

ACTIVIDAD

Uso de estructuras de datos avanzadas, control de excepciones, serialización y creación de interfaces de usuario

Desarrollo de Aplicaciones
Web/Desarrollo de Aplicaciones
Multiplataforma
Programación



Actividad

Uso de estructuras de datos avanzadas, control de excepciones, serialización y creación de interfaces de usuario

Objetivos

- Escribir programas que manipulen información seleccionando y utilizando los tipos avanzados de datos facilitados por el lenguaje.
- Utilizar las clases básicas para almacenar y procesar información (listas y tablas de Hash).
- Implementar la gestión de excepciones en la utilización de clases facilitadas por el lenguaje.
- Implementar el lanzamiento de excepciones.
- Crear programas que accedan a la información serializada en ficheros.
- Desarrollar interfaces gráficas utilizando las librerías adecuadas.
- Programar controladores de eventos.

¿Cómo lo hago?

1. Rellena los datos que se piden en la tabla “Antes de empezar”.
2. Haz uso de fuentes comunes como Arial, Calibri, Times New Roman etc.
3. Utiliza el color negro para desarrollar tus respuestas y usa otros colores para destacar contenidos o palabras que creas necesario resaltar.
4. Entrega un zip que contenga todos los archivos. java que has creado. Para poder aprobar un ejercicio, éste debe poder ejecutarse sin errores.
5. Recuerda nombrar el archivo zip siguiendo estas indicaciones:
 - Ciclo_Módulo o crédito_Tema_ACT_númeroactividad_Nombre y apellido
 - Ejemplo: AF_M01_T01_ACT_01_Maria Garcia

Antes de empezar...

Nombre	GARA
Apellidos	GONZÁLEZ SOSA
Módulo/Crédito	M03 PROGRAMACIÓN
UF (solo ciclos LOE)	UF5
Título de la actividad	Uso de estructuras de datos avanzadas, control de excepciones, serialización y creación de interfaces de usuario

Se debe entregar un zip que contenga todos los archivos .java que has creado. Para poder aprobar un ejercicio, éste debe poder ejecutarse sin errores. Crea los archivos .java dentro de una carpeta de nombre actividad06

1. Crea la clase Empleado con la siguiente información: un nombre, un apellido, un sueldo y una clave que tendrá por defecto el valor “patata”. Crea una clase con los atributos necesarios para almacenar los datos de cada empleado y los métodos constructores, getters y setters. La clase Empleado debe poder serializarse, excepto el atributo que guarda la clave.
2. Crea la clase VentanaPrincipal con el método main. Al ejecutarse deberá
 - a. crear varios empleados ,almacenarlos en una HashMap según su nombre y guardar este HashMap en un archivo
 - b. Iniciar una ventana definida en la clase VentanaBienvenida.
3. Crea la clase VentanaBienvenida para que:
 - a. muestre una ventana con los siguientes elementos:
 - i. Una caja de texto para introducir el nombre.
 - ii. Una caja tipo JPasswordField para introducir el password (que será el apellido)
 - iii. Un botón Login.

Yo esta ventana la llamé VentanaLogin, con el JFrame cree los siguientes botones :

- 1) Usuario y contraseña: El usuario debe introducir y estas deben coincidir. Debe existir el usuario por su nombre y la contraseña debe ser igual al apellido del atributo.
Traemos los valores introducidos por el usuario con la con la función getText()
- b. Crea una clase ErrorLeerArchivo que herede de Exception. Si no se puede acceder al archivo, lanza éste error
- c. Al abrir la ventana llama a una función “leerArchivo()” que intente acceder al archivo con la información de los empleados:
 - i. Si puede acceder, muestre por consola todos los datos almacenados en el archivo en el que se almacenó la HashMap y se guarden en una variable de la clase VentanaBienvenida.
 - ii. Si no puede acceder, lance una excepción del tipo “ErrorLeerArchivo” que va a tener que controlar la clase VentanaBienvenida mostrando un mensaje al usuario.
 - iii. El nombre del archivo debe de ser una variable privada de la clase VentanaBienvenida.

- d. programa el botón Login de tal forma que cuando se haga clic en él, compruebe que el usuario existe en el archivo con el nombre y mismo apellido (introducido en el password) indicado por el usuario.
- Si todo es correcto, se abrirá una nueva ventana dando la bienvenida al usuario con su apellido y un botón para cerrar la aplicación
 - Si el usuario no rellena ningún campo se mostrará el texto oculto "DEBES INTRODUCIR LOS DATOS" en color rojo
 - Si el nombre o password (el apellido) no concuerda se mostrará el texto "CREDENCIALES INCORRECTAS" en color rojo

