# **HAND OF GOD MADE IT SIMPLE✍️**

# OBJECT ORIENTED PROGRAMMING

Contents:

1. Brief explanation of OOP, OOA and OOD

## 2. Identifying UML classes, attributes and methods from word problems and their relationships 3. Drawing UML class diagrams

1. Brief explanation of OOP, OOA and OOD

**Definition:**

Object-Oriented Programming (OOP) is a way of writing software where the focus is on creating objects. Objects are like real-world items or entities, such as a car, a book, or a person. Each object has specific characteristics (called attributes) and actions it can perform (called methods).

**Explanation:**

Imagine you're trying to describe your phone to someone. You could talk about its color, brand, and size. These are the attributes of the phone. You could also describe what it can do, like making calls, sending messages, or taking pictures. These are the methods. In OOP, we think of things (like the phone) as objects, and we write code to define their attributes and methods.

**Scenario:**

Suppose you're building a software application for a library. In OOP, you would create objects like Book, Member, and Librarian. Each of these objects would have attributes (like the title of the book, the name of the member) and methods (like borrowing a book, returning a book).

**Example:**

• **Book** (Object) o **Attributes:** title, author, ISBN, number of pages o **Methods:** borrow(), return()

**Note:** we always have to add the opening and closing bracket to the method names to show that they are methods. Methods can be referred to as their actions or what an object can do!

**Object-Oriented Analysis (OOA)**

**Definition:**

Object-Oriented Analysis (OOA) is the process of understanding and defining the problem that needs to be solved by identifying the objects involved and how they interact with each other.

**Explanation:**

Before you start building something, you need to understand what you're trying to create. OOA is like planning out how your software will work by identifying all the important pieces (objects) and figuring out how they should interact. This is done before writing any code.

**Scenario:**

Let's go back to the library example. During the OOA phase, you would sit down and think about what the library needs to do. You'd identify objects like Book, Member, and Librarian and determine what they need to do in the system. You'd ask questions like, "How do books get checked out?" and "What happens when a member returns a book late?"

**Example:**

In OOA for the library system:

* Identify objects: Book, Member, Librarian, LibraryCatalog
* Determine interactions: Member borrows Book, Librarian adds Book to LibraryCatalog.

**Object-Oriented Design (OOD)**

**Definition:**

Object-Oriented Design (OOD) is the process of taking the analysis (from OOA) and figuring out how to actually implement the system in code. This involves deciding on the details of how objects will be structured and how they will communicate with each other.

**Explanation:**

Once you've figured out what your system needs to do (OOA), you need to decide how to make it happen. OOD is like creating a blueprint for your software. You determine how the objects will be organized, what specific methods and attributes they will have, and how they will work together to achieve the goals you identified during OOA.

**Scenario:**

Continuing with the library example, in OOD, you would decide how to represent each object in code. You'd think about the details like what specific data needs to be stored in each object and how one object can interact with another. You might decide that Book objects will be stored in a LibraryCatalog object, and that the Member object will have a method to borrow() a Book.

**Example:**

* **Book** object:
  + Attributes: title, author, ISBN, status (borrowed or available) o Methods: borrow(), return()
* **LibraryCatalog** object:
  + Attributes: a list of all books o Methods: addBook(), findBook()
* **Member** object:
  + Attributes: name, memberID, borrowedBooks (a list of books) o Methods: borrowBook(), returnBook()

**Note:**

* **OOP** (Object-Oriented Programming): Writing code by creating objects with attributes and methods, like creating a Book object with a borrow() method.

* **OOA** (Object-Oriented Analysis): Figuring out what the problem is and identifying the objects and their interactions before writing any code, like deciding a Member needs to borrow a Book.

* **OOD** (Object-Oriented Design): Planning how to build the system in code by detailing how objects will work together, like deciding what data the Book object will store and how it will be borrowed.

## 2. Identifying UML classes, attributes and methods from word problems and their relationships

**What is UML (Unified Modeling Language)?**

**Definition:**

UML is a way of visually representing the design of a system. Think of it as a blueprint or a diagram that shows how different parts of a system (like a software application) are connected and how they work together.

**Explanation:**

Just like architects use blueprints to plan out buildings, software developers use UML to plan out software systems. UML helps in visualizing the structure and relationships between different parts of the software before actually building it.

This makes it easier to understand and communicate complex systems.

**Example:**

Imagine you’re designing a system for a simple online store. UML would help you draw diagrams that show all the important parts of the store (like customers, products, orders) and how they interact with each other.

