MASTER EN MODELIZACIÓN MATEMÁTICA, ESTADÍSTICA Y COMPUTACIÓN
2017-2018

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

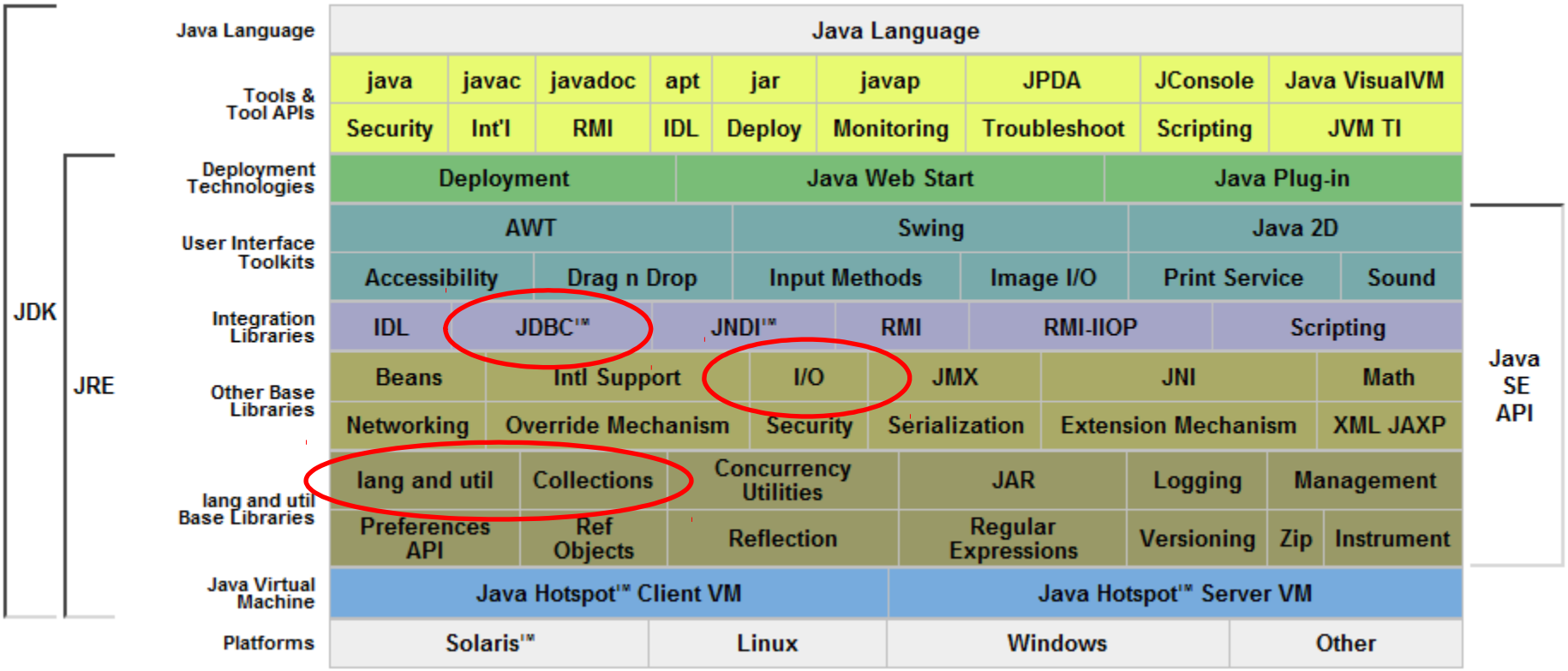
Librerías, estructura disponibilidad y uso.
Clases básicas
Entrada / salida (I/O)

El navegador como capa de cliente (HTML, javascript) y la capa de presentación(JSPs,…)
Acceso a bases de datos (MySQL)

