Trello - Simplified Social Media Platform

Overview

Trello is a simplified version of a social media platform where users can:

- **Post tweets** (messages), referred to as "trello tweets."
- **Follow** or **unfollow** other users, which dictates which users' posts appear in their news feed.
- **View the 10 most recent messages** in their news feed, which includes the tweets of the users they follow as well as their own tweets.

The objective is to create a platform that supports posting, following, and viewing a dynamic feed of tweets, with functionality to sort and limit the number of tweets visible to users.

Objectives

The **Trello** class (which serves as the main controller) is implemented with the following key objectives:

1. Trello()

- **Purpose**: Initialize the Trello object. This constructor sets up the user management system and prepares the data structures required for managing users, following relationships, and tweets.

2. postTweet(int userId, int tweetId)

- **Purpose**: Allows a user to post a tweet with a unique tweetId.
- **Process**: A Tweet object is created with the user's ID, the provided tweet ID, and an auto-incrementing timestamp. The tweet is then added to the user's collection of tweets, and the timestamp is generated by incrementing a global counter (timeCounter).

3. getNewsFeed(int userId)

- **Purpose**: Retrieves the 10 most recent tweet IDs in the user's news feed.
- **Process**: This method collects tweets from the users the given user follows. The tweets are then sorted by timestamp (most recent first). Finally, the method returns the IDs of the 10 most recent tweets (or fewer if there are less than 10).

4. follow(int followerId, int followeeId)

- **Purpose**: Allows one user to follow another.
- **Process**: The follower's followed list is updated to include the followeeId, which means the follower will now see the followee's tweets in their feed.

5. unfollow(int followerId, int followeeId)

- **Purpose**: Allows one user to unfollow another.
- **Process**: The follower's followed list is updated to remove the followeeId, meaning the follower will no longer see the followee's tweets in their feed.

Classes

To better understand how the project works, let's break down the primary classes and their responsibilities.

1. Tweet Class

The Tweet class represents a single tweet.

- **Attributes**:
  - tweetId: A unique identifier for the tweet.
  - userId: The ID of the user who posted the tweet.
  - timestamp: The timestamp when the tweet was posted.

Constructor: Initializes the tweetId, userId, and timestamp of the tweet.

Getters: Provides access to the tweet ID, user ID, and timestamp.

2. User Class

The User class represents a single user on the platform.

Attributes:

- userId: A unique identifier for the user.
- followed: A list of users this user follows. It stores the IDs of all users the user follows (including themselves by default).
- tweets: A list of tweets that the user has posted.

Methods:

- **follow(id)**: Adds a user ID to the followed list.
- **unfollow(id)**: Removes a user ID from the followed list (but doesn't allow the user to unfollow themselves).
- **postTweet(tweet)**: Adds a tweet to the user's list of tweets.
- **getTweets()**: Returns the list of tweets posted by the user.
- **getFollowed()**: Returns the list of users the current user follows.

3. UserManager Class

The UserManager class manages the users.

- **Attributes**:
    - userList: A list that stores all the users.

Methods:

- **createOrGetUser(id)**: Checks if a user already exists by the provided userId. If not, it creates a new user and adds them to the list.
- **getUser(id)**: Retrieves a user by their unique userId.

4. Trello Class

The Trello class serves as the entry point for interacting with the platform.

Attributes:

- userManager: Manages user creation and retrieval.
- timeCounter: A static variable used to generate unique timestamps for tweets.

Methods:

- **postTweet()**: Posts a tweet for the user by creating a Tweet object.
- **follow()**: Adds a user to another user's followed list.
- **unfollow()**: Removes a user from another user's followed list.
- **getNewsFeed()**: Retrieves the 10 most recent tweets from the users the current user follows, sorted by timestamp.

Flow of Operations

The sequence of operations within the platform:

Posting a Tweet

1. A user calls the postTweet(userId, tweetId) method.
2. A new Tweet object is created with a unique tweet ID, user ID, and a timestamp.
3. The tweet is stored in the user's collection of tweets.

Following a User

1. A user calls the follow(followerId, followeeId) method.
2. The followeeId is added to the followed list of the follower.

Unfollowing a User

1. A user calls the unfollow(followerId, followeeId) method.
2. The followeeId is removed from the followed list of the follower.

Retrieving the News Feed

1. A user calls the getNewsFeed(userId) method.
2. Tweets from all the users the current user follows are collected.
3. The tweets are sorted in descending order based on their timestamp.
4. The top 10 most recent tweet IDs are returned.

Time Complexity Analysis

- **Posting a Tweet**: O(1)
  - The operation simply involves adding the tweet to the user's tweet list, which takes constant time.
- **Following/Unfollowing**: O(1)
  - The operation involves adding/removing a user from the followed list, which is a constant-time operation.
- **Getting News Feed**: O(n log n)
  - Sorting the tweets by timestamp requires O(n log n), where n is the total number of tweets the user follows.
  - Extracting the top 10 tweets is O(10), but this is constant.

Conclusion

This simplified social media platform implements basic functionalities such as posting tweets, following and unfollowing users, and retrieving the most recent tweets from a user's news feed. Through object-oriented design principles, the platform ensures that the code is modular, scalable, and efficient.

By using efficient data structures and algorithms, such as sorting tweets by timestamp, we ensure that the platform performs well even as the number of users and tweets grows. The operations are optimized for both simplicity and speed, making it easy to extend and maintain in future versions.