PARTE 3: Ejercicio de resolución de un problema mediante un programa en JAVA

Duración 90 minutos

Este ejercicio comprende **2** actividades principales:

1. Comprensión del problema con su conjunto de datos asociado, análisis de métodos de ordenación que puedan aplicarse para resolver el problema, elección de uno de ellos y justificación.
2. Resolución del problema:
   a) Desarrollo del programa JAVA necesario para resolver el problema (de ordenación) planteado.
   b) Desarrollo de los casos de prueba ("test case", al menos uno) para verificar la corrección de la funcionalidad implementada.

ESCENARIO

Una empresa de comercio electrónico de India ofrece en línea una gran cantidad de productos.

Se cuenta con un conjunto de datos típico, que se obtiene habitualmente tomando de los diferentes proveedores de los productos físicos los datos correspondientes ("crawling"), en tiempo real.

La empresa necesita publicar también **los productos** en forma **descendente por código único** de producto.

Se provee una muestra de un conjunto de datos descargado y ensamblado en un cierto momento, "***flipkart_com-ecommerce_sample.csv***".

Cada producto se encuentra representado en una línea. El primer campo es "***uniq_id***", el identificador único de cada producto en la base de datos

Observa los atributos / datos asociados a cada producto. Uno de ellos es "***product_name***".
Descarga de la webasignatura el archivo "**Parcial3-2018.zip**" que contiene el conjunto de datos indicado y la clase java "**Producto**".

SE REQUIERE:

Análisis de los datos, alternativas de algoritmos de ordenación, elección y justificación.

1. Realiza una inspección de los datos para observar los campos incluidos en cada línea.
2. Analiza posibles algoritmos de ordenación apropiados para, a partir del conjunto de datos de entrada indicado, emitir un listado de los productos, ordenados en forma descendente por código único de producto.
3. Elige el algoritmo que te parezca más apropiado, tomando en cuenta criterios de tiempo de ejecución, consumo de memoria y complejidad del algoritmo. Justifica esta elección.
4. Reporta todas estas consideraciones en un **breve** (no más de 1 carilla) documento "***analisis.doc***" (a entregar junto con todo el código que desarrolles y el archivo de salida)

Funcionalidad a desarrollar:

1. Genera una estructura apropiada para representar el problema y cargarla a partir del archivo de entrada indicado. Para facilitar, se provee una clase "**Producto**" cuyo constructor toma como parámetro una línea completa del archivo de entrada y crea una instancia de esta clase, utilizando el campo "***uniq_id"*** como "clave" primaria (la clave por la cual deberá hacerse la ordenación). Observa que esta clave es alfanumérica.
2. Realiza la ordenación del conjunto de datos **con la implementación que decidiste en la primera parte del trabajo (análisis), en orden descendente por la clave (uniq_id). SOLO DEBES IMPLEMENTAR EL ALGORITMO QUE HAS SELECCIONADO Y DOCUMENTADO PREVIAMENTE (NO SE ACEPTARÁN OTROS, pero sí puedes documentar e implementar MAS de uno si lo deseas – será tenido en cuenta-).**

3. Emite un archivo de salida "**productos_ordenados.txt**", que contenga en cada línea el código de producto y el nombre del producto, separados por un espacio simple.

Test cases.

Implementa los **Casos de Prueba** necesarios para verificar el correcto funcionamiento de los métodos desarrollados.

ENTREGA

Subir a la webasignatura, en la tarea "**PARCIAL3-2018**" :

- El documento "analisis.doc"
- El proyecto completo realizado,
- El archivo de salida "**productos_ordenados.txt**".

RUBRICA DE CALIFICACIÓN:

Se utilizarán los siguientes criterios en la evaluación del trabajo remitido:

1. ANÁLISIS: 30%

- Identificación del problema, revisión de los datos.
- Evaluación de algoritmos de ordenación
- Selección de mejor opción y justificación de la misma

2. DESARROLLO y EJECUCIÓN 40%

- Estructura de datos utilizada (pertinencia Y eficiencia)
- Implementación del método de ordenación con el algoritmo **SELECCIONADO EN LA PARTE DE ANÁLISIS**
- Correcta lectura del archivo de datos y creación de las estructuras definidas.
- Programa principal: ejecución en tiempo razonable, aplicación correcta del método de ordenación y generación correcta del archivo de salida.

3. CALIDAD DEL CÓDIGO (10 %)

- Selección inteligente de estructuras auxiliares en aras de reducir complejidad, tiempo de ejecución, uso de recursos en general.
- Nombres de variables y métodos
- Aplicación correcta y rigurosa del paradigma de programación orientada a objetos
- Invocación racional y eficiente de métodos y funciones
- Encapsulación, modularidad
- Utilización apropiada de clases de colecciones y genéricos necesarios

4. PRUEBAS DE UNIDAD 20%.

- Calidad de los tests desarrollados, todas las condiciones normales y de borde, se testean todos los métodos, uso del enfoque inductivo en los tests.