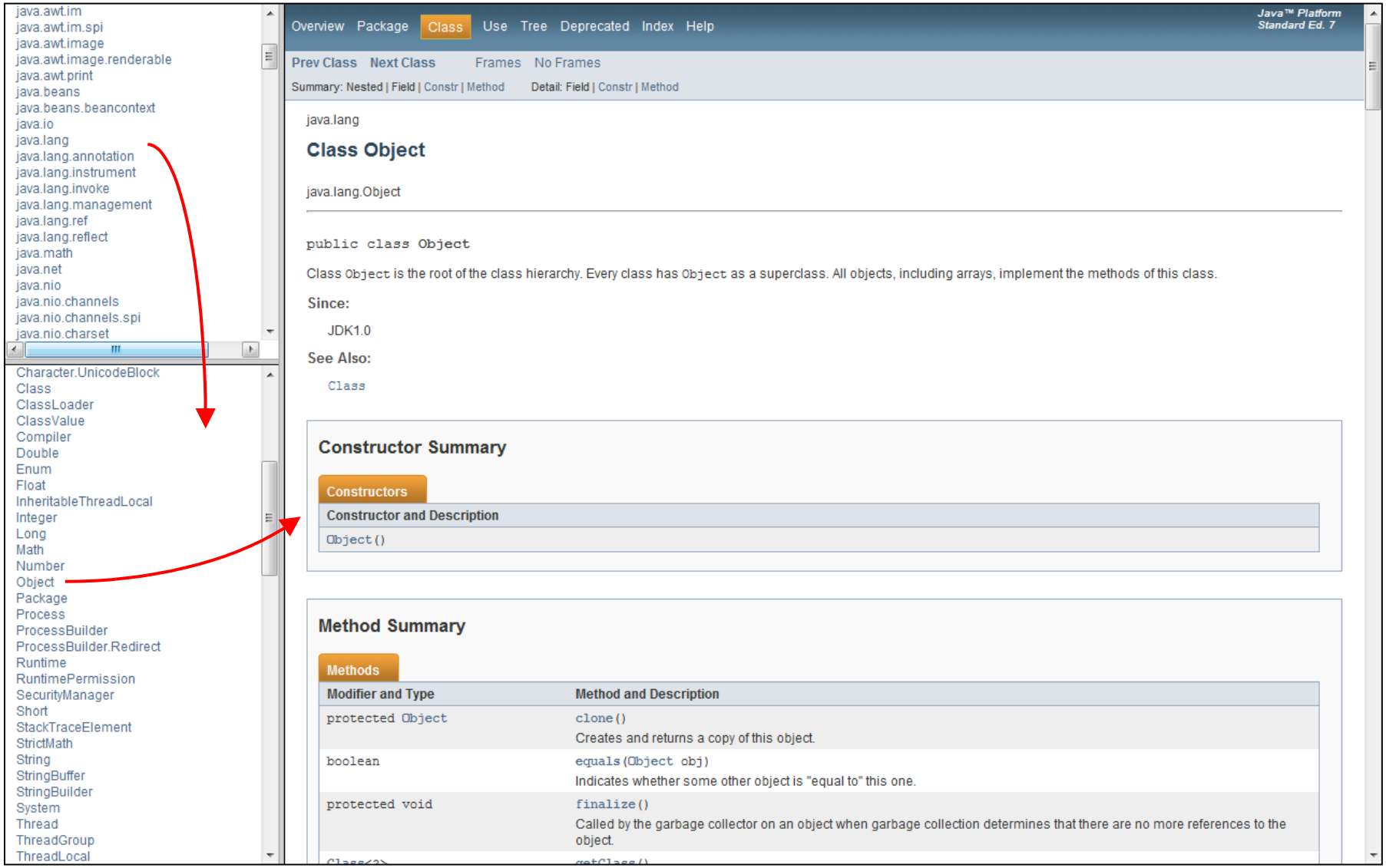
gb
german.bordel@ehu.es
http://gtts.ehu.es/German

Entorno de desarrollo de Java

Imagen global del marco de trabajo Java



gb
german.bordel@ehu.es
http://gtts.ehu.es/German


java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
java.nio.channels
java.nio.channels.spi
java.nio.charset
Character.UnicodeBlock
Class
ClassLoader
ClassValue
Compiler
Double
Enum
Float
InheritableThreadLocal
Integer
Long
Math
Number
Object
Package
Process
ProcessBuilder
ProcessBuilder.Redirect
Runtime
RuntimePermission
SecurityManager
Short
StackTraceElement
StrictMath
String
StringBuffer
StringBuilder
System
Thread
ThreadGroup
ThreadLocal
Overview Package Class Use Tree Deprecated Index Help
Java™ Platform Standard Ed. 7
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
Object()
Method Summary
Methods
Modifier and Type
Method and Description
protected Object
clone()
Creates and returns a copy of this object.
boolean
equals(Object obj)
Indicates whether some other object is "equal to" this one.
protected void
finalize()
Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.

