JAVA 

1. Introducción

1. Características

- **Portabilidad**
  - Interpretado (bytecodes)
  - Uso de Máquina Virtual Java (JVM)
- **Orientado a Objetos**
  - Jerarquía de clases
- **Extensible (packages)**
  - Swing (JFC), java3D, RMI, Beans, JDBC

1. Características (cont.)

- Sintaxis similar a C / C++
- Sin punteros: garbage collection
- 100% portable
- Integra librerías estándar para:
  - Interfaces de usuario
  - Objetos distribuidos
  - Threads
- Ejecutable en navegadores web (Applet, JSP, servlets)
- Versiones:
  - 1995 - JDK 1.0
  - 1997 - JDK 1.1
  - 1998 - JDK 1.2
  - 2000 - JDK 1.3
  - 2001 - JDK 1.4

2. Entorno de desarrollo

http://java.sun.com

JDK: Kit de desarrollo de aplicaciones Java

\jdk1.3.1\bin\javac	compilador (*.java)
\jdk1.3.1\bin\java	ejecuta (*.class)
\jdk1.3.1\lib\classes.zip	librerías

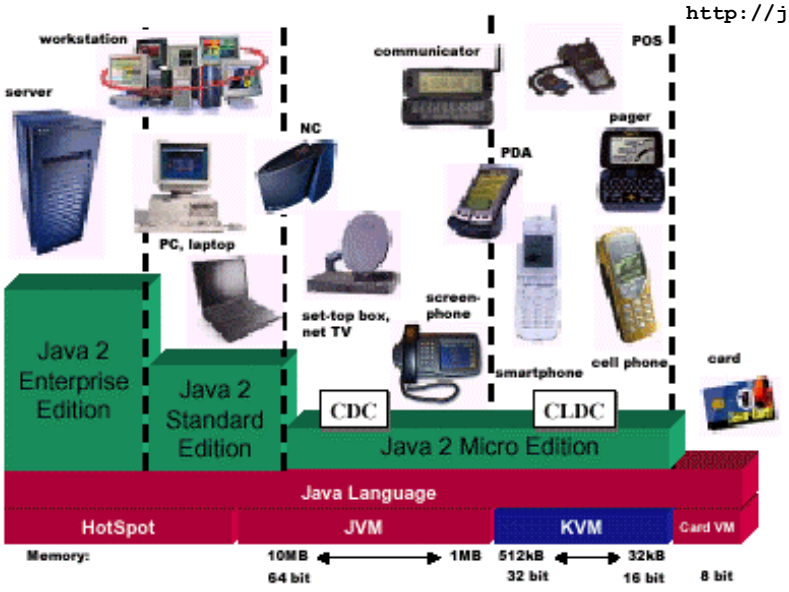
JRE: Máquina virtual Java (JVM)

```
set JAVAPATH=C:\jdk1.3.1
set PATH=.;%JAVAPATH%\bin;%PATH%
set CLASSPATH=.;c:\jdk1.3.1\lib\classes.zip
```

javac –classpath c:\...\classes.zip hola.java

2. Entorno de desarrollo: Plataformas

http://java.sun.com



J2EE

J2SE

J2ME

2. Entorno de desarrollo visuales

JBuilder

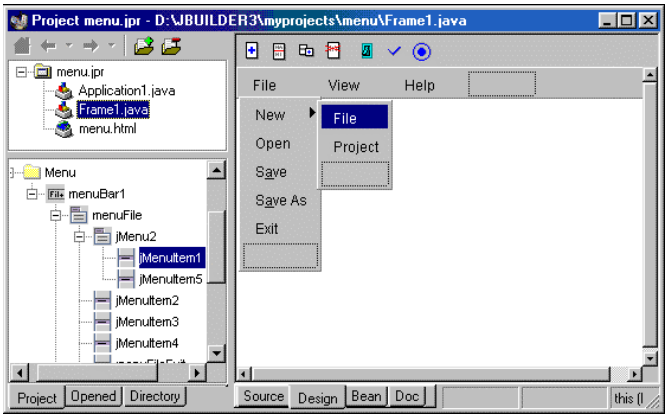
Forte For Java

Simantec Café

NetBeans

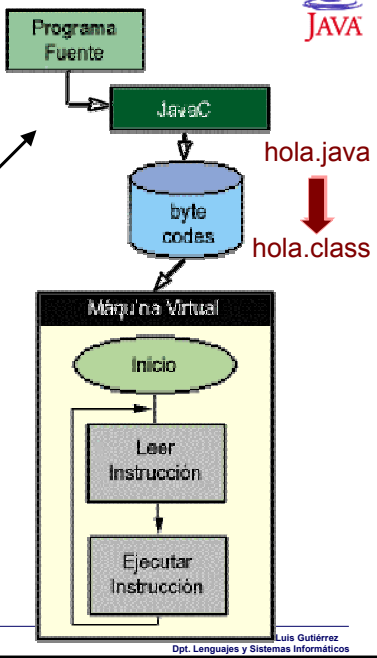
Forte

IBM Visual Age for Java



3. Compilación-Ejecución

```
/* hola.java */

public class Hola
{
  String s = "Hola mundo";

  public static void main (String[] args)  {
      System.out.println("Java dice "+ s);
   }

}
```



4. Conceptos

```
/* hola.java */
import java.awt.*;

public class Hola
{
  String s = "Hola mundo";

  public static void main (String[] args)  {
      System.out.println("Java dice "+ s );
     System.out.println(" longitud "+ s.length(); );

  }

}
```

