UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Estructuras de Datos

Práctica 1: Algoritmos inocentes.

Profesora:
Amparo López Gaona
Ayudante: Adrián Aguilera Moreno
Ayudante de Laboratorio: Kevin Jair Torres Valencia

Objetivos

Repasar sus conocimientos en programación con el `Java`, aplicar los conocimientos adquiridos sobre la serialización de clases e iniciar el análisis y la implementación de algoritmos básicos.

Introducción

La serialización de objetos permite escribir objetos a archivos con una sola instrucción, con lo cual quedan grabados hasta que se decida eliminarlos o modificarlos. También permite recuperar los objetos grabados en archivos.

Para que objetos de una clase puedan serializarse es necesario que dicha clase implemente la interfaz Serializable, que se encuentra definida en el paquete java.io. La interfaz es una interfaz de marcado que tiene el siguiente código:

```
public interface Serializable { }
```

Para que un objeto sea serializable todas las variables de la estructura deben ser serializables. Todos los tipos primitivos, los objetos de la clase `String` y algunos de otras clases de `Java` son serializables.

Para serializar objetos, además de especificar que serán serializables se requiere trabajar con un archivo, en el cual se almacenen o bien se recuperen de ahí cuando se requiera. Al trabajar con archivos es necesario considerar las posibles excepciones que pueden dispararse, éstas son subclases de `IOException` definida en el paquete `java.io`. En este paquete está también definida una serie de subclases de esta excepción, entre ellas `EOFException` y `FileNotFoundException`.

Para grabar objetos en un archivo, es decir, para serializar, se requiere crear un objeto de la clase `ObjectOutputStream` del paquete `java.io` utilizando la siguiente instrucción:

```
ObjectOutputStream obj = new ObjectOutputStream(new
    FileOutputStream(nombreArch));
```

Una vez creado el objeto de la clase `ObjectOutputStream`, dentro de la clase de interés, se puede utilizar el método `writeObject(objeto)` para grabar el objeto que toma como parámetro. Si el objeto que se intenta grabar no es de una clase que implemente la interfaz `Serializable` se dispara la excepción `NotSerializableException`.

Al terminar de serializar todos los objetos se debe llamar al método close para asegurar que no se pierdan los objetos grabados en el archivo especificado. Independientemente de que haya habido un error o no es necesario cerrar el archivo, por esto es recomendable incluirla en la clausula `finally`.

La operación complementaria a grabar objetos es la de recuperarlos. Para recuperar objetos de un archivo se requiere crear un objeto de la clase `ObjectInputStream` como sigue:

ObjectInputStream objeto = new ObjectInputStream(new FileInputStream(nombreArch));

Para leer los objetos serializados se utiliza el método `readObject()`, el cual construye un objeto de la clase indicada pero lo regresa como una referencia de tipo `Object`, por lo que es necesario hacer una conversión explícita.

Desarrollo

Dada una cadena A de n carácteres alfanuméricos, queremos contestar la pregunta ¿Hay algún elemento de A que aparezca al menos $\frac{n}{3}$ veces? Diseña un programa que lea la cadena A de un archivo de texto plano y responda la pregunta para cada uno de los archivos.

Contestar la pregunta anterior y además responder ¿Cuánto le toma a k de tiempo encontrarlo su programa, deben medir este tiempo e indicarlo. Para esto pueden usar el método `System.currentTimeMillis()` que devuelve un tipo de dato primitivo `long` que representa la hora actual en milisegundos.

El programa tendrá la siguiente estructura:

```
1  import java.io.BufferedReader;
2  import java.io.FileReader;
3  import java.io.IOException;
4
5  public class charFrequency {
6      public static void main(String[] args) {
7          String archivo = "archivo.txt";
8          String cadena = "";
9
10         try (BufferedReader br = new BufferedReader(new FileReader(archivo)
       )) {
11             cadena = br.readLine();
12         } catch (IOException e) {
13             System.out.println("Error al leer el archivo: " + e.getMessage
       ());
14             return;
15         }
16     }
17 }
```

Listing 1: Metodo que ayuda a leer un archivo txt

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. Cada práctica (sus archivos y directorios) deberá estar contenida en un directorio llamado `EquipoX_pY`, donde:
   (a) X es el número de equipo correspondiente.
   (b) Y es el número de la práctica.

   Por ejemplo: `Equipo09_p01`
3. NO incluir los archivos .class dentro de la carpeta.
4. Los archivos de código fuente deben estar documentados.
5. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
6. La práctica se debe subir al Github Classroom correspondiente.
7. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
8. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.