Concept: Upvoting
Purpose: For a user to bookmark freets to view at a later time
Structure:

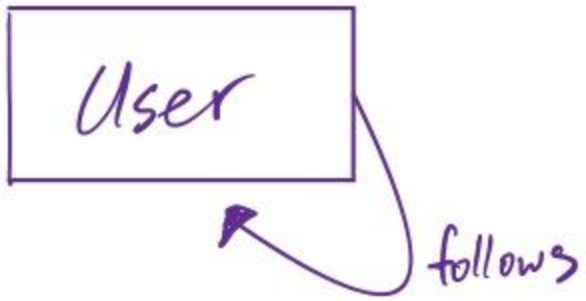
Since a user can upvote multiple freets, and a freet can be upvoted by multiple users, the multiplicities are 0 or more.

Actions:

upvote(u: User, f: Freet)
u.upvoted = u.upvoted + f
downvote(u: User, f: Freet)
u.upvoted = u.upvoted - f

Operational principle:
After a user u upvotes a freet f (with action upvote(u, f)), f will be in u.upvoted, which is u's upvoted freets; this way, user u can view their upvoted freets and find f at some later time. After a user u downvotes a freet f (with action downvote(u, f)) or hasn't upvoted a freet f, f will not be in u.upvoted, so user u cannot find f when they view their upvoted freets.

Concept: Following
Purpose: For a user to quickly find other users
Structure:



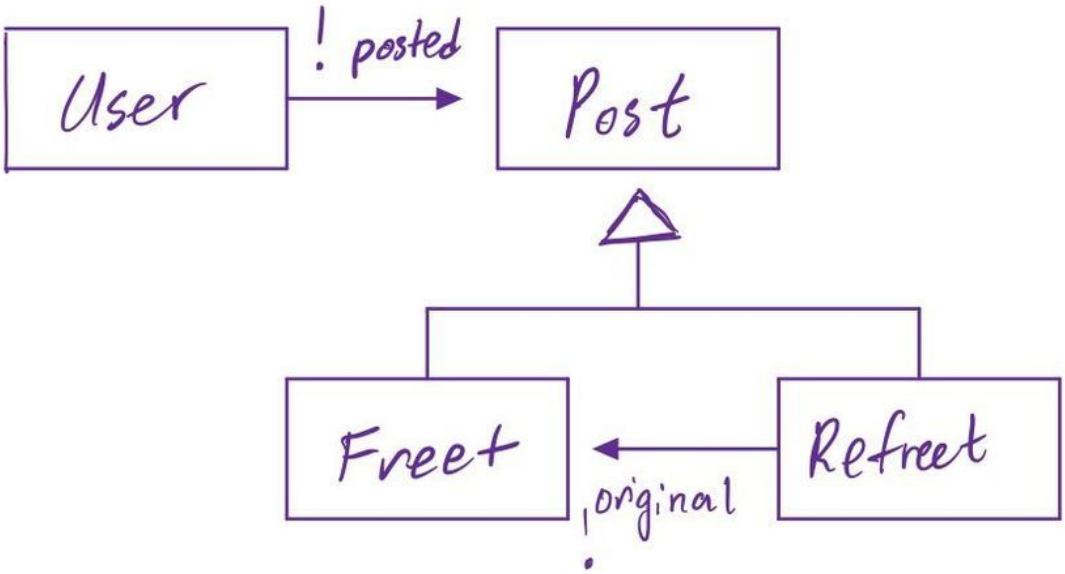
Since a user can follow multiple users, and a user can be followed by multiple users their multiplicities are 0 or more.

Constraits:
no iden[User] & follows
Actions:

follow(u: User, v: User)
u.follows = u.follows + v
unfollow(a: User, b: User)
u.follows = u.follows - v

Operational principle:
After a user u follows another user v (with action follow(u, f)), user u will be in u.follows; this way, user u can find the user v in u.follows (with an observable action) and use it to view some information about user v (by a search query for example). After a user u unfollows another user v (with action unfollow(u, f)), or hasn't followed user v, user v will not be in u.follows; this way, user u cannot find the user v in u.follows (so user u will have to find user v some other way if user u needs information about user v).

Concept: Refreeting
Purpose: For a user to share another user's freet
Structure:



Since a post needs to have a single author, there is a 1 multiplicity at User from User -> Post. Since a refreet must have a single original freet which it is sharing, there is a 1 multiplicity at Refreet from Refreet -> Freet.

