Prácticas Java

T06 CrowdFernan

Introducción

En este tema vas a realizar:

- Refactorización de la clase *Usuario* de la que heredarán *Inversor*, *Promotor* y *Administrador*.
- Refactorización de la clase *Inversion* para que se relacione con un usuario *Inversor* e implemente una interfaz con métodos para aumentar y disminuir la inversión.
- Versión preliminar del patrón MVC de la aplicación.
- Actualización del diagrama UML de clases.
- Documentación mediante JavaDoc
- Publicación en GitHub del nuevo código en una nueva rama

Todo ello, siguiendo las recomendaciones de código limpio y controlando todos los posibles errores.

Recuerda que tienes una explicación general de toda la práctica en su conjunto en el documento “Práctica obligatoria 23-24 Explicación general – CrowdFernan.docx” en la Moodle (apartado “General”).

Refactorización de *Usuario* con herencia entre clases

Refactoriza la clase *Usuario* para que sea la superclase de la que hereden las clases *Inversor*, *Gestor* y *Administrador*.

Todos los atributos y métodos comunes irán en *Usuario* y los específicos, en las subclases.

La clase *GestionUsuario* que implementaste en el tema anterior con el array de usuarios, debe seguir funcionando, sólo que para ciertas tareas necesitarás comprobar si el tipo de usuario que se trate es de una de las subclases.

Refactorización de *Inversion*

En una inversión relacionabas un usuario con un proyecto. Seguirá todo igual, sólo que ahora se tiene que relacionar con un usuario de la subclase *Inversor*.

Haz que *Inversion* implemente una interfaz con los métodos *aumentaInversion* y *disminuyeInversion* para que se pueda aumentar o disminuir el capital aportado al proyecto por el inversor que se trate.

Prácticas Java

Versión preliminar MVC

En una primera aproximación al patrón Modelo Vista Controlador, simplemente vas a delegar cualquier contacto con el usuario que use la app (entrada o salida por consola) a las clases más altas (*Main* o *GestionApp*).

De esta forma, clases como *Usuario* o *GestionUsuario* en vez de mostrar mensajes por consola, retornarán un String que se encargarán de mostrar las clases más altas como se ha mencionado.

Diagrama de clases

Crea el diagrama UML de clases de tu aplicación. Puedes utilizar cualquier aplicación como, por ejemplo https://app.diagrams.net/

Guarda el diagrama resultante en png.

Documentación

Generar documentación de todas las funciones con *JavaDoc* y entregar el html resultante.

Añadir al documento que elaboraste en el tema anterior, la nueva funcionalidad que se ha desarrollado en esta práctica para que la documentación esté al día.

Sería recomendable añadir en el nombre la *versión del documento* (v2, v3, v4…) y dentro del mismo, *la fecha de la última actualización*.

Repositorio en GitHub

Añadir el nuevo código al repositorio del proyecto en GitHub, pero en una nueva rama.

Vamos a tener una rama por cada tema, así que, crea una rama con nombre **tema 6** que será la que apunte a los últimos cambios que vayas realizando en esta práctica.

Cuando vayas a entregar la práctica, fusiona la rama **tema 6** con la rama **master**. De esta forma, la rama master tendrá siempre los últimos cambios y mantendrás el resto de ramas que vayas creando para ver la evolución del código tema a tema.

Entrega de la práctica

La entrega de la práctica se hará en la Moodle en un archivo .zip con lo siguiente:

a. Documentación del programa a modo de manual de usuario actualizado. Especificar en el documento la URL del repositorio de GitHub.
b. Diagrama UML de clases en png.
c. Documentación en html generada con JavaDoc.
d. Carpeta src/ con los ficheros .java con el código.
e. Archivo .jar.