UNIVERSIDAD NACIONAL DE INGENIERIA

Facultad de Ingeniería Industrial y de Sistemas

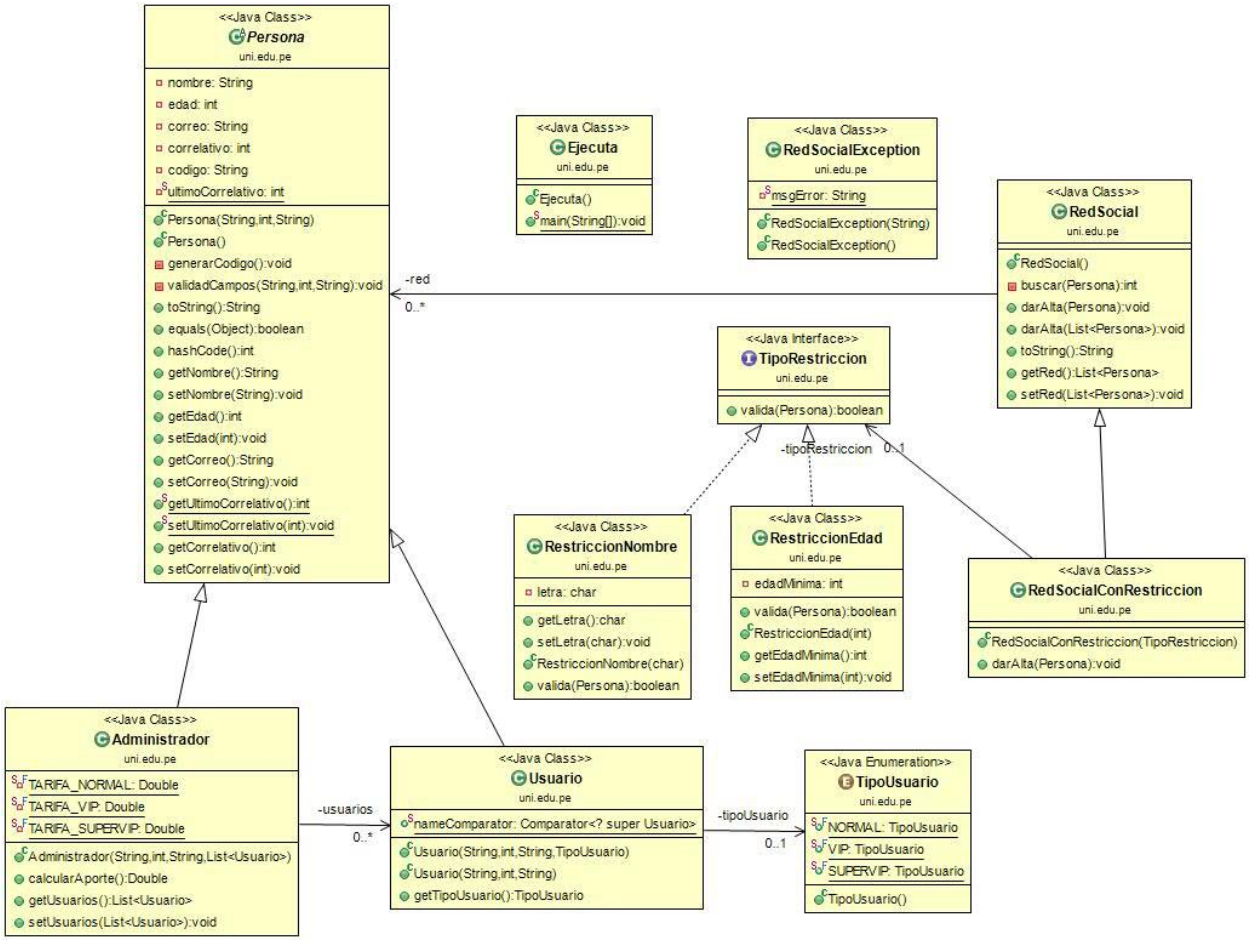
PRACTICA CALIFICADA II

Curso : Programación orientada a objetos
Código : SI302V
Alumno :

CICLO: 2022-I

Caso: Fiisbook

"Fiisbook" es una red social creada por alumnos de la FIIS. Esta red social está teniendo bastante acogida y se requiere gestionar los usuarios de la misma, para ello se ha elaborado el siguiente diagrama de clases:



Para desarrollar esta funcionalidad se tiene en cuenta las siguientes consideraciones:

1. (1 punto) La clase **RedSocialException** es una excepción no chequeada que se utilizará para tratar las distintas situaciones excepcionales que puedan surgir (siempre que se pida lanzar una excepción se debe enviar un mensaje explicativo).

