MÓDULO: PROGRAMACIÓN

PROYECTO FINAL

RA5	UD7. Lectura/Escritura De Información

Nombre del Proyecto: El ordenador inteligente

Usuario válido: admin

Contraseña válida: admin

Alumn@: Laudoiu Mihai Gabriel

FECHA LÍMITE DE ENTREGA:

26/05/2024 HORA: 23.59

LUGAR: Aula Virtual UT7. LECTURA/ESCRITURA DE INFORMACIÓN Exámenes Proyecto Final

PUNTUACIÓN: 8

MÍNIMO PARA APROBAR: 4

CONDICIONES INDISPENSABLES PARA LA CORRECCIÓN DE LA PRÁCTICA

Ante la mínima sospecha de plagio o de que no ha sido desarrollada por el alumn@ la práctica se puntuará a 0.

La práctica debe estar completamente libre de errores tanto sintácticos como de ejecución, esto quiere decir que se deben controlar todos las posibles excepciones (valores no permitidos, compatibilidad de datos, índices fuera de rango, valores vacíos, manejo de recursos…) si no fuera así se puntuará a 0.

Entregar todos los ficheros generados, incluidos los ficheros de datos.

Junto con el proyecto el alumn@ tiene que entregar un VIDEO (de baja calidad) explicativo de:

- **cómo se ha organizado la práctica, las clases, las estructuras de datos y de control más relevantes que ha usado, ficheros, ejecución de la aplicación probando todo tipo de casos incluidos casos límite, traza, etc.**
- **DURACIÓN MÁXIMA: 15 minutos. No se corregirá si lo supera**

https://hoymarketing.com/programas-grabar-video-tutoriales/

Objetivo de la Práctica

Crea una aplicación de consola personalizada, es decir, de creación propia, diferente a la del resto de los compañer@s, en la que hagas uso de todos los contenidos estudiados en el módulo a lo largo del curso.

Referencias de consultas (puedes usar las que quieras)

Curso Java

https://www.discoduroderoer.es/curso-java/
https://www.pildorasinformaticas.es/

Lectura y escritura de ficheros en Java

https://jarroba.com/lectura-escritura-ficheros-java-ejemplos/
https://www.youtube.com/watch?v=etQN4EfYN7k
https://www.youtube.com/watch?v=E0H4OzW2_1Y

Listas dinámicas

https://www.youtube.com/watch?v=q5a_QAWB7jU
https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=605:interface-list-clase-linkedlist-api-java-ejercicio-diferencias-entre-arraylist-y-linkedlist-codigo-cu00921c&catid=58&Itemid=180

Se pide

1. Crear un proyecto con nombre significativo pero que acabe con XXX (iniciales de tu nombre)
2. Crear los paquetes necesarios para organizar todas las clases que vas a utilizar.
3. Utilizar la estructura de clases que ya tienes implementada del tema de herencia. Sólo tienes que utilizar una subclase (ponte de acuerdo con tus compañer@s para repartíroslas, pues tienen que ser distintas para cada alumn@). Puedes hacer las modificaciones que creas oportunas siempre y cuando no cambies de clase.
4. Captura **TODAS** las excepciones que se puedan producir, de manera que la aplicación no se pare y se ejecute correctamente, mostrando información clara y concisa de lo que está pasando en cada momento. Realiza las pruebas adecuadas para asegurarte que así sea.
5. Crea una excepción personalizada y utilízala.
6. Crea una clase para usuarios de la aplicación.
7. Crea la clase no ejecutable de nombre ObjectOutputStreamXXX para sobrescribir la clase ObjectOutputStream de Java, que evitará que se cree la cabecera con metadatos que puedan hacer inaccesible el fichero para la lectura.
8. Crea la clase no ejecutable de nombre EscritorXXX que permita guardar los objetos de las clases en ficheros. Al crear un objeto de esta clase, se debe pedir por consola el nombre del fichero en el que se escribirá. Los métodos que se deben implementar en esta clase son:

    - **escribirXXX**

      Que escriba en el fichero un objeto de la clase pasado como parámetro. Tiene que devolver **1** si la operación de escritura tiene **éxito**; **-1** si ocurre alguna excepción. Debe comprobar si el fichero existe o no.

