PRÁCTICA 4 de JAVA:

Tipos genéricos, serialización, excepciones, interfaz y ordenación

Fecha tope entrega	Lunes, 10 de octubre.	**Fecha tope defensa**	Lunes, 17 de octubre.
Tipo	pareja	**Entrega**	Colaborador en Github

Requisitos:

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

1.1.- Las cuentas tendrán los siguientes atributos: número (entero), titular (alfanumérico, además sera **transient** o no serializable por seguridad), saldo y saldo mínimo (real) y fecha de apertura (fecha).

1.2.- El número de cuenta será aleatorio entre 1 y 100.

1.3.- La subclase **Cuenta de Ahorro** tendrán los atributos:

- Interés anual (real, se expresará en **tanto por ciento**)
- Otro libre a desarrollar por ti.

1.3.- La subclase **Cuenta Inversión** tendrán los atributos:

- Beneficio o perdida bursátil: cantidad monetaria que se gana o se pierde por inversiones en bolsa, se calcula de forma aleatoria entre el 10% del saldo de la cuenta y puede ser positivo o negativo, lo hará de manera trimestral.
- Otro libre a desarrollar por ti.

1.4.- La subclase **Cuenta Corriente** tendrán los siguientes atributos:

- Comisión de mantenimiento (real).
- Tipo de comisión (puede ser semestral o anual).

2.- Desarrolla un **interfaz e impleméntalo sobre las clase hijas** para los cálculo con fechas. Deberá:

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

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

3.- Los objetos se cargarán desde un **fichero**, si existe, a una lista (puede estar vacía). Y el proceso inverso: se podrán salvar o guardar desde la lista a fichero. Debe usarse el intefaz **Serializable**.

4.- Desarrollo un **sistema de excepciones** para controlar que el saldo de una cuenta NUNCA sea inferior al saldo mínimo. Su control debe ser jerarquizado (try, catch, …, catch y finally). Los mensajes de las excepciones solo serán visibles en la vista (MVC).

5.- Modifica las clases **Lista** y **Nodo** de a la práctica anterior:

5.1.- Tendrá un **índice** mediante un vector con un tamaño dado al crear el objeto.

5.2.- En cada elemento del vector (Nodo) almacenará un entero para el número de cuenta y la cuenta. Luego, la clase Nodo tendrá como atributos un entero y una cuenta, y los métodos necesarios.

5.3.- Seguirá definiéndose como genérica, es decir, para cualquier objeto.

5.4.- Desarrolla los métodos que creas necesarios, entre ellos intercambiar(x,y) que modificará las posiciones de esos nodos.

6.- 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.- **Insertar** o nuevo elemento (**cuenta ahorro o corriente)**. Para la defensa habrá objetos con datos para que cumplan las condiciones de tiempo para realizar cálculos sobre el saldo (mes, trimestre o año) y salte la excepción sobre el saldo mínimo.

6.4.- Visualizar en **JList** eĺ número, titular y saldo de todos los objetos indicando el tipo de cada uno (cuenta ahorro, inversión o corriente).

6.4.1.- Al seleccionar sobre cualquier elemento del Jlist veremos el resto de sus atributos.

6.5.- **Visualizar uno a uno** los elementos, con la posibilidad de operar solo sobre el visualizado. Tendrá los botones anterior, siguiente y **calcular** (no se hará de forma automática).

6.5.1.- El botón calcular solamente estará activo sí se cumple el periodo para hacer el cálculo, es decir, si se cumple un mes para calcular el interés de las cuentas de ahorro y un trimestres para las cuentas de inversión y un semestre o un año para las cuentas corrientes.

6.5.2.- Al pulsarlo se modificará el saldo si procede y se informará de lo ocurrido con detalle (incremento o decremento y cantidades).

7.- La aplicación también permitirá **ordenar** la lista con los siguientes pasos:

7.1.- Primero completará la lista con 100.000 cuentas aleatorias con números mayores a 1000.

7.2.- Seguidamente, implementa una **collections** y copiará la lista original en ella.

7.3.- Finalmente ordenarán las dos estructuras de datos por el campo número, se medirá y se mostrará el tiempo empleados en ordenar cada una.

7.4.- Tras ordenar se podrán hacer un listado por consola, o mejor en el Jlist.

8.- Estará estructura siguiendo el MVC, también los paquetes.

9.- Mejoras: Date Picker, Filechooser, refresco actualizaciones, control de errores, 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/java/generics/types.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