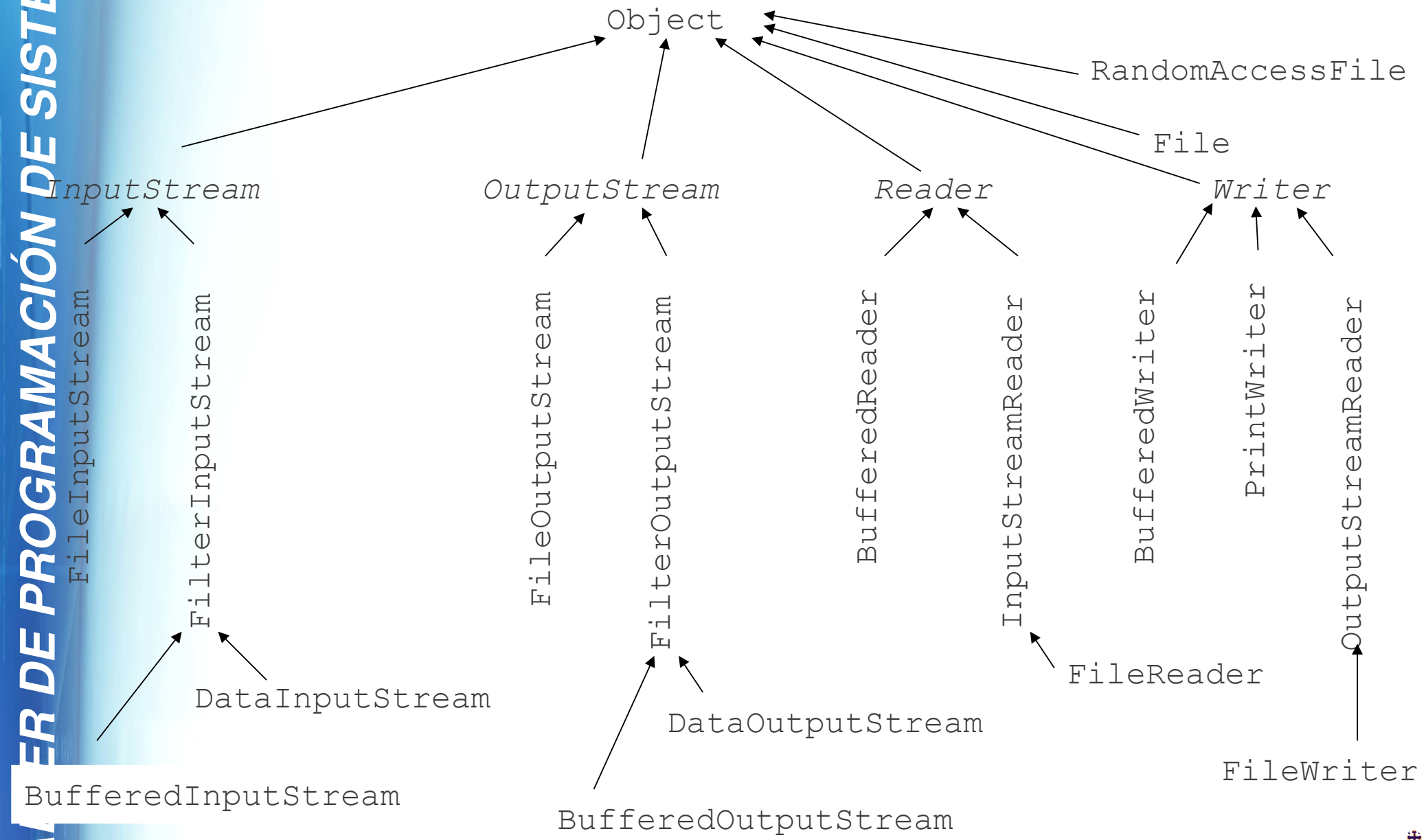
Manejo de archivos

Esencial para TODO el semestre

Clases para IO



Clases para flujos de entrada

Se utilizan para leer datos de una fuente de entrada (archivo, cadena o memoria)

Flujo de bytes: **`InputStream`**, **`BufferedInputStream`**, **`DataInputStream`**, **`FileInputStream`**

Flujo de caracteres: **`Reader`**, **`BufferReader`**, **`FileReader`**

Clases para flujo de salida

Son las homólogas a las clases de flujo de entrada y se utilizan para enviar flujos de datos a dispositivos de salida

