10.B. Flujos.

4. Flujos basados en caracteres.

Las clases orientadas al flujo de bytes nos proporcionan la suficiente funcionalidad para realizar cualquier tipo de operación de entrada o salida, pero no pueden trabajar directamente con **caracteres Unicode**, los cuales están **representados por dos bytes**. Por eso, se consideró necesaria la creación de las clases orientadas al flujo de caracteres para ofrecernos el soporte necesario para el tratamiento de caracteres.

Para los flujos de caracteres, Java dispone de dos clases abstractas: `Reader` y `Writer`.

`Reader`, `Writer`, y todas sus subclases, reciben en el constructor el objeto que representa el flujo de datos para el dispositivo de entrada o salida.

Hay que recordar que **cada vez que se llama a un constructor se abre el flujo de datos y es necesario cerrarlo** cuando no lo necesitemos.

Existen muchos tipos de flujos dependiendo de la utilidad que le vayamos a dar a los datos que extraemos de los dispositivos.

Un flujo puede ser envuelto por otro flujo para tratar el flujo de datos de forma cómoda. Así, un `bufferWriter` nos permite manipular el flujo de datos como un buffer, pero si lo envolvemos en un `PrintWriter` lo podemos escribir con muchas más funcionalidades adicionales para diferentes tipos de datos.

En este ejemplo de código, se ve cómo podemos escribir la salida estándar a un fichero. Cuando se teclee la palabra "salir", se dejará de leer y entonces se saldrá del bucle de lectura.

Podemos ver cómo se usa `InputStreamReader` que es un puente de flujos de bytes a flujos de caracteres: lee bytes y los decodifica a caracteres. `BufferedReader` lee texto de un flujo de entrada de caracteres, permitiendo efectuar una lectura eficiente de caracteres, vectores y líneas.

Como vemos en el código, usamos `FileWriter` para flujos de caracteres, pues para datos binarios se utiliza `FileOutputStream`.

```
try{
  PrintWriter out = null;
  out = new PrintWriter(new FileWriter("c:\\salida.txt", true));

  BufferedReader br = new BufferedReader(
            new InputStreamReader(System.in));
  String s;
  while (!(s = br.readLine()).equals("salir")){
          out.println(s);
  }
  out.close();
 }
catch(IOException ex){
 System.out.println(ex.getMessage());
```

Mismo código copiable:

```
try{
	PrintWriter out = null;
	out = new PrintWriter(new FileWriter("c:\\salida.txt", true));

	BufferedReader br = new BufferedReader(
		new InputStreamReader(System.in));
	String s;
	while (!(s = br.readLine()).equals("salir")){
		out.println(s);
```

```
        }
        out.close();
      }
    catch(IOException ex){
            System.out.println(ex.getMessage()); }
```

Autoevaluación

Indica si es verdadera o falsa la siguiente afirmación:

Para flujos de caracteres es mejor usar las clases Reader **y** Writer **en vez de** InputStream **y** OutputStream**.**

- ○ Verdadero.
- ○ Falso.