**Practical 6**  
**Modeling UML Class Diagrams**  
*Structural and Behavioral aspects | Class diagram | Elements in class diagram | Class | Relationships | Draw Class Diagram*

**Aim of the Experiment:**  
To design a UML Class Diagram for the **Garage Management System** that models its structure and relationships between entities.

**Introduction**

Classes are the structural units in object oriented system design approach, so it is essential to know all the relationships that exist between the classes, in a system. All objects in a system are also interacting to each other by means of passing messages from one object to another. Sequence diagram shows these interactions with time ordering of the messages.

**Objectives**

**After completing this experiment you will be able to:**

* Graphically represent a class, and associations among different classes
* Identify the logical sequence of activities undergoing in a system, and represent them pictorially

**1. Structural Aspects:**

These define the **static parts** of the system — what components exist and how they are related.

**Key Points:**

* Focuses on **classes**, **objects**, **attributes**, and **relationships**.
* Represents the **framework or blueprint** of the system.
* Shows what the system contains — but not how it behaves over time.

**Examples:**

* **Class Diagrams**
* **Object Diagrams**
* **Component Diagrams**
* **Deployment Diagrams**

**In GMS (Garage Management System):**

A structural aspect would show the Customer, Vehicle, Service, Invoice classes and how they relate to each other.

**2. Behavioral Aspects:**

These define the **dynamic behavior** of the system — how it responds to internal and external stimuli.

**Key Points:**

* Focuses on **interactions**, **flows**, and **state changes**.
* Describes how the system behaves over time.
* Helps in understanding **use cases**, **process flow**, and **logic**.

**Examples:**

* **Use Case Diagrams**
* **Sequence Diagrams**
* **Activity Diagrams**
* **State Machine Diagrams**

**In GMS:**

A behavioral aspect would describe how a customer **books a service**, how the system **sends an invoice**, or how **mechanics update service status**.

**Class diagram**

It is a graphical representation for describing a system in context of its static construction.

**Elements in class diagram**

Class diagram contains the system classes with its data members, operations and relationships between classes.

**Class**

A set of objects containing similar data members and member functions is described by a class. In UML syntax, class is identified by solid outline rectangle with three compartments which contain

* **Class name** A class is uniquely identified in a system by its name. A textual string is taken as class name. It lies in the first compartment in class rectangle.
* **Attributes** Property shared by all instances of a class. It lies in the second compartment in class rectangle.
* **Operations** An execution of an action can be performed for any object of a class. It lies in the last compartment in class rectangle.

**Relationships:**

* **Association**: Shows relationship between two classes (e.g., Customer — books —> Service)
* **Aggregation**: Represents a whole-part relationship (e.g., Garage has Mechanics)
* **Composition**: Strong form of aggregation (e.g., Invoice contains Billing Details)

.

**Case Study: Garage Management System (GMS)**

The **Garage Management System (GMS)** is a structured software system designed to manage the core activities of a garage such as service bookings, customer records, vehicle tracking, billing, and inventory control. Using **UML Class Diagrams**, we model the structural elements and their relationships for efficient system design.

**Class Diagram:**

The GMS includes the following main classes:

* **Customer**: Holds customer details.
* **Vehicle**: Represents customer vehicles.
* **Service**: Contains service-related details.
* **Mechanic**: Manages assigned services.
* **Invoice**: Handles billing operations.
* **Admin**: Oversees system and inventory.
* **Inventory**: Tracks parts and tools.

Each class contains **attributes** (data members) and **methods** (functions). For example, the Customer class includes attributes like name, contact, and methods like bookService().

**Relationships:**

* **Association**: A Customer is associated with one or more Vehicles.
* **Aggregation**: An Admin manages multiple Inventory items.
* **Composition**: An Invoice is composed of ServiceDetails—if the invoice is deleted, so are the details.

**Elements in Class Diagram:**

* **Class Name**: At the top of the box.
* **Attributes**: In the middle section.
* **Methods**: In the bottom section. Each class is represented using the standard three-compartment format.

**Composition:**

Composition is used between Invoice and ServiceDetails, indicating a strong lifecycle dependency—without the invoice, the service details are irrelevant.

This class diagram offers a blueprint for the GMS, ensuring clarity in the design and relationships among various system components.

**Conclusion :**

In this practical, we created a UML Class Diagram for the Garage Management System to represent its structural design. The diagram defines key classes like Customer, Vehicle, Mechanic, and Service, along with their attributes and operations. We identified relationships such as association and composition between these classes. This helped in visualizing how different components of the system interact. The diagram improves system understanding, design clarity, and supports object-oriented development. It also helps in planning future implementation efficiently. Thus, the class diagram plays a vital role in structured software development.