    - **escribirXXX**

      Que escriba en el fichero un conjunto de objetos de la clase, almacenados en una lista dinámica que se le pasará como parámetro. Tiene que devolver el **nº de objetos** escritos si la operación de escritura tiene éxito; **-1** si ocurre alguna excepción. Debe comprobar si el fichero existe o no.

9. Crea una clase no ejecutable de nombre LectorXXX que permita recuperar un conjunto de objetos de la clase desde el fichero. Se debe indicar el nombre del fichero a tratar por consola. Los métodos que debe ofrecer son los siguientes:

    - **leerXXX**

      Que lea del fichero todos los objetos de la clase y escriba su información por consola. Tiene que devolver el nº de objetos leídos si la operación de lectura ha tenido éxito; -1 si se ha producido alguna excepción. Debe comprobar si el fichero existe o no.

    - **leerXXX**

      Que lea del fichero todos los objetos de la clase y devuelva una lista dinámica con

todos ellos. Si el fichero estuviese vacío o se produjese alguna excepción la lista estará vacía. Debe comprobar si el fichero existe o no.

- **buscarXXX**

Que busque y recupere el objeto cuyo valor coincida con el indicado en el parámetro que le debes pasar, tú decides el parámetro por el que buscar, pero ten en cuenta que debe tener sentido y debe ser único. Devuelve el **objeto** de la clase si es que existe; **null** si no se encontró o se produce alguna excepción. Debe comprobar si el fichero existe o no.

10. La aplicación se iniciará con un login para que el usuario se identifique mediante un nombre de usuario y contraseña válidos, de tal manera que:

    ♣ Los datos de los usuarios se guardarán en un **fichero**.

    ♣ Si el usuario no existe se le debe dar la posibilidad de **registrarse**. Ten en cuenta que no puede haber dos nombres de usuario iguales. Las contraseñas sí podrán repetirse.

    ♣ Un usuario que ya existe podrá **modificar su contraseña,** pero no su nombre de usuario.

    ♣ Como máximo tendrá tres intentos para loguearse, si no acierta, la aplicación eliminará al usuario del fichero y a continuación se cerrará, teniendo que volver a registrarse de nuevo la próxima vez que acceda.

    ♣ Si las credenciales son correctas entonces podrá acceder al resto de la aplicación.

11. Una vez que el usuario se haya identificado correctamente, tendrá opciones para dar **altas, bajas, modificaciones, búsquedas** y obtener **listados** de los datos del fichero que contendrá objetos de tu clase personalizada.
12. Siempre debe permitir volver a las opciones anteriores o salir de la aplicación en cualquier momento.
13. La aplicación debe realizar alguna validación mediante expresión regular (RegEx) (comprobación de algún código, email, referencia, en definitiva algún patrón).
14. Utiliza listas dinámicas para recuperar y guardar varios valores en los ficheros, pues recuerda que cada acceso a ficheros ralentiza la aplicación.
15. Todas las lecturas que se realicen desde consola deben realizarse mediante un objeto **BufferedReader** (quedan totalmente prohibidos los objetos **Scanner**).
16. Los mensajes de aviso y/o error tienen que ser significativos y claros (cuidado con la redacción y la ortografía)
17. Crea una interfaz amigable, bien organizada, intuitiva, en definitiva, que sea fácil de seguir la lógica de la aplicación.
18. Añade otros elementos que mejoren la aplicación, que no se hayan contemplado en este documento y que creas adecuados para su buen funcionamiento.
19. Se deben entregar también los ficheros usados en las pruebas con información ya introducida. Deben estar en la misma carpeta del proyecto, claramente identificados y visibles.
20. Realiza una implementación de la aplicación siguiendo todas las reglas de una buena programación.

ÁNIMO, YOU CAN!!!