2. (2.5 puntos) **Clase Persona**

La clase Persona es una clase abstracta que mantiene la información sobre un usuario de la red social. Los atributos que tiene son su **nombre** (String), su **edad**(int), **correo** (String), **correlativo**

(int), **código** (String). Adicionalmente se cuenta con un atributo autogenerado estático **ultimoCorrelativo** que comienza en 10001:

Adicionalmente, la clase deberá proporcionar los métodos necesarios para:

- Encapsular los atributos.
- Constructores que permitan actualizar inicializar el atributo **correlativo** (con el valor actual del **ultimoCorrelativo**), aumentar el atributo de clase **ultimoCorrelativo** y generar el **código (**para esto usuará el método **void generarCodigo()** que forma el código concatenando el prefijo 'U' o 'A' al **correlativo,** dependiendo de si se trata de un usuario o un administrador respectivamente). El constructor sin argumentos realiza sólo realiza las inicializaciones descritas en las líneas previas. Adicionalmente se **proveerá** un **constructor** que reciba los valores del **nombre**, **edad** y el **correo**. Si el **nombre** o el **correo** son nulos o si la edad es un número negativo se lanzará una excepción del tipo **RedSocialException**.
- El método **public String toString()** muestra el **nombre**, **codigo** y **edad** separados por espacios en blanco. El correo se presentará también, encerrado entre paréntesis. Por ejemplo:

  Roland A10009 12(Roland@contoso.com)
- Redefinir los métodos equals y hashCode de modo que dos objetos de la clase Persona sean iguales sin coinciden su nombre y su email. No se tendrá en cuenta las diferencias por mayúsculas o minúsculas para el nombre (ni los espacios en blanco al inicio o al final)

3. (1.5 puntos) **Clase Usuario**

   La clase **Usuario** es una clase hija de la clase Persona que contiene un atributo **tipoUsuario**, el cual es un **enum** con los siguientes valores (**NORMAL, VIP y SUPERVIP**). Adicionalmente posee un **comparador** (Comparator) **nameComparator** que ayuda a ordenar los alumnos en base a su nombre, sin tener en cuenta las mayúsculas y minúsculas.
   Adicionalmente posee 2 constructores:
   - Un constructor que recibe el **nombre, edad, correo y tipoUsuario** que permite inicializar los atributos **nombre, edad y correo** en base a la lógica del padre e inicializar al atributo **tipoUsuario** en base al valor ingresado para este parámetro.
   - Un constructor que recibe el **nombre, edad, correo.** En este caso el valor del atributo **tipoUsuario** será asignado a **NORMAL** por defecto.

4. (2 puntos) **Clase Administrador**

   La clase **Administrador** es una clase hija de la clase Persona que contiene un atributo **usuarios (List<Usuario))** que corresponde a la relación de usuarios que han sido dados de alta por él.
   - Posee un constructor que recibe el **nombre, edad, correo** y la lista de usuarios que han sido inscritos por él **(usuarios)** que permite inicializar los atributos**.**
   - El método **Double calcularAporte ()** sirve para calcular el aporte que el administrador a obtenido por cada usuario inscrito. Los valores aportados en soles por tipos de usuario son: **NORMAL:** 5, **VIP:** 10 y **SUPERVIP:** 25.

5. (3.5 puntos) Clase **RedSocial**

Esta clase almacenará la información de los usuarios de la red social en una lista de personas **red (List<Persona>**). La clase dispondrá de los métodos necesario para:

- Definir el método **privado int buscar (Persona)** que determina si la persona pasada como argumento está en el listado y devuelve la posición en la que se encuentra.
  Si la persona no está en el listado devuelve -1.

- Dar de alta a un nuevo usuario (**void darAlta(Persona p)**) que se pasa como argumento. Si la persona que se desea añadir ya está en la red, no se hará nada. Se lanzará una

excepción del tipo **RedSocialExceptipn** si el objeto **p** es nulo. Puedes usar el método buscar -antes implementado- para la implementación de este método.

