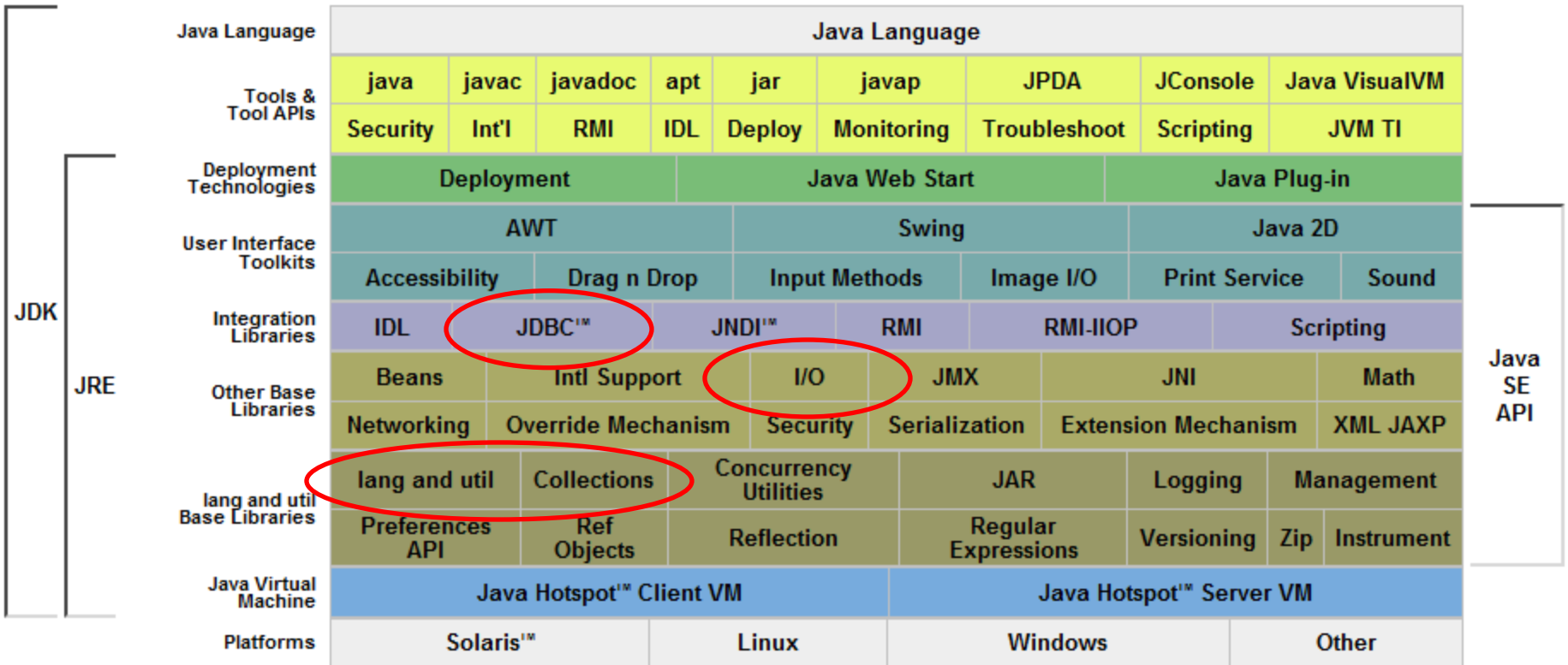


**MASTER EN MODELIZACIÓN  
MATEMÁTICA, ESTADÍSTICA Y  
COMPUTACIÓN  
2015-2016**

Curso: Bases de datos y programación  
orientada a objetos  
Parte POO

**Librerías, estructura disponibilidad y uso.  
Entrada / salida (I/O)**

# Entorno de desarrollo de Java



Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

## Class Object

java.lang.Object

```
public class Object
```

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

**Since:**

JDK1.0

**See Also:**

[Class](#)