**Understanding Classes in UML**

**Definition:**

A class in UML represents a blueprint or template for creating objects. An object is an instance of a class, much like how a cake (object) is made using a cake recipe (class). A class defines what an object will be made of (attributes) and what it can do (methods).

**Explanation:**

In real life, classes can be thought of as categories or types of things. For example, if we were talking about animals, “Dog” could be a class. This class would define what a dog is (its attributes) and what it can do (its methods).

**Example:**

* **Class:** Dog o **Attributes:** breed, age, color, size o **Methods:** bark(), eat(), sleep() **Identifying Classes from a Word Problem:**
* **Tip:** Look for nouns in the word problem. These often represent potential classes.
* **Scenario:**

Suppose you have a word problem about a library system. It mentions books, members, and librarians. These are all nouns, and each could represent a class in the system.

* + **Identified Classes:** Book, Member, Librarian

**Understanding Attributes in UML**

**Definition:**

Attributes are the characteristics or properties of a class. They describe what the class is like. For example, in a “Person” class, attributes could include the person’s name, age, and address.

**Explanation:**

Attributes are like the details that describe an object. If we take the Dog class again, attributes would be things like the dog’s breed or color. These are pieces of information that help describe the object.

**Example:**

* **Class:** Car o **Attributes:** make, model, year, color **Identifying Attributes from a Word Problem:**
* **Tip:** Look for descriptive words or phrases that provide more information about the nouns (potential classes) in the problem.
* **Scenario:**

In a word problem about an online store, it might describe a product as having a price, description, and stock quantity.

* + **Identified Attributes for Product Class:** price, description, stockQuantity

**Understanding Methods in UML**

**Definition:**

Methods are actions or functions that a class can perform. They define what an object can do. For example, a “Dog” class might have a method called bark(), which is something a dog can do.

**Explanation:**

If attributes are the “what” of a class, methods are the “how.” Methods are the behaviors or actions that the object can perform. In the Car class example, methods would be things like starting the car or honking the horn.

**Example:**

* **Class:** ATM o **Methods:** depositMoney(), withdrawMoney(), checkBalance() **Identifying Methods from a Word Problem:**
* **Tip:** Look for verbs or actions that describe what the classes (nouns) can do.
* **Scenario:**

In a word problem about a banking system, it might mention that customers can deposit money, withdraw money, and check their balance.

* + **Identified Methods for Customer Class:** depositMoney(), withdrawMoney(), checkBalance()

**Understanding Relationships in UML**

**Definition:**

Relationships in UML show how different classes are connected to each other. There are several types of relationships, but the most common are:

* **Association:** A general connection between two classes (e.g., a Teacher teaches Students).
* **Inheritance:** One class inherits the attributes and methods of another (e.g., a Dog is an animal, so Dog would inherit from the Animal class).
* **Aggregation:** A whole-part relationship where the part can exist independently of the whole (e.g., a Library has Books, but a Book can exist without the library).
* **Composition:** A stronger whole-part relationship where the part cannot exist without the whole (e.g., a House has Rooms, and rooms cannot exist without the house).

**Explanation:**

Relationships are the lines connecting different classes in a UML diagram. They show how classes interact with each other. For instance, in a school system, a Teacher might have a relationship with a Classroom, showing that the teacher teaches in that classroom.

**Example:**

* **Association:** A Doctor works at a Hospital.
* **Inheritance:** A Cat is a type of Animal (so Cat inherits from Animal).
* **Aggregation:** A Car has Wheels, but wheels can exist on their own.
* **Composition:** A Person has a Heart, and the heart cannot function without the person.

**Identifying Relationships from a Word Problem:**

* **Tip:** Look for connections between the nouns (potential classes) in the problem. These connections often indicate relationships.
* **Scenario:**

In a word problem about a school, it might mention that students enroll in courses, teachers teach courses, and students take exams.

* + **Identified Relationships:**
    - **Association:** Student enrolls in Course
    - **Association:** Teacher teaches Course
    - **Association:** Student takes Exam

**Step-by-Step Process to Identify UML Classes, Attributes, Methods, and Relationships from a Word Problem**

1. **Read the Word Problem Carefully:** o Look for nouns, verbs, and descriptive phrases. o Nouns will often become classes. o Descriptive phrases will point to attributes.

* + Verbs or actions will point to methods.

2. **Identify the Classes:**

* + List down all the important nouns. o Think about whether each noun represents something that could be an object in the system.
  + Example from a library system: Book, Member, Librarian.

