Almacén de datos

El almacenamiento de datos en variables o vectores (arrays) es temporal, los datos se pierden cuando el programa termina.

Las computadoras utiliza ficheros para guardar datos, llamados datos persistentes, pues existen más allá de la ejecución del programa.

Las operaciones que constituyen un flujo de información del programa con el exterior se denominan Entrada/Salida (E/S).

- E/S estándar - user
- E/S ficheros - disco

EXCEPCION

Cuando trabajamos con archivos, pueden ocasionar errores como no leer o no existir. Para manejar estos errores se utilizar excepciones.

FileNotFoundException
No encuentra archivo

IOException
Sin permiso de lectura o escritura, o dañado.

```
try{
}catch(IOE e){
}
```

flujos

Flujos de caracteres:
Se usan para manipular datos legibles por humanos, vienen determinados por clases abstractas **Reader** y **Writer** (16 bits).

Flujos de bytes:
Se usan para manipular datos binarios, legibles sólo por la maquina. Las clases abstractas que los determinan son **InputStream** y **OutputStream** (8 bits).

- ☒ BufferedInputStream
- ☒ BufferedOutputStream
- ☒ FileInputStream
- ☒ FileOutputStream
- ☒ StreamTokenizer
- ☒ StringReader
- ☒ StringWriter

Con la clase BufferedInputStream añadimos un buffer a un flujo FileInputStream, mejorando la eficiencia de los accesos a dispositivos que se almacena el fichero.

file()

Un **fichero** es un conjunto de información relacionada que se almacena en un ordenador con un determinado nombre para poder recuperarlo.

✓ ficheros de texto se pueden ver y editar con un simple editor de texto.

✓ Ficheros binarios almacenan información utilizando un formato que requiere un software específico para procesarlo.

¿Cómo leemos un fichero de texto?

1. Creamos objeto que represente el fichero con la clase **File()** de java.io.

2. Conectamos el fichero real con la clase **Scanner()** para poder leerlo.

3. Cerrar la conexión con el fichero con el método **close()**.

@romii278

¿Cómo escribimos un fichero de texto?

1. Crear el objeto que represente al fichero con `file()`.

2. Conectar el fichero real para poder escribir en él con la clase `PrintStream()`.

3. Cerrar la conexión con el fichero con `close()`.

La clase File() proporciona una representación abstracta de ficheros y directorios.

- RenameTo()
- Delete()
- setLastModified()
- List()
- hashNext()
- Next()

Para comprobar si existe o se lee usaremos canRead() o exists().

Para el acceso aleatorio usaremos la clase `RandomAccessfile()` con modo 'r' de sólo lectura y 'rw' de lectura y escritura.

Para convertir un objeto en una secuencia de bytes y guardarlo en un archivo hablamos de Serialización y usamos la interfaz `java.io.Serializable` utilizando las clases `ObjectInputStream` y `ObjectOutputStream`. Con el método writeObject() guardaremos el objeto.

Arrays {}

Los arrays permiten almacenar una colección de objetos o datos del mismo tipo.

× Declaración = tipo[] n
× Creación = n=new tipo[]

[Unidimensional / Multidimensional]

Para su creación, indicaremos el tipo de dato, nombre de la variable y nº de elementos (índice).

```
int n[] = new int[5];
```

Para declarar podemos definirlo en su inicialización con {} o desde cada índice.

```
int[]n = {1,5,9,8};
n[1] = 5;
```

Para conocer la longitud del array usaremos el bucle `for()` con `length`.

- ✓ Array.toString()
- ✓ Arrays.fill()
- ✓ Arrays.copyOf()
- ✓ Arrays.equals()
- ✓ Arrays.sort()
- ✓ Arrays.binarySearch()

@romii278

```
public class file {

  public static void main (Str[]) {

    File f= new (File ruta ("archivo.txt"));
     Sout (f.getAbsolutePath());

    String cadena;
    Scanner entrada = null;

    try {
      entrada = new Scanner (f);
      while (entrada.hasNext()) {
          cadena = entrada.nextLine();
          Sout (cadena);
    } catch (FileNotfoundException e) {
      Sout(e.getMessage());
      entrada.close();
    }
```