Clases básicas

java.lang

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,ClassLoade,Compiler, Runtime,etc

java.util

- 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

El “for” para colecciones

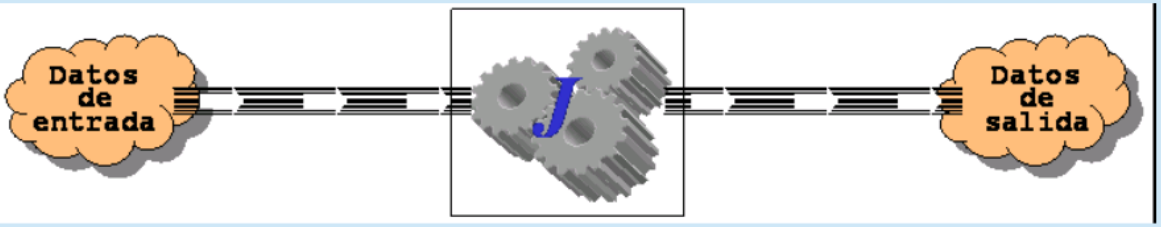
```
for (E e : Colección<E>) {}
```

p.ej.

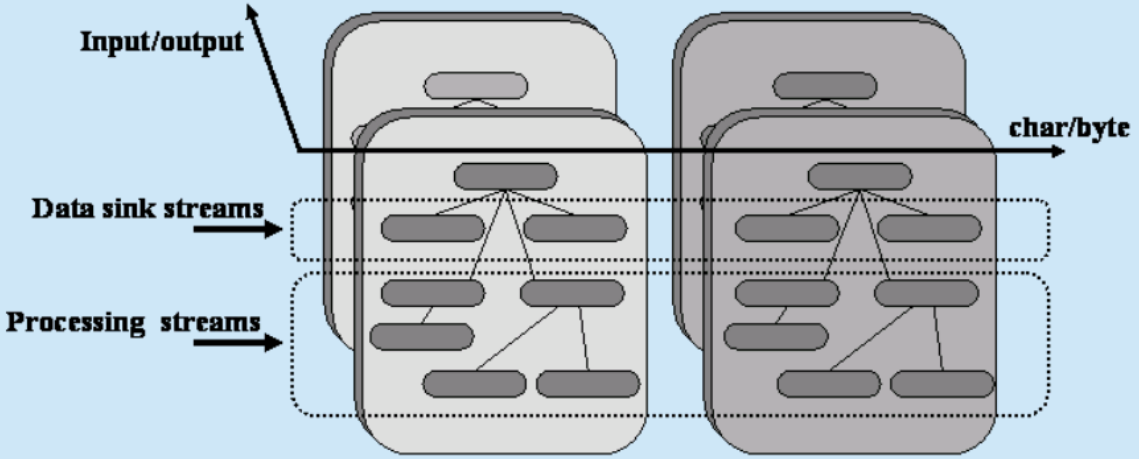
```
List<String> sl=new ArrayList();
//introducir elementos en sl
for (String s:sl){
    System.out.println(s);
}
```

gb
german.bordel@ehu.es
http://gtts.ehu.es/German

Clases básicas: entrada/salida



Las entradas y salidas de datos se ven desde la programación de procedimientos en Java de un modo uniforme como "streams" independientemente de cuales sean las especificidades de los orígenes y destinos de estos.



Las clases que tratan con "streams" se agrupan en cuatro árboles y dentro de cada uno hay dos tipos de funcionalidades: conexión con orígenes y destinos y procesammientos (la representación gráfica no refleja relaciones de herencia reales).

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

Reader
Writer

InputStream
OutputStream

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.

Hay algunas cosas más…

java.io.File
java.io.StreamTokenizer
java.io.RandomAccessFile
...

