Hoja de Problemas 2

Tema 2: Introducción a la Programación Orientada a Objetos

Nivel del ejercicio : (★) básico, (♣) medio, (♠) avanzado, (◇) examen

1. (◇) En la red social *Twitter*, cada usuario es propietario de una cuenta (*UserAccount*) en la que, básicamente, se especifica un *alias* (que cumple las funciones de identificador único) y un *email* de contacto. En la cuenta, además, se incluye el conjunto de *tweets* que el propietario va publicando a lo largo del tiempo.

   Como la cantidad de mensajes que maneja la red es inmensa, una característica original de *Twitter* es que cada usuario puede seleccionar la información que le interesa recibir. De esta manera, el propietario de una *UserAccount* puede convertirse en seguidor (*follower*) de otros usuarios, mostrando su interés en los *tweets* que ellos publiquen. Así, cada vez que un usuario publica un *tweet*, éste es incluido en el *timeline* de la *UserAccount* de cada uno de sus *followers* (es decir, el *timeline* se corresponde con el conjunto de *tweets* recibidos).

   Por otra parte, como se ha comentado anteriormente, la unidad básica de información se denomina *Tweet*. Un *Tweet* es creado en un instante de tiempo concreto (*time*), contiene un mensaje (*message*) con un máximo de 140 caracteres de longitud y es publicado por un usuario (conocido como *sender*). Además, existen dos tipos de *Tweet* especiales:

   - *DirectMessage*: Los mensajes directos son *Tweets* que permiten comunicarse, de manera privada, a dos usuarios dentro de la red. Estos *DirectMessage* son como *Tweets* ya que contienen un mensaje (*message*), son publicados por un emisor (*sender*) y son creados en un instante de tiempo determinado (*time*); *la única diferencia* es que incluyen a otro usuario como receptor (*receiver*) del *tweet*.
   - *Retweet*: Cuando un usuario lee un *tweet* interesante que le ha llegado a su *timeline*, y quiere reenviarlo a su lista de *followers*, crea un *retweet*. Este *Retweet* es como un *Tweet*, es decir, el usuario que lo publica (*sender*) puede poner un mensaje (*message*) y lo crea en un tiempo determinado (*time*); *la única diferencia* es que el *Retweet* incluye una referencia al *Tweet* que se reenvía.

   En la figura 1 se puede ver un modelo UML reducido del problema:

   En base a estas especificaciones se solicita que:

   (a) Implemente la clase *UserAccount* y su constructor. Incluya todos sus atributos (*alias, email, tweets, followers, timeline*) y establezca la visibilidad adecuada.

   (b) Implemente las clases *Tweet*, *Retweet* y *DirectMessage* escogiendo la jerarquía más adecuada. Añada los atributos que se especifican en el enunciado y establezca su visibilidad. Implemente los constructores de estas clases reutilizando los constructores del padre.

   (c) Reutilice todo el código que pueda. Si necesita un tipo para el atributo *time*, se recomienda utilizar la clase *Date* de *JAVA*. Recuerde que la clase *String* define el método `public int length()`.

   (d) Implemente, en *UserAccount*, un método que permita a un usuario seguir a otro:

```
public void follow(UserAccount user2)
```

Al ejecutar "`user.follow(user2)`", el usuario `user` se convertirá en *follower* de `user2`.

(e) Implemente, en *UserAccount*, un método que permita a un usuario publicar un *Tweet*:

```
public void tweet(Tweet tweet1)
```

Después de ejecutar el método "`user.tweet(tweet1)`", se deberá actualizar adecuadamente el atributo *tweets* de `user`. Además, todos los *followers* de `user` habrán recibido el `tweet1` en su *timeline*.

(f) Implemente el método `toString` en todas tres clases, reutilizando al máximo todo el código disponible.

(g) Añada, si lo necesita, todos los métodos auxiliares que crea convenientes para estructurar todo el código.

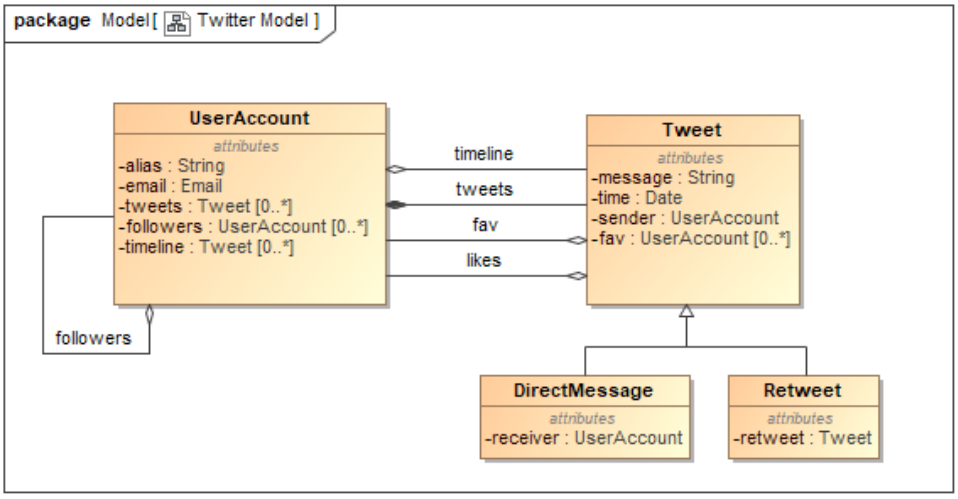

Figura 1: Modelo UML de Twitter