Constraint:
all r : Refreet | no r.~posted & r.original.~posted (user cannot refreet their own freet)

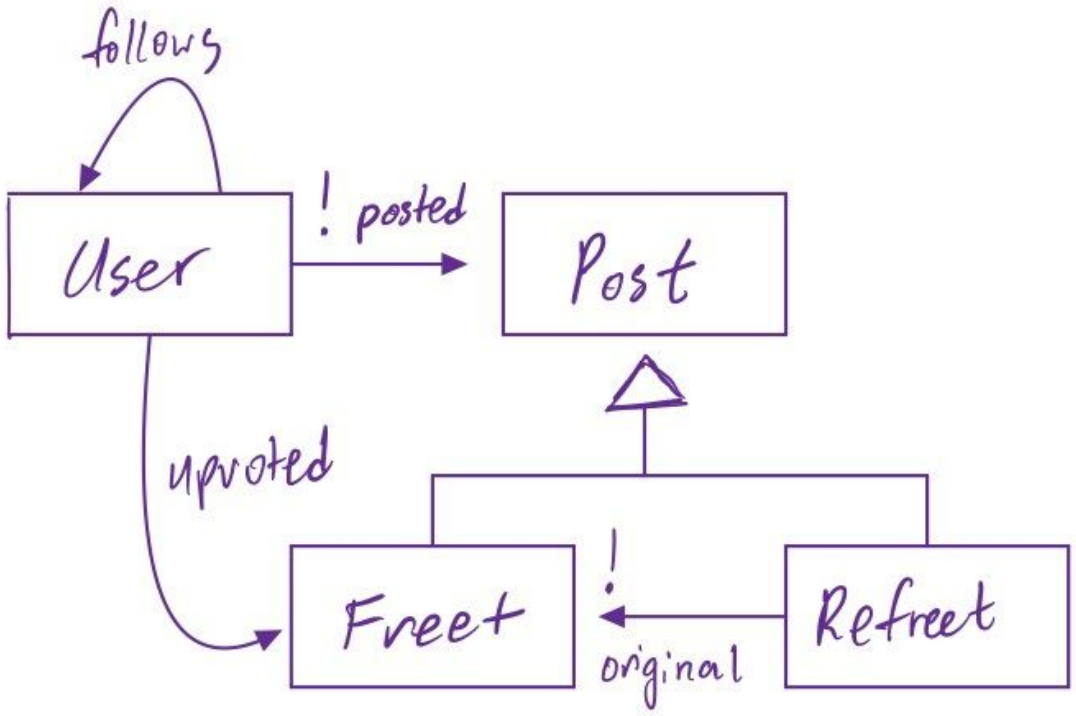
Actions:
refreet(u: User, f: Freet)
r = new Refreet
r.original = f
u.posted = u.posted + r
unrefreet(u: User, r: Refreet)

delete r from Refreets

r.original = empty set
u.posted = u.posted - r

Operational principle:
After a user u refreets a freet f (with action refreet(u, f, r)), r will be in u.posted; this way, the other users can see freet f through user u's refreet r (so freet f can be found from the posts of its original author and the posts of user u). After a user u unrefreets a freet f (with action unrefreet(u, f, r)) or has never refreeted freet f, then r is not in u.posted, so a freet f cannot be found from the posts of user u.

The whole structure of the application is as follows:



Design Decisions

1. A user cannot refreet their own freet.

I chose to disallow users from refreeting their own freet because I defined the purpose of refreeting to be sharing other users' posts. Since a freet's purpose is for a user to share their thoughts, allowing a user to refreet their own freets would allow a user to reshare the same exact thoughts, which is redundant. If refreeting and freeting are understood as similar actions, usrs could be confused as to what the purpose of a refreet is. One limitation to this idea is that users may want to reshare their own thoughts without having to create a duplicate freet, which they would't be able to do.

The other alternative to this decision would be to allow users to share their own freets. One advantage of this alternative is that users could perceive refreets as emphasized freets, so they might want to create refreets of their own freets to highlight their own important thoughts. This would mean, however, that users may ignore non-refreeted freets because they assume that they aren't as important as refreeted freets; this could force users to have to freet and refreet - which is an extra step - everytime they want to share an important thought. In addition, if a particular user has refreeted many of their own freets, then other users may be confused as to which of their refreets is actually meant to be emphasized. Another advantage of this alternative would be that if there is an old freet that a user wants to reiterate at a later time, the user can refreet this old freet instead of rewriting it. I chose against this idea because freets are short and creating another freet with some link to the old freet would add emphasis, which is better at reiterating an idea.

