TEMA 3 - JAVA

3.1 INTRODUCCIÓN A JAVA

List, Map, Set → INMUTABLE: ni añadir ni eliminar elementos una vez inicial
ArrayList → MUTABLE: métodos array.add(sth), array.remove(sth), array.addAll(array

HASHSET: Implementado usando una tabla hash. COMPLEJIDAD O(1)
- add()
- remove()

Utiliza equals() para detectar duplicaciones
HashSet no garantiza el orden.
HashSet es más rápido que TreeSet.

TREESET: Implementado utilizando un árbol rojo-negro (un árbol binario de búsqueda equilibrado). Complejidad O(log(n))
- add(), remove()
- first(), last()
- headSet(), tailSet()

Utiliza compareTo() para detectar duplicaciones.
TreeSet mantiene el orden con el método Comparable or Comparator.

HASHMAP: Implementado utilizando una tabla hash.
Guardada los datos con el formato (clave, valor) : (key, value)
No garantiza el orden de las claves.
Más eficiente que un Treemap.

TREEMAP: Implementado utilizando un árbol rojo-negro (un árbol binario de búsqueda equilibrado).
Garantiza el orden de los elementos (mediante la clave o ke

LEER Y PROCESAR UN ARCHIVO DE TEXTO

`import java.io.*;`

```
BufferedReader buffer = new BufferedReader(
                        new InputStreamReader(
                        new FileInputStream("nombre.txt")));
```

throws IOException

```
while ((linea = buffer.readLine() != null) {
}
```

(linea: String)

```
while ((linea = buffer.readLine()!=null){
    for(String palabra : linea.split(" ")
}
```

(" " : separator)

```
buffer.close();
```

Está bastante bien:

```
import java.util.*;
import java.io.*;
import java.nio.*;
```

```
List<String> lineas = Files.readAllLines(Paths.get("nombre.txt"));
```

(List: inmutable cuidado)

ó

```
Files.lines(Paths.get("nombre.txt")).forEach(linea -> {

});
```

SALIDA DE UN FICHERO CON FORMATO

```
PrintWriter salida = new PrintWriter(new FileOutputStream("nombre.txt"));

for(double i=0.15; i<=0.2; i=i+0.2){
    salida.printf("%5.2f\n", i);
    salida.flush();
}
```

```
salida.close();
```

PERSISTENCIA (entrada/salida) DE OBJETOS

Se guardan los objetos en formato binario.

Un FileOutputStream se convierte en ObjectOutputStream.
Un FileInputStream se convierte en ObjectInputStream.
→ Casting: responsabilidad del programador.

java.io.*;

implements Serializable

```
ObjectOutputStream salida = new ObjectOutputStream(
                            new FileOutputStream("nombre.txt"));
```

[ej: `List<Punto> puntos = new LinkedList<>(Arrays.asList(new Punto(2,1), new Punto(4,5)));`

```
salida.writeObject(puntos);
salida.close();
```

```
ObjectInputStream entrada = new ObjectInputStream(
                            new FileInputStream("nombre.txt"));

List<Punto> puntos = (List<Punto>) entrada.readObject();
                      casting!

entrada.close();
```

LIMITACIONES
↳ Si cambia la clase del objeto serializado, posiblemente los objetos guardados serán ilegibles.

⇒ SIN DUDA, es mejor QUE hacerlo con conversión a texto.

(3.2) CLASES Y OBJETOS EN JAVA

```
public class CuentaBancaria {

    * static long nmrCuentas = 0;          // VARIABLES DE CLASE

    * long numero;                         // VARIABLES DE INSTANCIA
    * String titular;
    * long saldo;

    * = private

    public CuentaBancaria(long num, String tit, long sal ){   // CONSTRUCTOR
        nmrCuentas += 1;
        numero = num;
        titular = tit;
        saldo = sal;
    }

    public void ingresar(long cantidad){                      // MÉTODOS DE INSTANCIA
        saldo += cantidad;
    }

    public void retirar(long cantidad){
        if(cantidad > saldo)
            System.out.println("Saldo insuficiente);
        else saldo -= cantidad;
    }

    public static int getNumCuentas(){                        // MÉTODOS DE CLASE
        return nmrCuentas;
    }
}
```

- **CONSTRUCTORES**

- No son exactamente métodos
- No se invocan sobre un objeto
- No son componentes del objeto
- No son accesibles desde otros métodos
- Sin dato de retorno
- Puede haber varias constructores

Importante: patrón Singleton

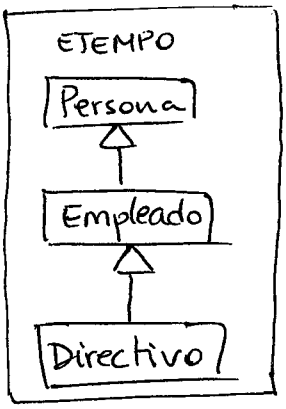
- El constructor privado impide la instanciación de otras clases.
- Solo puede existir un objeto

```
private Constructor(){}
public static Object getInstan
```

3.3 y 3.4) JERARQUÍAS DE CLASES Y CONTROL DE ACCESO

```
Directivo dir = new directivo();
Empleado emp = dir;
Persona p = dir;

((Empleado) p).sueldoBruto = 1000; // OK
((Directivo) p).sueldoBruto = 1000; // OK
((Directivo) emp).fijarIncentivo(0); // OK
```



⊛ No obstante, los castings han de evitarse siempre que sea posible

- Sobreescritura
- Sobrecarga
- Encubrimiento de variables