ASSIGNMENT: SOCIAL NETWORK SIMULATION

OBJECTIVE

Design and implement a simplified social network simulation in Java that allows users to create profiles, manage friendships, post content, and interact with other users' posts. The program must utilize loops, arrays, ArrayLists, inheritance, polymorphism, abstract classes, and interfaces.

REQUIREMENTS

1. User Profiles

- **Abstract Class `UserProfile`:**
  - **Attributes:**
    - `String username`
    - `String email`
    - `ArrayList<UserProfile> friends`
  - **Methods:**
    - `addFriend(UserProfile friend)`

        - `removeFriend(UserProfile friend)`
        - `viewFriends()`
- **Interface `Postable`:**
    - **Methods:**
        - `createPost(Post post)`
        - `viewTimeline()`

2. User Types

- **Class `StandardUser` extends `UserProfile` implements `Postable`:**
    - Can create text posts.
    - Can like and comment on posts.
- **Class `PremiumUser` extends `UserProfile` implements `Postable`:**
    - All features of `StandardUser`.
    - Can create groups.
    - Can send private messages.

3. Posts and Content

- **Abstract Class `Post`:**
    - **Attributes:**
        - `String content`
        - `Date timestamp`
        - `UserProfile author`
        - `ArrayList<Comment> comments`
        - `int likes`
    - **Methods:**

        - `addComment(Comment comment)`
        - `likePost()`
        - `displayPost()`
- **Subclasses of `Post`:**
    - `TextPost`
    - `ImagePost`
    - `VideoPost`
    - Each subclass may have additional attributes (e.g., `ImagePost` might have an image URL).

4. Friend List Management

- Utilize `ArrayList<UserProfile>` to manage friends.
- Implement search functionality to find friends by username.

5. Groups

- **Class `Group`:**
    - **Attributes:**
        - `String groupName`
        - `PremiumUser admin`
        - `ArrayList<UserProfile> members`
        - `ArrayList<Post> groupPosts`
    - **Methods:**
        - `addMember(UserProfile user)`
        - `removeMember(UserProfile user)`
        - `postToGroup(Post post)`

6. Interfaces and Abstract Classes

- Use interfaces (`Postable`) to define behaviors that can be implemented differently by various classes.
- Use abstract classes (`UserProfile`, `Post`) to provide base functionalities and enforce a common structure.

7. Data Storage

- Store users in an `ArrayList<UserProfile>`.
- Store posts in an `ArrayList<Post>`.
- Use arrays where appropriate for fixed-size data.

8. Interaction

- Users can like and comment on any post.
- Implement loops to iterate over posts and comments for display purposes.

9. Advanced Features (Optional for Extra Credit)

- **Messaging System:**
  - Implement private messaging between users.
  - **Class `Message`:**
    - `String content`
    - `UserProfile sender`
    - `UserProfile receiver`
    - `Date timestamp`
- **Notification System:**
  - Notify users of new friend requests, messages, likes, and comments.

DELIVERABLES

1. **Source Code:**
    - Well-organized and commented Java code files.
2. **README File:**
    - Instructions on how to compile and run the program.
    - Overview of the program's features and how to use them.
3. **UML Diagrams:**
    - Class diagrams showing inheritance and associations.

EVALUATION CRITERIA

- **Functionality:**
    - Correct implementation of all specified features.
    - Proper use of loops, arrays, and ArrayLists.
- **Object-Oriented Principles:**
    - Effective use of inheritance and polymorphism.
    - Appropriate use of abstract classes and interfaces.
- **Code Quality:**
    - Readability and organization.
    - Meaningful variable and method names.
    - Error handling and input validation.
- **Creativity:**
    - Additional features beyond the requirements.
    - User-friendly interface (console-based or GUI).

GETTING STARTED

Below is a brief guide to help you start the assignment.

1. Set Up Your Project

- Create a new Java project in your preferred IDE (e.g., Eclipse, IntelliJ IDEA).

2. Define Abstract Classes and Interfaces

- `UserProfile` Abstract Class

```java
public abstract class UserProfile {
    protected String username;
    protected String email;
    protected ArrayList<UserProfile> friends;

    public UserProfile(String username, String email) {
        this.username = username;
        this.email = email;
        this.friends = new ArrayList<>();
    }

    public void addFriend(UserProfile friend) { /* ... */ }
    public void removeFriend(UserProfile friend) { /* ... */ }
    public void viewFriends() { /* ... */ }
}
```

- `Postable` Interface

```java
public interface Postable {
    void createPost(Post post);
    void viewTimeline();
}
```

3. Implement User Types

- `StandardUser` Class

```java
public class StandardUser extends UserProfile implements Postable {
    private ArrayList<Post> timeline;

    public StandardUser(String username, String email) {
        super(username, email);
        this.timeline = new ArrayList<>();
    }

    @Override
    public void createPost(Post post) { /* ... */ }

    @Override
    public void viewTimeline() { /* ... */ }
}
```

- `PremiumUser` Class

```java
public class PremiumUser extends UserProfile implements Postable {
    private ArrayList<Post> timeline;
    private ArrayList<Group> groups;

    public PremiumUser(String username, String email) {
        super(username, email);
        this.timeline = new ArrayList<>();
        this.groups = new ArrayList<>();
    }

    public void createGroup(String groupName) { /* ... */ }

    @Override
    public void createPost(Post post) { /* ... */ }

    @Override
    public void viewTimeline() { /* ... */ }
}
```

4. Implement Posts and Content

- `Post` Abstract Class

```java
public abstract class Post {
    protected String content;
    protected Date timestamp;
    protected UserProfile author;
    protected ArrayList<Comment> comments;
    protected int likes;

    public Post(String content, UserProfile author) {
        this.content = content;
        this.author = author;
        this.timestamp = new Date();
        this.comments = new ArrayList<>();
        this.likes = 0;
    }

    public void addComment(Comment comment) { /* ... */ }
    public void likePost() { /* ... */ }
    public abstract void displayPost();
}
```

- `TextPost` Class

```java
public class TextPost extends Post {
    public TextPost(String content, UserProfile author) {
        super(content, author);
    }

    @Override
    public void displayPost() { /* ... */ }
}
```

5. Build the Main Application

- `SocialNetworkApp` Class

```java
public class SocialNetworkApp {
    private ArrayList<UserProfile> users;

    public SocialNetworkApp() {
        this.users = new ArrayList<>();
    }

    public void registerUser(UserProfile user) { /* ... */ }
    public UserProfile login(String username, String email) { /* ... */ }
    public void displayAllUsers() { /* ... */ }
    public static void main(String[] args) { /* ... */ }
}
```

- ○

SUBMISSION

- Compress your project folder into a ZIP file.
- Ensure that all source code is included, zip it with your name and mat number.