✓✓

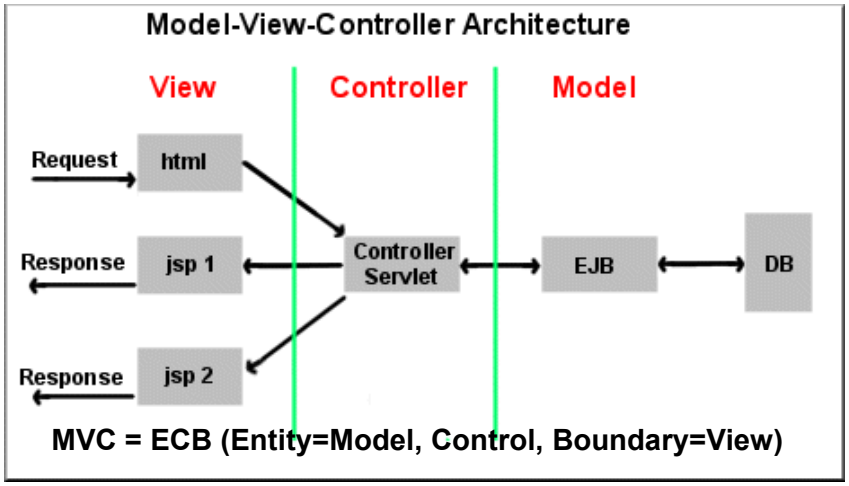
```
1- InputStreamReader entrada=new InputStreamReader(System.in));
2- BufferedReader entradaB=new BufferedReader(entrada);
3- //....
4- String s=entradaB.readLine();
```

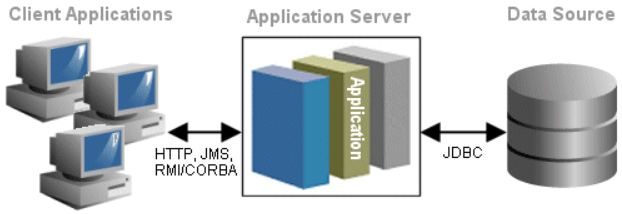
```
1- BufferedReader entradaB=new BufferedReader(new InputStreamReader(System.in));
2- //....
3- String s=entradaB.readLine();
```

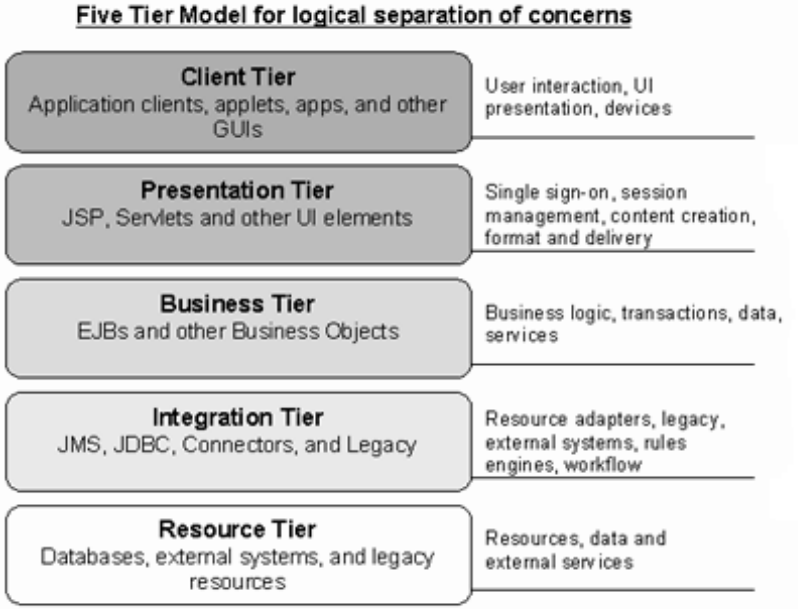
```
BufferedReader br=null;
try {
    br = new BufferedReader(new FileReader("MiFichero.txt"));
    String linea;
    while ( (linea=br.readLine()) != null){
        if (linea.trim().length()==0) continue;
        //TODO aquí lo que se quiera hacer con cada línea
    }
} catch (IOException ex) {
   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

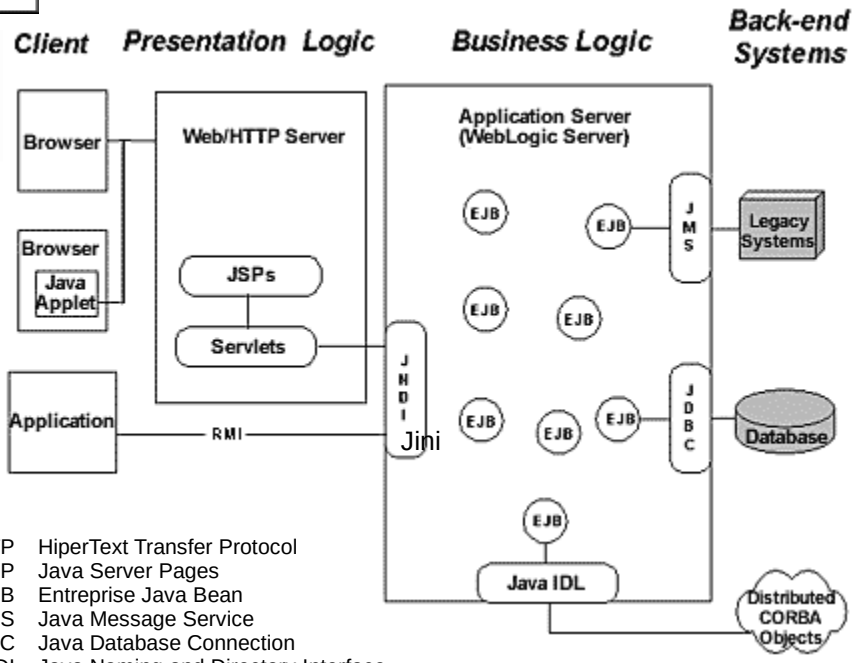
```
try ( BufferedReader br = new BufferedReader(new FileReader("MiFichero.txt")) ){
    String linea;
    while ( (linea=br.readLine()) != null){
        if (linea.trim().length()==0) continue;
        //TODO aquí lo que se quiera hacer con cada línea
    }
} catch (IOException ex) {
    ex.printStackTrace();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

