Analiza și Modelarea Sistemelor Software - Lab 3[1]

Traian Șerbănuță

2025

Agenda

- Class diagrams
- Package diagrams
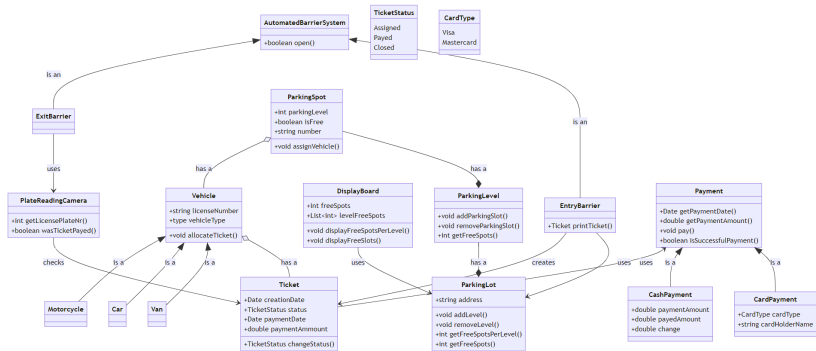- Component diagrams

Tools

- Lucidchart — component and package diagrams
- Mermaid.js — class diagrams

Parking Lot Exercise

Scenario: Airport parking lot

- ▶ Contains multiple levels – *3 levels*
- ▶ Sensors at each parking slot to detect if a slot is free or not
- ▶ One entrance and one exit
- ▶ Entrance has a display showing:
    - ▶ Total free spots
    - ▶ Free spots per level
- ▶ Exit has a camera that:
    - ▶ Reads license plates
    - ▶ Opens the barrier if the ticket was paid
- ▶ Multiple payment kiosks on each level
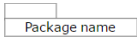- ▶ Payment can be made with **cash** or **credit card**

AutomatedBarrierSystem
+boolean open()

TicketStatus
Assigned
Payed
Closed

CardType
Visa
Mastercard

ExitBarrier

Is an

ParkingSpot
+int parkingLevel
+boolean IsFree
+string number
+void assignVehicle()

Is an

PlateReadingCamera
+int getLicensePlateNr()
+boolean wasTicketPayed()

has a

Vehicle
+string licenseNumber
+type vehicleType
+void allocateTicket()

has a

DisplayBoard
+int freeSpots
+List<int> levelFreeSpots
+void displayFreeSpotsPerLevel()
+void displayFreeSlots()

ParkingLevel
+void addParkingSlot()
+void removeParkingSlot()
+int getFreeSpots()

EntryBarrier
+Ticket printTicket()

Payment
+Date getPaymentDate()
+double getPaymentAmount()
+void pay()
+boolean isSuccessfulPayment()

checks   Is a   Is a   Is a

has a   uses   has a   creates   uses   uses   Is a   Is a

Motorcycle

Car

Van

Ticket
+Date creationDate
+TicketStatus status
+Date paymentDate
+double paymentAmount

+TicketStatus changeStatus()

ParkingLot
+string address

+void addLevel()
+void removeLevel()
+int getFreeSpotsPerLevel()
+int getFreeSpots()

CashPayment
+double paymentAmount
+double payedAmount
+double change

CardPayment
+CardType cardType
+string cardHolderName

Package Diagram

▶ A **package** is a collection of logically related UML elements.

  ▶ Simplify complex class diagrams by grouping classes into *packages*.

Package diagrams are commonly used to:

▶ Provide a visual organization of layered architecture
▶ Represent logical structure within UML classifiers (like software systems)

Symbol	Name	Description
Package name	Package	Groups common elements based on data, behavior, or user interaction
·······>	Dependency	Depicts the relationship between one element (package, named element, etc) and another

Dependencies in Package Diagrams

There are two main types of dependencies between packages:

1. **Import dependency** — allows access to all public elements of another package

2. **Access dependency** — limits access to specific elements only
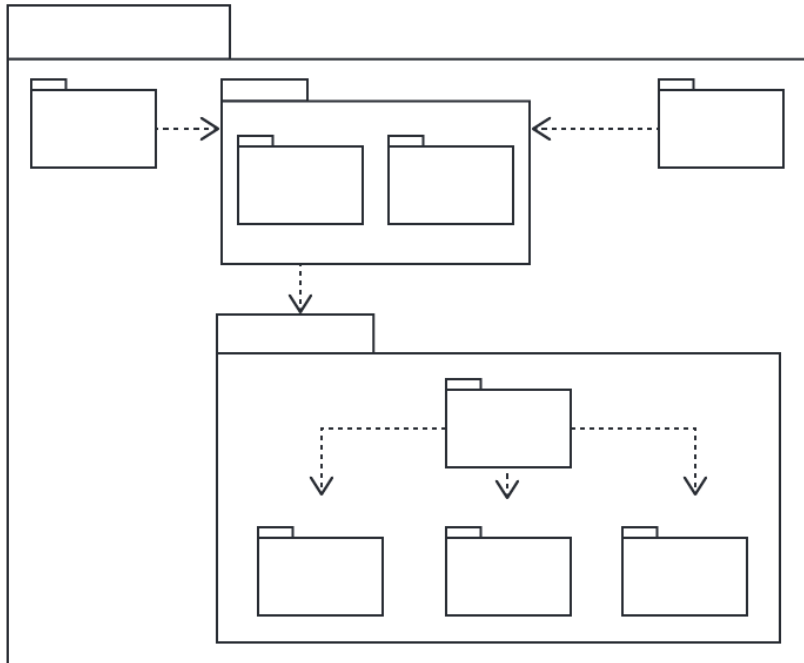
Create a **package diagram** based on the **class diagram for the parking lot**.

Component Diagrams

▶ Specialized class diagrams that focus on a system's **components**.

▶ Used to model the **static implementation view** of a system.

Symbol	Name	Description
Component	Component	Modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment
▷	Required interface	Represents the services needed/used by the component
─○	Provided interface	Represents the services delivered by the component

Exercise

Create a **component diagram** based on the **class diagram for the parking lot**.