Flujo de bytes: **`OutputStream`**, **`PrintStream`**, **`BufferedOutputStream`**, **`DataOutputStream`** y **`FileOutputStream`**

Flujo de caracteres : **`Writer`**, **`PrintWriter`**, **`FileWriter`**

Clases de Archivo

`File` y **`RandomAccesFile`** (mayor control sobre los archivos)

La clase File

- La clase java.io.File permite obtener la información de las propiedades de un archivo. No es posible escribir o leer el contenido de un archivo, pero es útil para determinar el estado del mismo (p.e. si es un archivo o directorio). La forma general de construcción es:

```
File obj = new File(ruta_del_archivo);
```

- Si el archivo no existe, aún así el objeto se construye.

La clase RandomAccessFile

- Los archivos de acceso aleatorio se caracterizan porque es posible mover el apuntador de archivo a cualquier posición dentro del mismo y efectuar operaciones de escritura y lectura a la vez. La forma general de construcción es:

```
RandomAccessFile arch = new
RandomAccessFile(String nombre, String
modo);

RandomAccessFile arch = new
RandomAccessFile(File obj, String modo);
```

Flujos de archivos

- Otra forma de manipulación de archivos es tratándolos como flujos continuos de bytes. Cada archivo entonces sería una cinta con la secuencia de información para un archivo dado.
- A diferencia del acceso aleatorio, en éstos se presenta un acceso secuencial. Al inicio, el puntero de archivo se coloca en el primer byte y tendrá que ir consumiendo bytes para colocarse en los siguientes bytes hasta llegar al final.

... Flujos de archivos

- Si se desea leer desde el principio el archivo se tendrá que “rebobinar” la cinta y volverlo a leer.
- La clase **InputStream** es el flujo de entrada, mientras que **OutputStream** lo es para salida. Java ofrece subclases de estas clases base que especializan la interpretación de los bloques de bytes. Por ejemplo, **FileInputStream** asocia un archivo como un flujo de entrada, mientras que **FileOutputStream** trata a un archivo de escritura como un flujo de salida.

... Flujos de archivos

- Los métodos clásicos son read() y write(), respectivamente

Flujos de objetos

- Es posible escribir y leer objetos de un flujo de cualquier clase. Las clases **ObjectInputStream** y **ObjectOutputStream** ofrecen la posibilidad de escribir tanto datos primitivos y objetos.
- Lo interesante de estas clases es que nos permiten “guardar” permanentemente el estado de un objeto y posteriormente recuperarlo. Por ejemplo, una ventana pudiera ser guardada en archivo y luego, restaurarla.

... Flujos de objetos

- Veamos la “magia” de esta clase. Por principio de cuentas, la serialización es el proceso de transferir los bytes correspondientes a un objeto a través de un flujo. Este proceso garantiza que el objeto se transmite completamente (de allí su nombre).
- Un objeto Java serializable es uno cuya clase implementa la interface **Serializable**. Esta interface no posee métodos, solo indica a la máquina virtual que ese objeto puede transmitirse por un flujo.
-

... Flujos de objetos

- La forma genérica de un objeto serializable es:

```
import java.io.*;
public class NomClase implements
Serializable {
//Código clase
}
```

Actividad

- Hagamos un pequeño programa que construya un objeto y lo guarde en archivo. Otro programa leerá el archivo y le “volverá a dar vida” al objeto.

La clase BufferedReader

- Almacenamiento temporal en un buffer, para no actuar directamente sobre el stream.
- Igual que los streams de bytes se deben cerrar explícitamente para liberar sus recursos asociados (close).

```
BufferedReader in  = new BufferedReader(new
FileReader(new File(ruta));
```

- Para leer se utliza readLine()

La clase BufferedWriter

- Para la escritura solo cambia el método

```
BufferedWriter out  = new BufferedWriter(new
FileWriter(new File(ruta));
```

- Para escribir se utiliza write()

Actividad

- Generar un programa que lea un archivo, mientras lee cada renglon, deberá mostrarlo con un contador en pantalla y deberá detenerse al encontrar la palabra END.

Tarea

- Investigar las diferencias entre las clases para lectura de archivos y evaluar la mas conveniente para las prácticas de este semestre.
-