PRÁCTICA 4 de JAVA:

Serialización, excepciones, interfaz y ordenación

Fecha tope entrega	Viernes, 3 de noviembre.	**Fecha tope defensa**	Viernes, 3 de noviembre.
Tipo	Parejas	**Asunto email**	JAVA04
Formato fichero	Colaborador Git (el README indicará los autores)		

Requisitos:

1.- Modelo. Partiendo de la clase **Empleado** desarrolla dos subclases que hereden de ella. Características:

1.1.- Empleado tendrá los siguientes atributos: número (entero), nombre (alfanumérico, además sera **transient** o no serializable por seguridad), sueldo y sueldo máximo (real para cada empleado) y fecha de alta (fecha).

1.2.- La subclase **Analista** tendrán los atributos:

- Plus anual (real): porcentaje extra sobre el sueldo mensual que se aplica anual.
- Otro que debes personalizar.

1.3.- La subclase **Programador** tendrán los siguientes atributos:

- Sueldo extra mensual (real).
- Otro a personalizar, pero distinto en nombre y tipo al de Analista

2.- Modelo. Desarrolla un **interfaz e implementarse sobre las clases hijas** en los cálculo con fechas. Deberá:

2.1.- Tener los métodos para controlar si se cumplen meses o años. Dichos métodos no tendrán parámetros y compararán la fecha de apertura de la Empleado y la fecha del sistema.

2.2.- Tendrá las constantes de la clase Calendar (DAY_OF_MONTH, MONTH y YEAR) en **español**.

3.- Modelo. Desarrollo un **sistema de excepciones** para implementar dentro de la clase Empleado (en el constructor, setSueldo y setSueldoMaximo) y evitar que el sueldo de cada empleado nunca supere el sueldo máximo. Su control debe ser jerarquizado (try, catch, …, catch y finally). Los mensajes de las excepciones solo serán visibles para el usuario de la aplicación, en la la vista del MVC.

4.- Controlador. Debes utilizar las clases **Lista** y **Nodo** de la práctica anterior con modificaciones:

- Los nodos tendrán un **índice** (int) para guardar el número de empleado.
- Tendrá nuevos métodos como:
- boolean **existe** (int numEmp): nos indica si existe o no un empleado por su número.
- void **interCambiar** (int numeEmp1, int numEmp2). Cambia el valor del indice y del atributo main (objeto empleado), de dos empleados en la lista sin tocar los punteros. Se usara para ordenar la lista.
- boolean **sort** (). Ordena la lista por número de empleado con el método de la burbuja u otro.
- Cualquiera que estimes oportuno.

5.- Controlador. Los empleados se podrán cargar desde un fichero, si existe, a una lista (puede estar vacía). Y el proceso inverso, se podrán guardar desde la lista a fichero. Debe usarse el interfaz **Serializable** (los atributos transient se sustituirán por asteriscos).

6.- Vista. Implementa una aplicación gráfica con las siguientes opciones:

6.1.- **Cargar** datos del fichero a lista.

6.2.- **Guardar** datos de lista al fichero.

6.3.- Crear e **Insertar** un nuevo empleado (**analista o programador)**. El número de empleado será inferior a entre 1.000 y 2.000. Habrá unos 10 empleados creados de este modo.

6.4.- Creado masivo y aleatorio de empleados. Primero se crearan unos 100.000 empleados con numero de empleado aleatorio entre 2.001 y 1.000.000. Después 20 empleados con número de empleado entre 1 y 2000. Lo que importa es el número de empleado, tampoco si en Analista o Programador, el resto de atributos pueden ser iguales, nulos… Su utilidad será a la hora de ordenar la lista.

6.4.- Visualizar en **JList** todos los objetos indicando el tipo de cada uno (Analistas o Programador).

6.5.- **Visualizar uno a uno** los empleados. Tendrá los botones: cargar, guardar, anterior, siguiente, primero, último, nuevo o insertar, creado masivo, ordenar y **calcular** (el nuevo sueldo con el plus o el sueldo extra según el tipo de empleado).

6.6.- El botón **calcular solo estará activo sí se cumple los periodos**: cumple meses (igual día de la fecha de alta y de la del sistema) para Programadores, o cumple años (igual día y mes de las fechas) para Analistas. Modificará el sueldo si no supera al sueldo máximo e informará de lo ocurrido con detalle.

6.5.- Para la defensa habrá empleados con datos para que cumplan las condiciones de tiempo para realizar cálculos sobre el sueldo (mes o año) y para que salte la excepción sobre el sueldo máximo.

7.- El botón ordenar realizará los siguientes pasos:

7.1.- Crear los empleados con ya se ha mencionado.

7.2.- Insertara los empleados en la lista y en una **collections** a elegir.

7.3.- Finalmente mide el tiempo (en mili-segundos) utilizado en ordenar la lista por un lado y la collection por otro, por el atributo número de empleado. Se mostrarán los tiempos de ordenación en un JDialog y los 100 primeros empleados por consola.

7.4.- Se puede modificar el número de empleado creados aleatoriamente para que los tiempos sean significativos y no salga cero en la collection.

8.- Controlador. Lista y gestión de ficheros.

9.- Estará estructura siguiendo el MVC con paquetes.

10.- Mejoras: Date Picker, Filechooser, refresco actualizaciones, control de errores, Jlist seleccionable, LookAndFeel, manejo de fechas, etc.

Ayuda:

Serializable:

http://chuwiki.chuidiang.org/index.php?title=Serializaci%C3%B3n_de_objetos_en_java

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html

Generics:

https://docs.oracle.com/javase/tutorial/extra/generics/index.html

LookAndFeel

http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

Ejemplos prácticos:

http://es.wikihow.com/serializar-un-objeto-en-Java

http://www.chuidiang.com/java/ficheros/ObjetosFichero.php