3. **Identify the Attributes:**

* + Look for descriptive phrases that give more information about the nouns.
  + Consider what characteristics each class would need to have.
  + Example: Book class might have attributes like title, author, ISBN.

4. **Identify the Methods:**

* + Look for verbs or actions associated with the classes. o Consider what each class needs to do.
  + Example: Member class might have methods like borrowBook(), returnBook().

5. **Identify the Relationships:**

* + Look for connections between the classes. o Determine how the classes interact with each other.
  + Example: Librarian adds Book to the LibraryCatalog.

**Example: Applying This to a Word Problem**

**Word Problem:**

"A school system needs to track students, courses, and teachers. Students can enroll in courses, and teachers can teach multiple courses. The school also keeps track of grades for each student in each course." **Step 1: Identify Classes**

* Student
* Course
* Teacher
* Grade

**Step 2: Identify Attributes**

* Student class: name, studentID
* Course class: courseName, courseCode
* Teacher class: name, employeeID
* Grade class: gradeValue **Step 3: Identify Methods**
* Student class: enrollInCourse(), viewGrades()
* Teacher class: assignGrade(), teachCourse() **Step 4: Identify Relationships**
* **Association:** Student enrolls in Course
* **Association:** Teacher teaches Course
* **Association:** Student receives Grade in Course

**Example 1: E-commerce System**

**Word Problem:** "A company wants to develop an e-commerce system where customers can browse products, add items to their shopping cart, place orders, and make payments. The system should also allow administrators to manage product inventory and track orders." **Step 1: Identify Classes**

* Customer
* Product
* ShoppingCart
* Order
* Payment
* Administrator

**Step 2: Identify Attributes**

* Customer class: name, customerID, email
* Product class: productName, price, stockQuantity
* ShoppingCart class: cartID, items (list of Products)
* Order class: orderID, orderDate, totalAmount
* Payment class: paymentID, paymentMethod, paymentDate
* Administrator class: adminID, username **Step 3: Identify Methods**
* Customer class: browseProducts(), addToCart(), placeOrder()
* ShoppingCart class: addItem(), removeItem(), viewCart()
* Order class: calculateTotal(), confirmOrder()
* Payment class: processPayment(), refundPayment()
* Administrator class: manageInventory(), trackOrder() **Step 4: Identify Relationships**
* **Association:** Customer has a ShoppingCart (A customer owns a shopping cart).

**Association:** ShoppingCart contains Product (A shopping cart has products).

* **Association:** Customer places an Order (A customer places an order).
* **Composition:** Order includes Payment (A payment cannot exist without the order).
* **Aggregation:** Administrator manages Product inventory (Products exist independently, but the administrator manages them).

**Example 2: University Management System**

**Word Problem:** "A university management system needs to track students, courses, professors, and departments. Students can register for courses, and professors can teach multiple courses. Each department offers several courses, and professors belong to a department." **Step 1: Identify Classes**

* Student
* Course
* Professor
* Department
* Enrollment

**Step 2: Identify Attributes**

* Student class: name, studentID, major
* Course class: courseName, courseCode, credits
* Professor class: name, professorID, specialty
* Department class: departmentName, departmentCode
* Enrollment class: enrollmentID, grade **Step 3: Identify Methods**
* Student class: registerForCourse(), viewSchedule(), viewGrades()
* Course class: assignProfessor(), addStudent(), removeStudent()

Professor class: teachCourse(), gradeStudent()

* Department class: offerCourse(), assignProfessor()
* Enrollment class: enrollStudent(), assignGrade() **Step 4: Identify Relationships**
* **Association:** Student enrolls in Course through Enrollment (A student can enroll in multiple courses, and each course can have many students).
* **Association:** Professor teaches Course (A professor can teach multiple courses).
* **Aggregation:** Department offers Course (Courses can exist independently, but they are offered by a department).
* **Composition:** Professor belongs to a Department (A professor is tightly coupled with a department).

**Example 3: Hospital Management System**

**Word Problem:** "A hospital management system needs to handle patients, doctors, appointments, treatments, and medical records. Patients book appointments with doctors, doctors can have multiple appointments, and each appointment results in a treatment. Medical records track all treatments for each patient."

