Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Programación Avanzada
Semestre 2023-1
Proyecto Final

1. Introducción

Las aplicaciones basadas en programación necesitan comunicarse con su entorno, tanto para obtener datos e información que deben procesar, como para devolver los resultados obtenidos. En JAVA, el manejo de archivos se realiza a través de streams o flujos de datos desde una fuente hacia un repositorio. La fuente inicia el flujo de datos, por lo tanto, se conoce como flujo de datos de entrada. El repositorio termina el flujo de datos y se conoce como flujo de datos de salida. Es decir, tanto la fuente como el repositorio son nodos de flujos de datos.

En ocasiones, cuando se manejan archivos de texto plano como parte de una aplicación, estos pueden contener grandes volúmenes de datos (archivos formados exclusivamente por texto sin formato que no requieren ser interpretados pero si pueden ser procesados). El procesamiento de los mismos, por ejemplo, implica leer un registro, aplicar un poco de lógica de negocios y actualizar una base de datos con el resultado obtenido. Pero cuando se requiere que esto se realice en poco tiempo, no es funcional resolver el problema como un proceso único. A medida que los sistemas se escalan, recibimos más carga, es decir, archivos más grandes con millones de registros ocasionalmente, como en los datos del clima, de covid, de enfermedades, bancos, etc. En este escenario es ideal tener tiempos de espera cortos.

Así, una forma de resolverlo es aplicar concurrencia. De tal manera que, se divida el archivo de registros correspondiente en múltiples subarchivos y procesar cada archivo en paralelo. Por lo que es necesario utilizar para su análisis y procesamiento múltiples hilos.

2. Objetivo

Que el alumno ponga de manifiesto los conocimientos adquiridos en la materia de Programación Avanzada en Java, mediante la propuesta, desarrollo y solución de un proyecto donde se aplique un caso hipotético o real.

3. Funcionalidad

1. Definir un problema tal que, como parte de la solución e implementación del mismo, el programa lea archivos en formato de texto plano CSV.

2. Para el proceso y análisis de los registros, el archivo origen (dataset) debe ser dividido en múltiples subarchivos (por lo menos 4 veces el número de CPU's que detecte la JVM en cada arquitectura, es decir, que el código debe ser dinámico y el usuario no tenga que estar estableciendo ese valor en cada ejecución en distintas arquitecturas) y procesar cada archivo en multihilo.

3. La cantidad de registros contenidos en el archivo origen debe ser por lo menos de 5 millones y el número de columnas de por lo menos 20.

4. Se debe de crear la estructura lógica suficiente para realizar concurrentemente lo siguiente:

   a) **Lectura.** Sobre cada archivo generado, producto de la partición del dataset original, cada hilo debe realizar las funcionalidades que se solicitan, es decir, vamos a crear un pool de hilos que cada uno atienda a un archivo en forma concurrente.

   b) **Escritura de resultados.** Este será el recurso compartido, donde todos los hilos deben ingresar a escribir sus resultados en forma sincronizada, dicho archivo debe cumplir con lo siguiente: Se deberá crear en un subdirectorio local (en la misma ruta del archivo origen) con el nombre *resultados*, y dentro estará el archivo destino, el cual deberá nombrarse con la siguiente nomenclatura:

   nombreOrigen_filtered(AAAAMMDD_HHMM).csv

   Donde AAAAMMDD_HHMM es un estándar de especificación de fecha propio de esta materia y se compone de lo siguiente:

   - AAAA. 4 dígitos del año
   - MM. 2 dígitos del mes
   - DD. 2 dígitos del día.
   - HH. 2 dígitos de la hora en formato de 24 hrs.
   - MM. 2 dígitos de los minutos

   Por ejemplo: datasetDengue_filtered(20211101_1720).csv

c) Si el directorio *resultados* no existe, se debería de crear en automático, pero si existe, no se deben borrar los archivos ya contenidos en él.

5. Al final se debe de mostrar un mensaje donde indique que la tarea finalizó con éxito y el tiempo que se tardó en hacer el procesamiento.

4. Ejemplo de uso y Salida

El archivo de salida debe ser un subconjunto del archivo origen, por lo tanto el usuario debe poder ingresar las columnas de su interés que lo conformarán, así como también un criterio para filtrar los registros de su interés.

Una de las principales características de java es que es *multiplataforma*, así que el programa no debe estar "amarrado" a un único sistema (windows o linux por ejemplo).

