+getTime(): int
+getPlate(): String
+isCompact(): boolean
+isArrival(): boolean