**Step 1: Identify Classes**

* Patient
* Doctor
* Appointment
* Treatment
* MedicalRecord

**Step 2: Identify Attributes**

* Patient class: name, patientID, dateOfBirth, medicalHistory
* Doctor class: name, doctorID, specialty

Appointment class: appointmentID, date, time

* Treatment class: treatmentID, description, treatmentDate
* MedicalRecord class: recordID, patientID, treatmentHistory (list of Treatments)

**Step 3: Identify Methods**

* Patient class: bookAppointment(), viewMedicalRecord()
* Doctor class: scheduleAppointment(), diagnosePatient()
* Appointment class: setAppointmentDetails(), confirmAppointment()
* Treatment class: administerTreatment(), updateTreatmentDetails()
* MedicalRecord class: addTreatment(), viewRecord() **Step 4: Identify Relationships**
* **Association:** Patient books Appointment with Doctor (A patient can have many appointments, and a doctor can have many appointments).
* **Composition:** Appointment results in Treatment (A treatment is directly tied to an appointment; it cannot exist without it).
* **Aggregation:** MedicalRecord contains Treatment (Medical records aggregate treatments, which can exist independently).
* **Composition:** MedicalRecord belongs to Patient (Medical records are specific to a patient and cannot exist without the patient).

**Summary of Relationships in the Examples:**

* **Association** is used to show a general connection between classes, such as a Customer having a ShoppingCart or a Student enrolling in a Course.
* **Composition** is a stronger relationship where one class cannot exist without the other, such as an Order including a Payment or an Appointment resulting in a Treatment.
* **Aggregation** is a whole-part relationship where the part can exist independently of the whole, such as a Department offering Courses or a MedicalRecord containing Treatments.

**Inheritance** could also be used if there were classes that shared common attributes and methods, but none of these examples specifically required it.

By breaking down these word problems, identifying the relevant classes, attributes, methods, and relationships, and then visually representing them using UML, anyone can gain a clearer understanding of how complex systems are structured and how the different components interact with each other.

## 3. Drawing UML class diagrams

**Drawing UML Class Diagrams**

Drawing UML class diagrams is a key skill in object-oriented design and analysis. These diagrams help visualize the structure of a system by showing its classes, their attributes, methods, and the relationships between them. Below is a stepby-step guide on how to draw UML class diagrams, explained in a way that even a layman can understand.

1. **Understanding the Basics of a UML Class Diagram**

**Definition:** A UML class diagram is a visual representation of the classes in a system. Each class is represented as a rectangle divided into three sections:

* + The **top section** contains the class name.
  + The **middle section** contains the attributes (characteristics) of the class.
  + The **bottom section** contains the methods (actions) the class can perform.

**Structure:**

* + **Class Name:** The name of the class (e.g., Book, Customer).
  + **Attributes:** The properties or characteristics of the class (e.g., title, author for a Book class).
  + **Methods:** The functions or actions that the class can perform (e.g., borrow(), return() for a Book class).

A diagram of a book

AI-generated content may be incorrect.

1. **Step-by-Step Guide to Drawing a UML Class Diagram Step 1: Identify the Classes**
   * **Start with a Word Problem:** Read the problem description carefully to identify the main entities (nouns) that could be classes.
   * **List the Classes:** Write down all the potential classes. These are usually the key nouns in the description.

**Example:** For a library system, you might identify the following classes: Book, Member, Librarian.

**Step 2: Define the Attributes**

* + **Think About Characteristics:** For each class, consider what details you need to store about that entity.
  + **List the Attributes:** Write down the attributes under each class. Attributes are usually adjectives or descriptive phrases.

**Example:** For the Book class, attributes might include:

* + title: String
  + author: String
  + ISBN: String
  + yearPublished: int

**Step 3: Define the Methods**

* + **Identify Actions:** Think about what actions or behaviors each class should be able to perform.
  + **List the Methods:** Write down the methods below the attributes. Methods are usually verbs or action phrases.

**Example:** For the Book class, methods might include:

* + borrow(): void
  + return(): void
  + reserve(): void

**Step 4: Determine Relationships Between Classes**

* + **Analyze Connections:** Look at how the classes are related to each other. This involves understanding how they interact or depend on each other.
  + **Types of Relationships:** Common relationships include:
    - **Association:** A basic connection between two classes (e.g., a Member borrows a Book).
    - **Inheritance:** One class is a subtype of another (e.g., Manager inherits from Employee).
    - **Aggregation:** One class is a part of another but can exist independently (e.g., Library has Books).
    - **Composition:** A stronger form of aggregation where the part cannot exist without the whole (e.g., House has Rooms).