A continuación se muestran ejemplos de posibles entradas de datos para el uso y filtrado de información en un caso específico, el alumno es libre de elegir la forma de obtener estos datos que el usuario debe ingresar.

```
≫ Ingresa la ruta del archivo csv:
      C:/archivoFuente/datosDengue.csv
≫ Ingresa las columnas de interés:
      nombre, apePat, escolaridad, nacionalidad, fecha_nac
≫ Ingresa el criterio de filtrado:
      edad = 18
```

```
≫ Ingresa la ruta del archivo csv:
      D:\pruebas\misArchivos\datosOrigen.csv
≫ Ingresa las columnas de interés:
      *
≫ Ingresa el criterio de filtrado:
      nacionalidad = mexicano
```

Nota: En este ejemplo, el carácter asterisco ($*$) funciona como comodín que indica *"todo"*.

5. Presentación de los datos

La presentación de resultados es una de las partes mas importantes de toda investigación o análisis de datos, no es nada recomendable presentarlos únicamente como un conjunto de datos en texto plano, es importante contar con el sustento pero estos deben ser representados de manera visual. Por lo tanto, es necesario que, una vez obtenido el archivo filtrado, los resultados tengan un fin (estádistico, analisis de datos, etc) y sean representados de alguna manera visual para aquellos que lean su trabajo o esten presentes en la exposición. Esto puede ser en gráficas de cualquier tipo, mapas geográficos, mapas de calor, etc. Cualquier elemento que nos indique visualmente los resultados obtenidos después del filtrado de la información, así como su interpretación.

Sólo esta última parte de la presentación de resultados se puede hacer con el software de su preferencia. Las figuras 1, 2 y 3 son ejemplos de la representación de los datos filtrados.

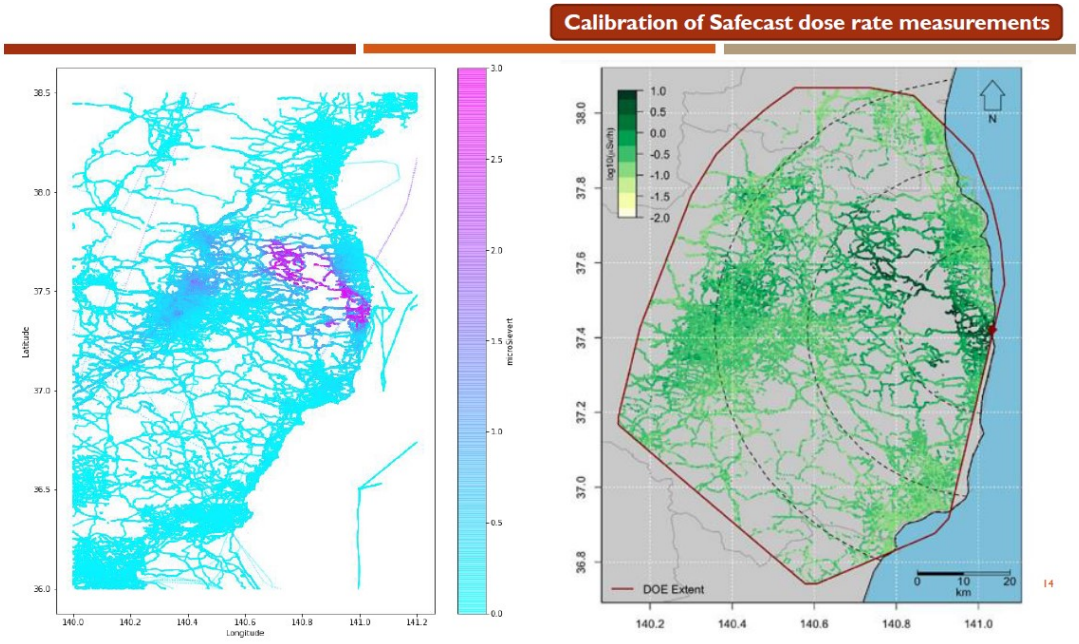
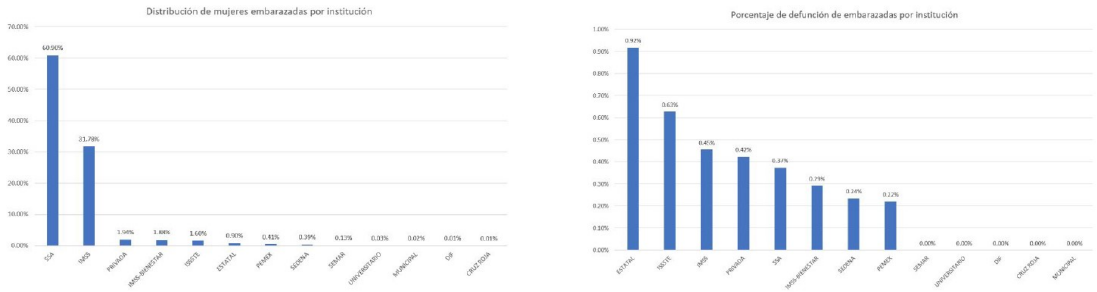


