Recuperación UD1. Acceso a datos. 2º DAM. 23/01/2025

Antes de empezar:

- Verifica que Eclipse o VisualStudio, funcionan. Eclipse está instalado y también tienes en tu carpeta un Eclipse portable.
- Verifica que tienes los archivos java y jars necesarios para el examen
- Verifica que en tu home (/home/examenADAT/) tienes los zip con los PDFs del tema. En caso contrario descárgalos del aula virtual antes de que se corte internet para el examen.
- Lee dos veces el enunciado de todos los ejercicios y pregunta al profesor cualquier duda que te surja sobre los mismos.

Normas:

- Está prohibido el uso de cualquier dispositivo electrónico (móvil, auriculares, reloj inteligente …) y el uso de internet salvo las indicaciones expresas de acceso al aula virtual que de el profesor para el acceso y entrega del examen.

Entrega:

- El código fuente realizado. Puedes hacer todos los ejercicios en una sola clase pero separando cada apartado en un método distinto. Para ello te proporciono la clase Recu_UD1 como esqueleto a completar. Puedes no usarlo y/o crear todos los métodos y/o clases que necesites.

Notas:

- Deberás completar la clase Recu_UD1.java y la clase City.java según necesites para lograr el cometido.
- Tienes ficheros de ejemplo de cual debe ser el resultado de cada ejercicio.

Ejercicio 1.- HSV.

1.1.- (0,75 puntos) A partir de la clase City.java y el fichero cities.hsv implementa un método que lea los contenidos del fichero y genere una lista de ciudades a partir de los datos del fichero (100 líneas, puedes quitar la primera que es el encabezado)

El fichero HSV es como un CSV pero con ### en vez de comas. Algunos registros no tienen todos los campos. puedes ignorarlos.

1.2.- (0,75 puntos) Implementa otro método que a partir de la lista de ciudades escriba un fichero CSV con nombre cities.csv con los valores extraidos de la lista: una línea por cada objeto City y los campos separados por comas simples.

1.3. - (0,5 puntos) Guarda el objeto config de tipo Properties en un fichero en formato xml de Properties en RUTA_CONFIG. Tras ejecutarlo una vez comenta todas las líneas entre la 20 y la 28 para que luego leePropertiesXml lea ese fichero con las rutas. En ambos casos captura las excepciones.

Ejercicio 2. XML.

2.1.- (1 punto) A partir de la lista de ciudades del ejercicio anterior genera un archivo xml que refleje la estructura de lista y los atributos de cada elemento en mayúsculas. Puedes usar cualquier librería de los jars proporcionados pero no Properties (ejercicio anterior) pues no queremos una estructura de clave-valor.

2.2.- (1 punto) modifica el punto anterior para que el xml generado tenga el countryCode como atributo de ciudad, todos los tags vayan en mayúscula y el fichero empiece por CITIES, tipo:

```
<CITIES>
  <CITY COUNTRYCODE="NHA">
    ...
  </CITY>
...
</CITIES>
```

2.3.- (1 punto) implementa un método que lea el fichero generado por el

apartado anterior y reconstruya la lista de ciudades. Si no te han salido los apartados anteriores puedes generar el fichero a mano.

Notas: xstream.addPermission(AnyTypePermission.ANY);

Ejercicio 3.- JSON.

3.1.- (1 punto) implementa un método que guarde la lista cities en formato json cambiando con las etiquetas json a mayúsculas.

3.2.-(1,5 puntos) implementa un método que reconstruya una lista de ciudades a partir del json generado en el apartado anterior.

Ejercicio 4. Ficheros binario.

4.1.- (1 punto) Implementa el método guardaBinario que guarde los valores del array de enteros nums en un archivo binario. Cada valor será un int binario (4 bytes)

4.2.- (1,5 puntos) Implementa el método guardaUnbinario que sobreescriba en el archivo anterior solo la posición marcada (posición) por un nuevo valor (valor) sin recorrer ni tocar el resto de valores.