**Example:**

* + Member borrows Book (Association)
  + Librarian manages Book (Association)
  + Manager inherits from Employee (Inheritance)

**Step 5: Draw the Diagram**

* + **Start with Classes:** Draw a rectangle for each class, dividing it into three sections (name, attributes, methods).
  + **Add Attributes and Methods:** Fill in the attributes and methods for each class.
  + **Draw Relationships:** Use lines to connect the classes, indicating relationships. Label the lines with the type of relationship if needed.

A black and white image of a white background

AI-generated content may be incorrect.

**4. Common Pitfalls to Avoid**

1. **Overcomplicating the Diagram:**
   * **Mistake:** Adding too much detail in a single diagram can make it hard to understand.
   * **Solution:** Focus on the key elements and use multiple diagrams if necessary.
2. **Mislabeling Relationships:**
   * **Mistake:** Confusing associations, inheritance, aggregation, and composition.
   * **Solution:** Make sure you understand each type of relationship and use the correct symbols.
3. **Inconsistent Naming:**
   * **Mistake:** Using inconsistent or unclear names for classes, attributes, or methods.
   * **Solution:** Use clear, consistent, and descriptive names.

**5. Practice Example: Drawing a Simple UML Class Diagram**

**Word Problem:** "A school system needs to track teachers, students, and classes. Teachers teach multiple classes, and students enroll in multiple classes."

**Step-by-Step Solution:**

1. **Identify Classes:** Teacher, Student, Class
2. **Define Attributes:**
   * Teacher: name, employeeID o Student: name, studentID o Class: className, classCode
3. **Define Methods:**
   * Teacher: teachClass() o Student: enrollInClass() o Class: scheduleClass()
4. **Determine Relationships:**
   * Teacher teaches Class (1..\* association) o Student enrolls in Class (1..\* association)

**Understanding Associations in UML**

In UML, an **association** represents a relationship between two classes that shows how they interact with each other. Associations are used when one class needs to be connected to another class to reflect some interaction or dependency between them. Let's break down why the relationships between Teacher and Class, and between Student and Class are considered associations, and specifically why they are 1..\* associations.

**1. *Teacher teaches Class (1.. Association)*\* Why it's an Association:**

* **Interaction Between Entities:** In a school or university system, the concept of a teacher is inherently tied to the concept of a class. A teacher's role involves teaching one or more classes. This interaction is why an association is used.
* **Direct Relationship:** The relationship is direct because the system needs to understand which teacher is responsible for teaching which class. Without this association, it would be impossible to track or manage the relationship between teachers and the classes they teach. *Why it's a 1.. Association:*\*
* **1:** The "1" in the association means that each Class is taught by exactly one Teacher. This reflects the reality in most educational settings where a class typically has one teacher responsible for it.
* \*\* \* :\*\* The " \* " means that each Teacher can teach multiple Classes. This is common in educational settings, where a single teacher often handles several classes.

**Summary:**

* A Teacher must be associated with at least one Class (hence the "1"), but a Teacher can be associated with many Classes (hence the "\*").
* The association shows the relationship between these two entities, making it clear that they are connected in the system. **2. *Student enrolls in Class (1.. Association)*\* Why it's an Association:**
* **Interaction Between Entities:** Students in an educational system are directly connected to the classes they take. The process of a student enrolling in a class is a key interaction that must be tracked, hence the need for an association.
* **Direct Relationship:** This relationship is essential for the system to understand which students are taking which classes. Without this association, the system wouldn't be able to manage enrollments or track which students are in which classes. *Why it's a 1.. Association:*\*
* **1:** The "1" in the association means that each Class can have multiple students enrolled in it. This reflects the common educational structure where a class typically has many students.
* \*\* \* :\*\* The " \* " means that each Student can enroll in multiple Classes. Students usually take several different classes each term or semester, so this multiplicity is necessary to represent that reality.

**Summary:**

* A Student must be associated with at least one Class (hence the "1"), but a Student can be associated with many Classes (hence the "\*").
* This association clearly shows that there is a relationship between students and classes, which is fundamental to any educational system.

**Visual Representation:**

Here’s how these associations are typically represented:

* **Teacher teaches Class:** This shows that a Teacher is responsible for teaching one or more Classes.

**NO NEED TO ADD THE TYPES LIKE THE STRING AND ALL THAT, NO**

**NEED TO ALSO ADD THE PLUS(+) AND MINUS(-**)

A black and white image of a line

AI-generated content may be incorrect.

* **Student enrolls in Class:** This indicates that a Student is enrolled in one or more Classes

A black and white image of a line

AI-generated content may be incorrect.