Figura 1: Ejemplo de representación visual de datos



La SSA y el IMSS atiende a más del 90% de las mujeres embarazadas, mientras que el porcentaje de defunción de mujeres embarazadas atendidas por institución es menor al 1% en todas las instituciones.

Figura 2: Ejemplo de representación visual de datos

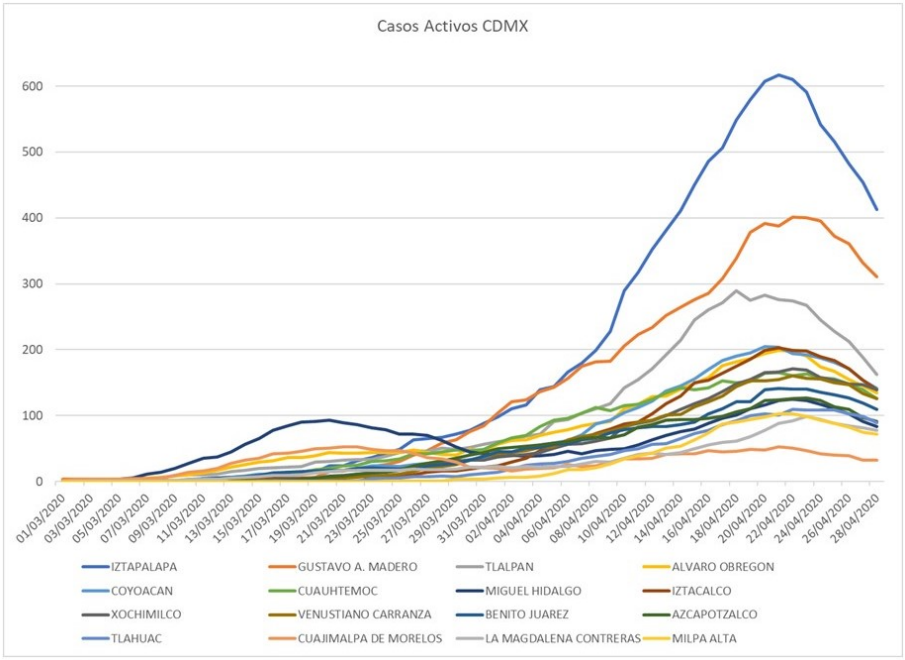


Figura 3: Ejemplo de representación visual de datos

6. Requisitos de implementación

El proyecto debe hacer uso de los conceptos vistos en clase y cumplir con los siguientes requisitos:

- Empleo correcto de los pilares de la POO:
  - Abstracción
  - Encapsulamiento
  - Herencia
  - Polimorfismo
- Manejo de Excepciones.
- Manejo de archivos de texto plano, mediante I/O.
- Manejo de hilos implementando el patrón *Manager-Workers*
- Se debe de poder hacer todo lo solicitado sin tener que modificar nada en código. Esta será una aplicación para usuario final.
- **No se permite el uso de ninguna biblioteca o framework externo a las herramientas incluidas en el API del JDK 8. Esto incluye cualquier biblioteca para manejo de archivos o hilos, se debe realizar con lo visto en clase.**

7. Entregables

Entregar tres archivos, los cuales deben de cumplir con el siguiente nombrado y contenido. Ejemplo:

```
Equipo01_codigo
Equipo01_presentacion
Equipo01_reporte
```

- **Código fuente del proyecto.** Debidamente organizado y en archivo rar o zip. Colocar dentro del zip un archivo TXT con la liga a la base de datos (archivo CSV) que usaron para su proyecto, no suban la base completa pues debe ser muy pesada por la gran cantidad de registros que debe tener.

- **Presentación en formato PDF.** Pueden usar cualquier recurso para su realización. Asignaremos tiempo en horario de clase para que realicen sus presentaciones.
- **Reporte técnico en formato PDF.** Formato libre. Debe de ser un escrito formal (cuidar presentación, estilos, formatos, alineación de texto, figuras, etc.)