### Constructor Summary

**Constructors**

Constructor and Description
<code>Object()</code>

### Method Summary

**Methods**

Modifier and Type	Method and Description
protected <code>Object</code>	<code>clone()</code> Creates and returns a copy of this object.
boolean	<code>equals(Object obj)</code> Indicates whether some other object is "equal to" this one.
protected void	<code>finalize()</code> Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
<code>Class&lt;C&gt;</code>	<code>getClass()</code>

# Clases básicas

## java.lang.Object

- ✓•Clone
- ✓•Equals
- ✓•toString
- ✓•hashCode
- Finalize
- getClass

## java.lang.System

- ✓•getenv
- ✓•getProperty, getProperties, setProperty, setProperties y clearProperty
- ✓•Exit
- ✓•Arraycopy
  - setIn, setOut y setErr

java.lang.<objetos asociados a tipos>

- ✓java.lang.String (operador +, main); java.lang.StringBuffer✓
- java.lang.Math; java.lang.StrictMath;
- java.lang.Throwable
- (otras)✓Thread, Process, SecurityManager, ClassLoader, Compiler, Runtime, etc

java.lang

- ✓•Java.util.Date; java.util.Calendar;✓
- Java.util.BitSet
- Java.util.Random
- Java.util.Timer; java.util.TimerTask

## ✓•Java.util.Properties

- Java.util.ResourceBundle

Java.util.Scanner; java.util.Formatter✓

## FRAMEWORK COLECCIONES

- Interfaz List y clases Vector, Stack, ArrayList y LinkedList
- Interfaces Map y SortedMap, y clases Hashtable, HashMap, LinkedHashMap y TreeMap
- Interfaces Set y SortedSet, y clases HashSet, LinkedHashSet y TreeSet
- El interfaz Queue y la clase PriorityQueue.

- ✓•El interfaz Comparator.

- ✓•Arrays (search sort)

EL SUBPAQUETE java.util.zip✓

java.util

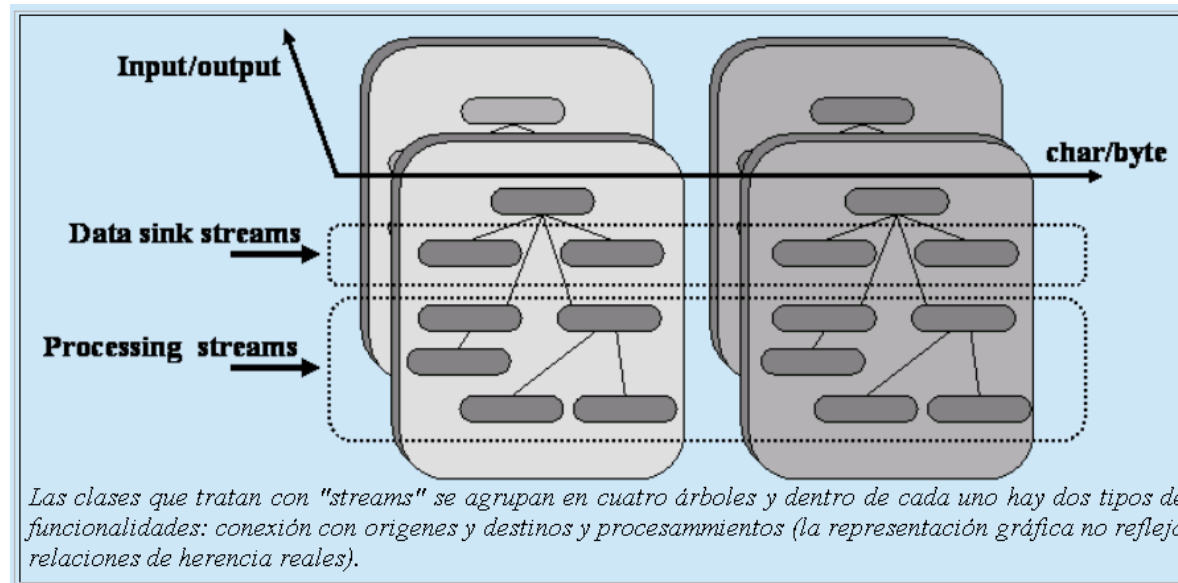
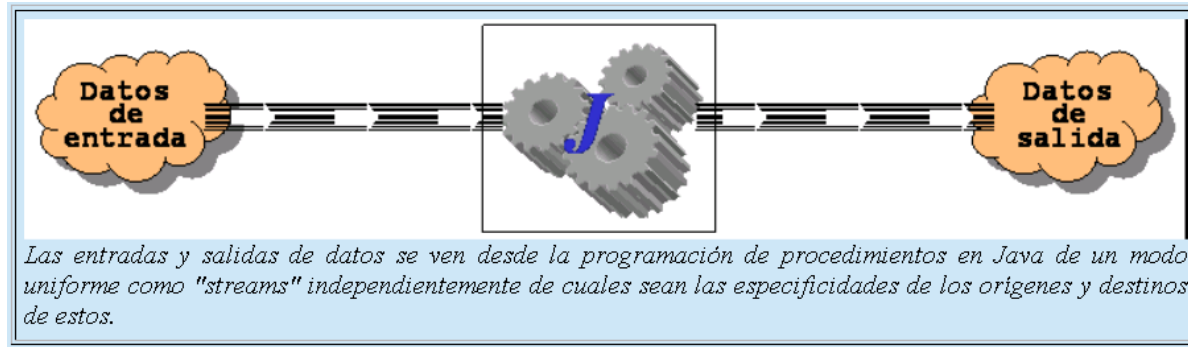
El "for" para colecciones

```
for (E e : Colección<E>) {}
```

