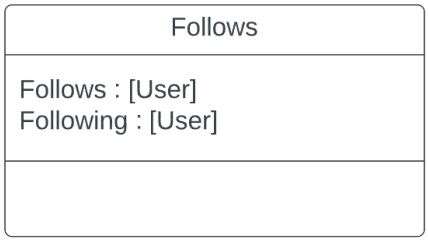
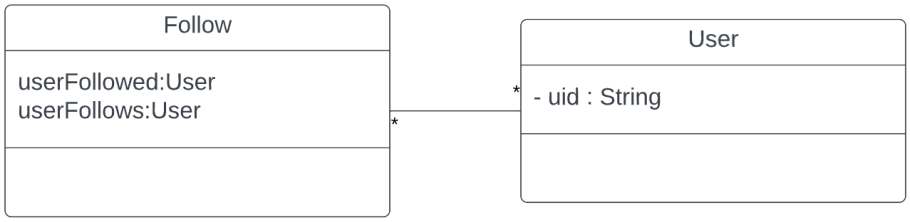
1. Follows

- This is an alternative to a bad design where there exists only one class that has 2 object arrays of User type. The cons of this approach are that there is no detail on which user follows which user. Since there is no implication of a user id and thereby can cause some confusion as to the recipients of the request. This design is not taken into consideration.

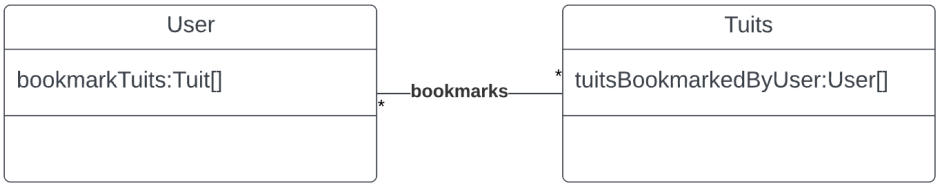


- The relationship between users and the users they follow describes a ***many-to-many*** relationship, that is, each user can track many users, and many users can follow each user. The pros of this design are that it's more extensible and separates the verb from the noun to make it clear where you want to know which user follows which user. This design is better.

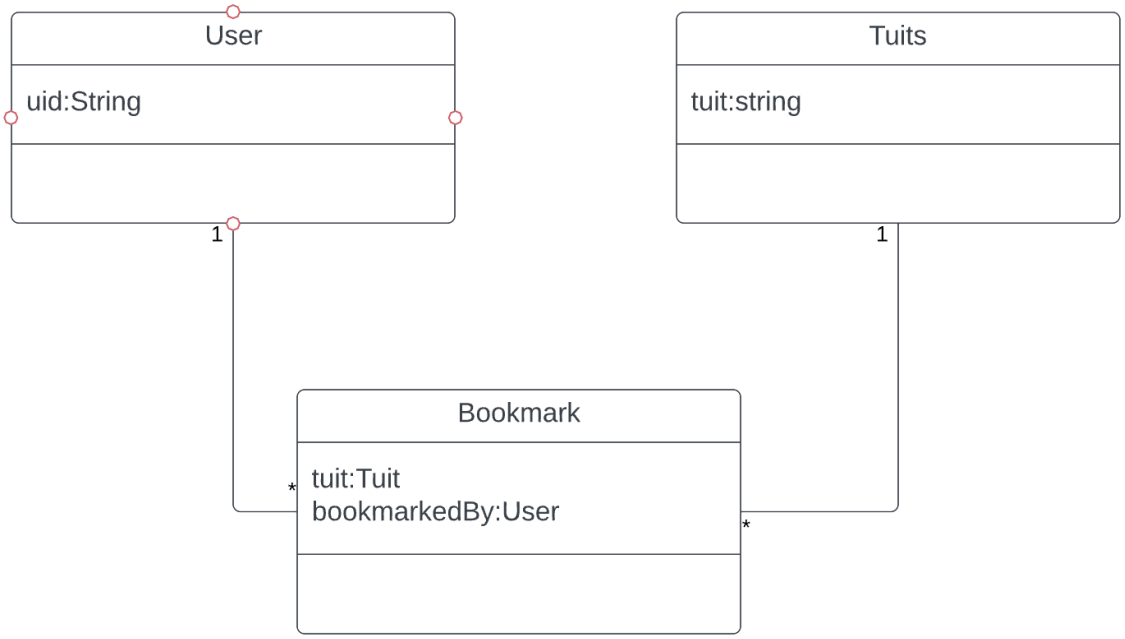


2. Bookmarks

- The relationship between users and the tuits they bookmark describes a ***many-to-many*** relationship, that is, each user can bookmark many tuits, and many users can bookmark each tuit. It doesn't really help.

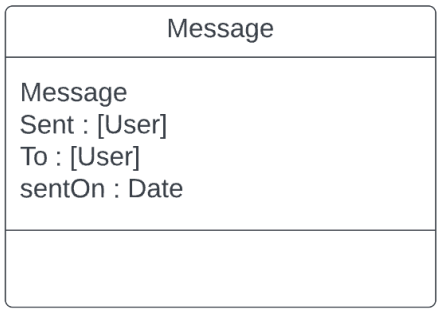


- Use a mapping table to keep track of which user bookmarks which tuit. Again, this option is the most versatile but might complicate joining in a nonrelational database, but this complexity is less honorous than maintaining synchronous arrays. So we're going to settle on ***this alternative*** to implement the bookmarks Web service API.



3. Messages

- This is an alternative to a bad design where there exists only one class that has 2 object arrays of User type. The cons of this approach are that there is no detail on which user messages which user. Since there is no implication of a user id and thereby can cause some confusion as to the recipients of the message. This design is not taken into consideration.



- The relationship between users and the users they message describes a ***many-to-many*** relationship, that is, each user can message many users, and many users can follow each user. The pros of this design are that it's more extensible and separates the verb from the noun to make it clear where you want to know which user messages which user. This design is better.