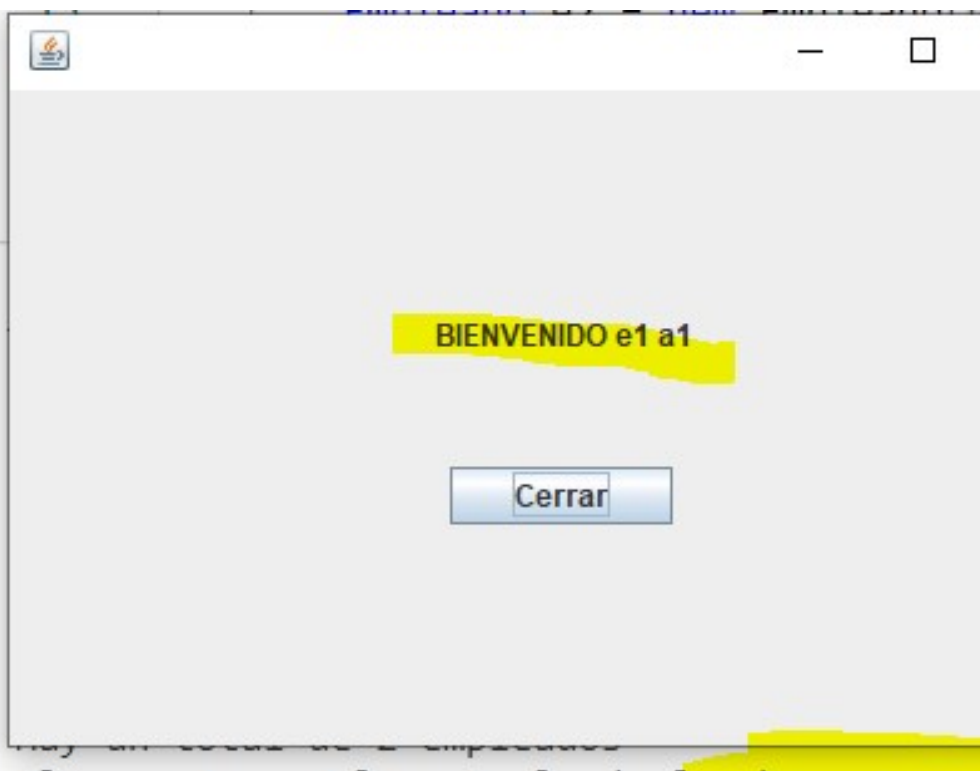


LOGIN

Nombre usua... e1

Contraseña (a... ..

Aceptar



Primero creamos la Clase Empleado que implementa de la interfaz Serializable. Vamos a guardar los objetos Empleado en un archivo.



Con el modificador Transient hacemos que no se guarde en el fichero el atributo clave.

Creamos los constructores para inicializar los atributos, los getters y setters. También con el toString para luego en la Ventana Login muestre el nombre y el apellido del empleado que introdujo su usuario y contraseña.

Creamos la clase VentanaPrincipal con el main y almacenamos los empleados en un HashMap. Utilizamos la función .put para añadir cada empleado en el listado. Nuestro HashMap es de tipo String (porque la clave va a ser el nombre de cada empleado) y de tipo Empleado que es el tipo de objeto que va a guardar.

En este punto vamos a establecer ya el nombre de un archivo y lo vamos a guardar en una variable que va a ser fija.

Para guardar nuestro HashMap en un archivo, hacemos un método escribirFichero y le pasamos nuestro HashMap definido en el main por parámetros, y el nombre del archivo guardado en una variable que se va a llamar "listado trabajadores".



Con el Iterador recorreremos nuestro hashMap y mostramos por consola la clave (nombre) y el valor (nombre y apellido). El sueldo no porque el toString lo modifiqué para cuando abra la ventanaBienvenida solo muestre el nombre y apellido del empleado.

Desde el main llamamos a VentanaLogin que es la que le va a aparecer al empleado (con el método set.Visible) para rellenar su usuario y contraseña.

Recorremos una listaEmpleados con el containsKey y nos aseguramos que el nombre que escriba el usuario esté en nuestra lista de empleados y la guardamos en una variable testEmpleado.

Una vez se comprueba que exista el usuario, forzamos a que la contraseña deba ser el apellido de este usuario, igualando con un equals. Si esto se cumple nuestra ventanaLogin se hará invisible y se abrirá otra que será nuestra VentanaBienvenida. Como ya tenemos los datos del empleado guardado en la variable testEmpleado, se lo pasamos por parámetro a VentanaBienvenida para que cuando abra su ventana, de la bienvenida + especifica con el nombre y apellido del usuario, motivo por el cual modificamos nuestro toString.

Trataremos los posibles errores que pueda poner el usuario con una excepción personalizada que la llamaremos ErrorLeerArchivo:

- 1) Si el usuario no introduce usuario o contraseña, lanzaremos la excepción, que llegará a la clase padre de Excepcion pasando por parámetros el mensaje de error que le corresponda.
- 2) Si el usuario añade el nombre del empleado y no existe.
- 3) Si el usuario añade una contraseña que no sea el apellido del empleado.

Haremos un try/catch para recoger todas estas posibles excepciones. Y aprovechamos para codificar en la ventanaBienvenida una nueva Label que, cuando el programa entre en el catch se vuelva visible la label y muestre mensaje de error, tanto en la pantalla como en la ventana.