p.ej.

```
List<String> sl=new ArrayList();  
//introducir elementos en sl  
for (String s:sl){  
    System.out.println(s);  
}
```

## Clases básicas: entrada/salida



**Reader  
Writer**

**InputStream  
OutputStream**

Origen/Destino	Streams de caracteres	Streams de Bytes
Memoria	CharArrayReader CharArrayWriter	ByteArrayInputStream ByteArrayOutputStream
	StringReader ✓ StringWriter	StringBufferInputStream
"Pipes"	PipedReader PipedWriter	PipedInputStream PipedOutputStream
Ficheros ✓	FileReader FileWriter	FileInputStream FileOutputStream

*Clases de entrada y salida de los orígenes y destinos básicos.*

Procesamientos	Streams de caracteres	Streams de Bytes
Conversión de bytes a caracteres ✓	InputStreamReader OutputStreamWriter	
Buffering ✓	BufferedReader BufferedWriter	BufferedInputStream BufferedOutputStream
Filtrado ✓	FilterReader FilterWriter	FilterInputStream ✓ FilterOutputStream
Concatenación ✓		SequenceInputStream
Conversión de datos		DataInputStream DataOutputStream
Conteo	LineNumberReader	LineNumberInputStream
Peeking Ahead	PushbackReader	PushbackInputStream
Impresión ✓	PrintWriter	PrintStream
Serialización de objetos ✓		ObjectInputStream ObjectOutputStream

*Clases para entradas y salidas con procesamiento de datos.*

Un comentario sobre

- java.io.File
- java.io.RandomAccessFile



```
1- InputStreamReader entrada=new InputStreamReader(System.in);
2- BufferedReader entradaB=new BufferedReader(entrada);
3- //...
4- String s=entradaB.readLine();
```

```
1- BufferedReader entradaB=new BufferedReader(new InputStreamReader(System.in));
2- //...
3- String s=entradaB.readLine();
```

```
BufferedReader br=null;
try {
    br = new BufferedReader(new FileReader("MiFichero.txt"));
    String linea;
    while ( (linea=br.readLine()) != null){
        if (linea.trim().length()==0) continue;
        //TODO aquí lo que se quiera hacer con cada línea
    }
} catch (IOException ex) {
    ex.printStackTrace();
} finally {
    try {
        br.close();
    } catch (IOException ignore) {
        //se da tras FileNotFoundException
    }
}
```



```
try (BufferedReader br = new BufferedReader(new FileReader("MiFichero.txt"))){
    String linea;
    while ( (linea=br.readLine()) != null){
        if (linea.trim().length()==0) continue;
        //TODO aquí lo que se quiera hacer con cada línea
    }
} catch (IOException ex) {
    ex.printStackTrace();
}
```