- Se debe sobreescribir el método **toString()**. Se debe usar la clase StringBuilder. La información para cada usuario dado de alta se muestra en una línea diferente, que contendrá el nombre, el código, la edad y el email entre paréntesis. Ejemplo:

  Roland A10009 12(Roland@contoso.com)
  Miguel A10010 15(Miguel@contoso.com)

6. (1 punto) Interfaz **TipoRestriccion**

Esta interfaz especifica un método **public boolean valida(Persona p)** que sirve para conocer si la persona p cumple las restricciones establecidas por la red social para darse de alta

7. (2.5 puntos) Clase **RedSocialConRestriccion**

Esta clase se comporta como la clase RedSocial, con la diferencia que los usuarios deben cumplir una serie de restricciones antes de ser añadidos a la red. Cada objeto de esta clase tiene una variable de instancia **restricción (**cuyo tipo seria la interfaz **TipoRestriccion)** que servirá para aplicar las restricciones pertinentes. Se deben implementar los métodos necesarios para:

- Construir un objeto de la clase, dado el objeto de la clase **TipoRestriccion.**
- Redefinir el método **public void darAlta(Persona p)** para que sólo añada un nuevo usuario si cumple las restricciones definidas por el objeto **restricción.** En caso de que **p** sea nulo debe lanzarse una excepción del tipo **RedSocialException**.

8. (1.5 puntos) Clase **RestriccionEdad**
   Esta clase implementa la interfaz **TipoRestriccion**. Tiene una variable de instancia **edadMinima(int)**. Dispondrá de los siguientes métodos:

   - Un constructor al que se le pasará como argumento un valor entero con la edad e inicializará la edad mínima que un usuario debe tener para poder dase de alta en la red social. Recuérdese que una edad negativa no es correcta y debe lanzar una excepción **RedSocialException**
   - El método **boolean valida(Persona)** para que cualquier persona con una edad menor que la indicada por la **edadMinima** no sea válida.

9. (1.5 puntos) Clase **RestriccionNombre**
   Esta clase implementa la interfaz **TipoRestriccion.** Tiene una variable de instancia **letra (char).** Dispondrá de los siguientes métodos:

   - Un constructor al que se le pasará como argumento un caracter, e inicializará la variable letra por la que deben empezar los nombres de los usuarios que se podrán dar de alta.
   - El método **boolean valida(Persona)** para que cualquier persona cuyo nombre no empiece por el carácter exacto indicado por **letra** no sea válido.

10. (2.5 puntos) Ahora, cree una clase ejecutable **"Ejecuta"** que haga lo siguiente:

    - Genere 8 Usuarios y 3 administradores (tomar como referencia el siguiente cuadro):

Usuarios

Orden	nombre	edad	correo
1	Nestor	24	(nrafael1@contoso.com)
2	JUan	12	(JUan@contoso.com)
3	Marcos	6	(Marcos@contoso.com)

4	martin	9	(martin@contoso.com)
5	Roland	12	(Roland@contoso.com)
6	Miguel	15	(Miguel@contoso.com)
7	julian	6	(julian@contoso.com)
8	Carla	9	(Carla@contoso.com)

Administradores

Orden	nombre	edad	correo	Orden Usuarios
1	Luis	12	(Luis@contoso.com)	1,4,5
2	santiago	15	(santiago@contoso.com)	2,6
3	jaime	17	(jaime@contoso.com)	3,7,8

- Dar de alta a los administradores a la **RedSocial** y los muestre:

  #### Imprimiendo los administradores #####
  Luis A10009 12(Luis@contoso.com)
  santiago A10010 15(santiago@contoso.com)
  jaime A10011 17(jaime@contoso.com)

- Calcule el aporte de los administradores a la red social y los muestre.
  Nótese el formato a mostrar debe incluir el nombre del administrador, monto del aporte y la lista de alumnos ordenados por nombre (en orden alfabético y sin distinguir mayúsculas y minúsculas)

  ##### Aporte a la red social #####
  Aporte a la red social del administrador: Luis es: 40.0 soles, por los usuarios: martin Nestor Roland
  Aporte a la red social del administrador: santiago es: 30.0 soles, por los usuarios: JUan Miguel
  Aporte a la red social del administrador: jaime es: 25.0 soles, por los usuarios: Carla julian Marcos

- Dar de alta a los alumnos a la **RedSocialConRestricción** con **TipoRestriccion** por edad (**12**) e imprima el contenido del objeto **RedSocialConRestriccion**

  #### Restringiendo por edad #####
  Nestor U10001 24(nrafael1@contoso.com)
  JUan U10002 12(JUan@contoso.com)
  Roland U10005 12(Roland@contoso.com)
  Miguel U10006 15(Miguel@contoso.com)

- Dar de alta a los alumnos a la **RedSocialConRestricción** con **TipoRestriccion** por nombre **('M')** e imprima el contenido del objeto **RedSocialConRestriccion**

  #### Restringiendo por nombre #####
  Marcos U10003 6(Marcos@contoso.com)
  martin U10004 9(martin@contoso.com)
  Miguel U10006 15(Miguel@contoso.com)

- Imprima el último correlativo

11. (0.5) Suba la carpeta con el código fuente y el enunciado de esta práctica en un directorio privado en github llamado practica-2 y brinde acceso al profesor del curso.