CSE 102L Computer Programming Laboratory – Exercise 10

Due Date: 23:59 Sunday June 25th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *Ex10_yourStudentNumber.java*. Submissions made after the deadline will not be accepted, be sure to submit your work before the due date and make sure to click turn in button. Your code will be automatically controlled, so be sure to have the same class, method, variable names as described here. Failure to do so may result in you receiving 0 from this exercise. **All classes should be written to a single Java file. In a single java file there can only be single public class. Do NOT use Inner Classes. Be careful naming your file. If your editor inserts the file into a package, remove that line from the file but do NOT delete the import statements.**

Write set of classes according to the following specifications. Declare all attributes as **private** if not requested otherwise and use camelCase formatting for attributes.

Classes

1. User

- Attributes:
    - id: int
    - username: String
    - email: String
    - followers: set of Users
    - following: set of Users
    - likedPosts: set of Posts
    - messages: a map from User to Queue of Messages
- Methods:
    - Constructor that takes *username* and *email*. Initializes attributes. Uses itself's ***hashCode()*** method to initialize *id* field.
    - Accessors and Mutators for *username* and *email*
    - Accessors for *followers, following,* and *likedPosts*
    - message(recipient: User, content: String): None – if the *messages* field doesn't **contain** *recipient* as key, **puts** an empty *list* with *recipient* as key. Repeats this process for *recipient* instance (pay attention to keys). Initialize a new Message instance with *this* and *content*. Add this Message instance to queue of recipient and *this*. Call read(*recipient*) method.
    - read(user: User): None – Displays all the messages from *user*.
    - follow(user: User): None – this method has multiple use case. Firstly, it follows the given user. Secondly, if its already following the user, unfollows user. (Don't forget to add or remove from corresponding sets)

    - like(post: Post): None – like follow method, likes the given post. If already liked, unlike. Calls the **likedBy()** method of *post*
    - post(content: String): Post – Initializes a new Post with *content* and returns it
    - comment(post: Post, content: String): Comment – Initializes a new Comment with *content* and returns it after calling **commentBy()** method of *post*.
    - Overrides equals method using the *id*s of *this* and given *object*.
    - Overrides hashCode method with using **Objects** class' **hash** method. Uses only *email* attribute to hash the object.

2. Message

- Attributes:
    - seen: boolean
    - dateSent: java.util.Date
    - content: String
    - sender: User
- Methods:
    - Constructor that takes *sender* and *content*. Initializes dateSent with current time and seen with false.
    - read(**reader**: User): String – if *sender* is **not** *reader*, sets seen as true. Displays a message "Sent at: " + dateSent, returns content
    - hasRead(): boolean

3. Post

- Attributes:
    - datePosted: java.util.Date
    - content: String
    - likes: set of Users
    - comments: a map from User to List of Comments
- Methods:
    - Constructor that takes *content*. Initializes *datePosted* with current time and initializes other attributes.
    - likedBy(user: User): boolean – **adds** the user to likes. If cannot add, removes *user* from likes. Returns the result of adding operation.
    - commentBy(user: User, comment: Comment): boolean – puts the given *comment* to user's list of comments.
    - getContent(): String – Displays a message "Posted at: " + datePosted, and returns the content
    - getComment(user: User, index: int): Comment – gets the comment from comments if user has any comments on post and if index doesn't exceed user's size of comments. Otherwise returns null
    - getCommentCount(): int – returns the **total** number of comments. (be careful)
    - getCommentCountByUser(user: User): int – returns the **total** number of comments for given user. (be careful)

4. Comment – child class of Post

5. SocialNetwork

- Attributes:
  - postsByUsers: a **static** map from Users to list of posts
- Methods:
  - register(username: String, email: String): User – instantiates a user object with given parameters and if *postsByUsers* doesn't have instantiated object, puts an empty list with user key and returns user. If the same user exists in map, returns null
  - post(user: User, content: String): Post – if given *user* is registered instantiates a new post with given content and adds to list in *postsByUsers* with user as key.
  - getUser(email: String): User – uses **hash()** method of **Objects**t to hash the given email. If hashed email is the same as user's id, returns user, if no user found in *postsByUsers*, returns null
  - getFeed(user: User): Set of posts – for the given user, adds the posts of everyone which user follows to a set and returns the set.
  - search(keyword: String): a map from Users to String – for every user in *postsByUsers* if their name contains the given keyword, puts the user with their name in a map and returns.
  - reverseMap(map: Map<K, V>): Map<V, Set<K>> - based on the method signature complete this method.
  - **All the methods In this class are static**