Estructura de una aplicación: capas (tiers)









HTTP	HiperText Transfer Protocol
JSP	Java Server Pages
EJB	Entreprise Java Bean
JMS	Java Message Service
JDBC	Java Database Connection
JNDI	Java Naming and Directory Interface
RMI	Remote Method Invocation
Java IDL	Java Interface Description Language (CORBA)
CORBA	Common Object Request Broker Architecture
JINI	(Jini is not Initials)(diablo en Swahili) infraestructura para SOOA
SOOA	Service-Object-Oriented Architecture

gb
german.bordel@ehu.es
http://gtts.ehu.es/German

Algunas tecnologías (mínimamente) necesarias para 5 capas…

HTML + JavaScript : (HiperText Markup Language)
Ver mi curso páginas web (que es extremadamente antiguo y básico, pero por eso vale como introducción mínima)

Servlets/JSPs (Java Server Pages)
Ver curso en la web

Tecnologías avanzadas…

AJAX (Asyncronous JavaScript And XML)
Frameworks… (Hibernate,Spring,...)
WebServices…

...todas estas tecnologías serían impensables sin objetos.

Acceso a bases de datos (MySQL)

(1) Disponer de la base de datos. El modo más general de acceder consiste en configurarla como un servicio en un determinado puerto (si reside en la misma máquina que la aplicación puede accederse también igualmente a través de "pipes").

(2) Establecer la conexión desde nuestra aplicación con la base de datos, cosa que se hará a través de un driver específico para la base concreta que usemos. (si pensamos en MySQL, el driver lo encontramos en la librería Connector/J)

```
import java.sql.*
try {
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/DataBase","usuario", "clave");
//Continúa la aplicación
    ResultSet rs = con.createStatement().executeQuery("select * from autores");
}
catch(SQLException ex) {
        System.err.println("Problema de conexión con la base de datos: “+ex.getMessage());
}
catch(ClassNotFoundException ex) {
        System.err.println("No se encuentra el driver JDBC: “+ex.getMessage());
}
catch(InstantiationException ex) {
        System.err.println("No puede instanciarse el driver JDBC: “+ex.getMessage());
}
catch(IllegalAccessException ex) {
        System.err.println("Intento de acceso ilegal al driver JDBC: “+ex.getMessage());
}
```