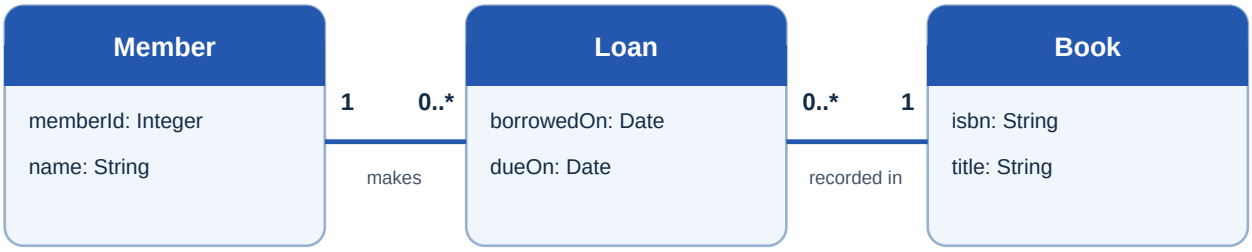
UML Multiplicities: A Worked Class Diagram

Free sample for university software engineering students | A small library-loan model with explanations, common mistakes, and a self-check exercise.

01 The modelling brief

A library lets members borrow books. Each borrowing event has a date and a due date. A member can have multiple loans, and a book can be borrowed across multiple loans over time. We want a model that can record each individual transaction.

02 Example class diagram



Conceptual example: a library borrowing system

03 Read each relationship from both directions

Relationship	Interpretation
Member (1) - Loan (0..*)	Each Loan belongs to exactly one Member. One Member may have no loans or many loans.
Book (1) - Loan (0..*)	Each Loan records exactly one Book. One Book may have no recorded loans or many historic loans.

Important design assumption: This example models historical borrowing records, not the rule that a physical book can be on loan to only one member at a time. That constraint must be handled separately using a business rule or additional modelling detail.

Check Your Understanding

Practice interpreting associations. This sample is for learning, not a submission-ready university assignment.

04 How to check the UML model

• Identify the classes from the nouns in the requirements, but do not turn every noun into a class.

• Model a Loan as a separate class if borrowing dates, returns and transactions must be tracked.

• State multiplicities on both ends of every association. Read them against a single instance on the opposite end.

• Check unusual cases: a newly registered member has no loans; a book may never have been borrowed; a returned book may appear in many historical loans.

• Validate the diagram against stated business rules and describe rules that the class diagram alone cannot enforce.

05 Common mistakes and fixes

Mistake	Better approach
Connecting Member directly to Book	Use the Loan class when attributes belong to each borrowing event.
Using * everywhere	State whether zero is allowed and where exactly one is required.
Misreading 1 and 0..*	A number at one end indicates how many of that class can be linked to one object at the other end.
Ignoring simultaneous-loan rules	Document and validate the rule in the design; a historical 0..* relationship does not guarantee active-loan uniqueness.

06 Self-check question

Question: A new member registers but has not borrowed a book. Does the diagram permit this? What multiplicity allows it?

Answer: Yes. The 0..* multiplicity at the Loan end means a Member can be linked to zero Loan instances.

Further learning and guided feedback

Study the full lesson: How to Read UML Class Diagram Multiplicities.
For feedback on your own software-design reasoning: Request UML tutoring and project guidance.

Reference: Object Management Group, UML specifications.
© Quality Assignment Help. Original instructional sample. University academic-integrity policies apply.