2. A user can delete an upvote.

I chose to allow users to delete an upvote because I defined the purpose of upvoting to be bookmarking a freet to view at a later time. Since user preferences change and upvoting is mostly for their efficiency, it is beneficial to allow users to control what is and isn't in their bookmarked posts. A disadvantage to this method is that if a user wants to get rid of a bookmarked post, they will need to explicitly delete it, which may annoy the user.

The other alternative would be disallow users to delete an upvote for a freet after they have already upvoted a freet. The advantage of this would be that a record of all the upvoted freets would be kept for the user and they wouldn't need to delete anything since that option isn't available. I chose against this design because I think it's more beneficial for users to have a few upvoted freets that they can easily skim through them to refind the ones they want. A common use case would be to show a freet to a friend at a later time, after which the user will probably not want to keep it bookmarked.

3. If a freet f is modified by the original author, all the refreets for f will remain.

I chose to keep the refreets of modified freets because if a user refreets a freet, then they show trust in the original author. If a refreet was deleted everytime a freet was modified, then users might be annoyed at having to keep refreeting to - what might seem to them - the same freet. One limitation to this idea would be that the original freet that has been refreeted has a

dramatically different idea than the modified freet, which would mean that users are sharing freets that they might not want to be sharing.

The other alternative would be to delete the refreets of a freet that has been modified. I chose against this idea because I think that most modifications to freets would be typos or other small errors that the users who refreeted the freet wouldn't care about. In addition, if a user wants to still refreet the modified freet, this alternative would require the user to find the modified freet, which could require the user to scan through all other freets.

4. If a user is deleted, then all the refreets associated with the user's freets will be deleted.

I chose to delete the refreets of a deleted user's freets because the deletion of an account signifies that the user does not want to use the service anymore, so their freets, which are their thoughts, should not be visible. Since the deleted user's thoughts no longer exist, other users should not be allowed to share these deleted thoughts. This highlights how refreeting is a sharing of another user's thoughts rather than authoring a similar thought. One limitation to this is that a user's account may be accidentally deleted, which means a lot of their data is deleted.

The other alternative would be to keep the refreets and their associated freets. An advantage might be that a refreet signifies that the user agrees with a particular freet, so the author of the refreet would expect their refreets, which they own, to remain because they still agree with the particular freet. I chose against this alternative because if another user's thought no longer exists, then there cannot exist a sharing of the nonexistent thought.

Social / Ethical Implications

1. Modifying freets may misrepresent people's original refreets

I chose to keep the refreets of a freet that has been modified which may misrepresent people's original intentions of refreeting the freet. For example, if a freet f represented some idea x that a user u agreed with refreeted, then f can later be modified to represent some idea y; now, user u's refreet still remains but user u might not agree with idea y. This may be mitigated by notifying the user that a freet corresponding to their refreet has been modified, so they can redecide if they still want to refreet.

2. Allowing users to delete their upvotes allows them to change their thoughts

I chose to allow users to delete their upvotes which allows users to delete upvotes for freets they may no longer agree with. If users were not able to delete upvotes for freets, then they would also be forced to see freets that they bookmarked from previously that they may no longer want to see, which is unethical to force on the user. Since upvotes are also public to other users, this also allows users to delete their upvotes if they don't want other users to see them; this gives users flexibility in case they accidentally upvote something they don't want

others to see but it also might pressure them to delete upvotes so that others view them a certain way.

3. Deleting the refreets associated with the freets of a deleted user may lift consequences from some users

I chose to delete the refreets associated with the freets of a deleted user which may allow the users who refreeted to avoid consequences if the deleted user was deleted by a higher authority. For example, if there was a user whose freets violated the terms and conditions and that user was forcefully deleted, then the refreets of those freets would also be automatically deleted, so the other users shared the unallowed freets wouldn't face any consequences. A way to mitigate this would be to warn users who refreeted the unallowed freets and use a striking system; an example of such a striking system would be Instagram's system that has consequences that get harsher with more warnings a user gets.