Ejercicio Obligatorio. Sesión 3

1. Instrucciones

Este ejercicio deberá ser entregado **48 horas antes** de la siguiente clase de laboratorio.

Antes de la entrega, renombra los proyectos dentro de Eclipse (usando Refactor -> Rename) con :

nombre_apellido_1_apellido2_session3_task_dome
nombre_apellido_session3_task_sesion3_post en minúsculas y sin tildes.

Exporta los proyectos juntos a un fichero comprimido y súbelo a la tarea correspondiente en el Campus Virtual.

2. Ejercicio 1

A partir del proyecto Dome realizado en la Sesión 3:

1. **Validación de parámetros**. Añade la validación de parámetros que te parezca oportuno, lanzando una excepción en caso erróneo. En los setters de los atributos, reemplaza la validación existente por otra que lance una excepción de tipo *IllegalArgumentException*.
   En el caso de cadenas, comprueba que la cadena existe y que no esté formada únicamente por espacios en blanco. Realiza las pruebas de robustez para comprobar que son correctos.
2. Añade o finaliza las pruebas de la clase MediaLibrary, para los métodos que tienes implementados **salvo el método list**. Crea getItems() para devolver una copia de la lista de ítems.
3. Añade a la clase MediaLibrary un **método *searchItem(Item theItem***) que busque en la lista un item cuyos datos coinciden con los del ítem recibido como parámetro y devuelva la posición que ocupa, si lo encuentra, o bien -1 si no lo ha encontrado. Realiza las pruebas unitarias.
4. Añade a la clase MediaLibrary un **método printResponsable(PrintStream out)** que imprima en el objeto out todos los responsables de los elementos que tienen propietario. En el caso de los CDs, el responsable es el artista. En el caso de los DVDs el responsable es el director. **No es necesario** realizar pruebas de este método.
5. Incluye en la aplicación la posibilidad de guardar también **videojuegos**, además de CDs y DVDs. Un videojuego (`VideoGame`) tiene las siguientes características:
   - Un título
   - Un autor (que será el responsable).
   - No se guarda la duración. *(Ojo!, no tendrá la propiedad playingTime)*
   - Propietario
   - Comentario
   - Número de jugadores
   - Una plataforma que podrá ser XBOX, PLAYSTATION o NINTENDO. *(Nota: deberás crear un tipo enumerado Platform* con los tres valores).

   Sus características se pueden imprimir en el objeto out con un método print.

Aplica la herencia y el polimorfismo al añadir esta nueva clase. Recuerda que la superclase debe contener únicamente los datos comunes de las subclases. Si algún dato ya no es común a todas las clases, debes bajarlo a las subclases que los posean.

6. Crea un constructor VideoGame(titulo,autor,jugadores,plataforma). Realiza las pruebas de funcionalidad y robustez de esta clase `VideoGame` para probar el constructor con JUnit. Utiliza @Before para crear un título, autor, plataforma y jugadores. **Añade estas pruebas a AllTest**.
7. Modifica las clases de prueba de los métodos de `MediaLibrary`, para contemplar objetos `VideoGame`. Añade las pruebas del método numberOfItemsOwned() si no lo has realizado. Deberá contemplar también objetos VideoGame.
8. Comprueba que el método list en MediaLibrary también muestra videogames en consola.

9. Ejecuta todos los test existentes.
10. Añade otro paquete al proyecto dome, llamado uo.mp.s3.dome.application, y crea una clase MediaPlayer que incluya el método estático `main` para poder ejecutar la aplicación (y no solo comprobar que los test funcionan correctamente). En el método `run()` se deben crear al menos 1 objeto de cada clase (CD, DVD, `VideoGame`) y un objeto de la clase `MediaLibrary` donde se añadan los ítems. Muestra el resultado de la ejecución de los métodos `numberOfItemsOwned`, printResponsable y list. Al ejecutar estos métodos pasa como objeto out la pantalla (System.out).

3. Ejercicio 2

Es necesario realizar primero, el diseño UML aunque no es necesario entregar el diagrama.

Se desea realizar una aplicación que gestione los ***post*** que se incluyen en una red social. Los usuarios tienen la posibilidad de incluir dos tipos de post: ***mensajes*** de texto y ***fotos***.

Un mensaje de texto consiste en una cadena en la que se almacena el ***mensaje***, mientras que la inserción de una foto implica guardar el ***nombre del fichero*** donde se almacena la foto y un ***título*** para la foto. Además, todos los post guardarán información de los ***likes*** que han recibido. Asimismo, se pueden introducir ***comentarios*** a los post, por lo que cada post guarda una lista con los comentarios asociados al mismo. Un comentario será almacenado como una cadena. Cada post lleva asociado también el identificador del ***usuario*** que lo ha creado.

Cuando se crea un post, se reciben los datos como parámetros salvo los likes que se inicializan a 0 y la lista de comentarios que se crea vacía.

Se desea crear un proyecto que realice la gestión de la red social en relación a los post que se pueden incluir, de forma que **la red** almacene todos los post y se puedan realizar las siguientes operaciones:

1- **addPost(Post post).** Esta operación añade el post a la red.
2- **print(PrintStream out)**. Esta operación imprime, en el objeto out que recibe como parámetro, los atributos de todos los post.
3- **findPostsByUser(String userId).** Recibe un usuario como parámetro y devuelve la lista de todos los post correspondientes a dicho usuario.

Lanza una excepción *IllegalArgumentException* si al crear el post el nombre de usuario, del fichero o el mensaje es *null* o cadena vacía.

Lanza una excepción *IllegalArgumentException* si al añadir un post se pasa *null*.

Realiza las **pruebas de funcionalidad y robustez** para asegurarte de que se crean bien los post (mensajes o fotos), que vuelcan la información correctamente (toString), así como las pruebas de addPost y findPostByUser de la red social.

El formato de los post sería:

Photo user: Usuario1 File: foto1 Title: titulo1 Likes: 0 Comments: [ ]
Message User: Usuario1 msg: Mensaje1 Likes: 0 Comments: [ ]

Nótese que en esta versión no es posible incluir comentarios ni likes, por lo que siempre serán vacíos.

Crea una clase para lanzar la aplicación que cree la red y añada un post de cada tipo para dos usuarios diferentes, busca imprime los post de un usuario.