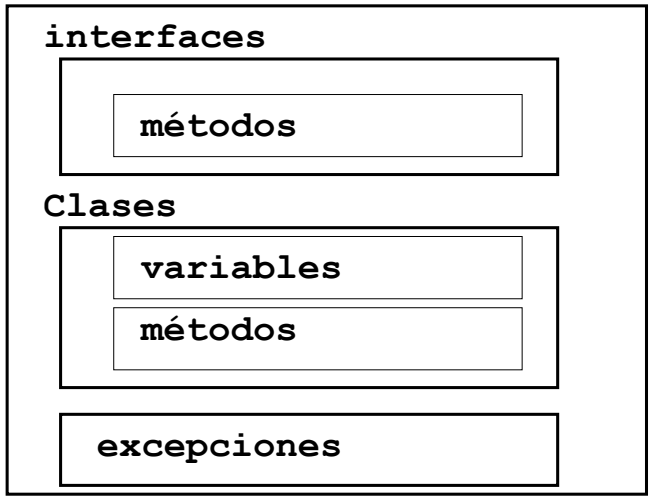
Uso de paquetes (import)

Definición de clases

Declaración y creación de objetos

Métodos (main)

4. Orientado a objetos



paquetes
- java.awt
- java.Swing
- java.io
- java.util
- java.sql
- ... java.rmi

4. Orientado a objetos

Organización del código en clases (objetos)
Nombre fichero igual que la clase (Alumno.java)

```
package Centro;
import java.io.*
public class Alumno {
      String nombre;
      int edad;
      public Alumno () {
            nombre = "Pepe"; edad=10;
       }
      public ponerEdad(int x) {
           edad=x;
           System.Out.println("edad ="+edad)
           }
}
```

Librerías usadas

Variables de instancia

Constructor (no existe destructor)

Métodos

4. Orientado a objetos

- Organización lógica de clases en paquetes (packages)

  Importación de paquetes para su uso:

```
import java.io.*; import java.awt.*
```

- Se pueden **extender** la funcionalidad de las clases de un paquete

```
public class Hola extends
    java.applet.Applet { ... }
```

- Modificadores acceso (clases/métodos):

public	**Visible dentro y fuera del paquete**
protected	**Visible en paquete y herencia**
privated	**No es visible**

4. Orientado a objetos

- No posee destructor (*garbage collection*)
- Llamada a métodos (*objeto.método*):

```
Circulo ovalo = new Circulo(10,10, 30,30);
ovalo.draw();
```

- Interfaces = Herencia múltiple y clases abstractas

```
public interface Grafico {  void draw();  }
class Circulo implements Grafico {  . . . }
```

- Excepciones. Asociadas a métodos

```
public class Alumno {
  public void almacenar(String) trows IOException {...}
```

- Manejo excepciones mediante try - catch

```
try { ... } catch (Exception e) { ... }
```

5. Sintaxis

Sensible a mayúsculas/minúsculas

Uso: **N**ombre**C**lase, **n**ombre**Método**, **v**ariables, CTES

Delimitación de:

caracteres: (‘a’), cadenas (“hola”)

Constantes lógicas: true / false

Sentencias separadas por punto y coma (;), bloques: {... }

Comentarios:

// hasta final de la linea,

/* múltiples líneas */

5. Sintaxis

Tipos Básicos: byte, char, short, int, long, float, double, boolean

Operadores:

Aritméticos: ++, --, +, -, *, /, % ..

Lógicos: ! (NOT), & (AND), ^ (XOR), | (or) ..

Relacionales: == , !=, >=, <=, ...

Variables: tipo Identif (= “valorDefecto”);

```
char letra ='a';
int  valor = 0;
boolean estado = true;
```

5. Sintaxis

Tipos estructurados Array

Declaración: *tipo Identif [ ] = { valorDefecto } ;*

```
int a[];
float c[][];   // multidimensional
```

Asignación:

```
int b[] = { 3, 1, 2, 4};     // al declarar
float cood[][] = {{3, 2.1},{6, 1}};
a = new int [4];
int v[][] = new int [4][]; // array de arrays
```

Asignación de valores:

```
a[0] =7; a[1] = 3;
v[0] = new int[4];  // asignación de arrays
```


5. Sintaxis

Tipos estructurados String, StringBuffer

Declaración: **String** *Identif [ ] = { valorDefecto } ;*

```
String s = new String ("hola");
float c[][];   // multidimensional
```

Asignación:

```
 s= "Adios";
 s = s1 + s2;
```

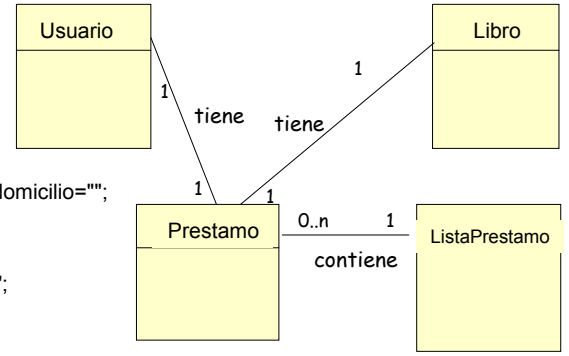
Métodos:

```
char c = s.setCharAt(1,'H');
int i  = s.lenght();
int i = Integer.parseInt(s);    // conversión
String s = Integer.toString(i); // conversión
String s2 = s.toUpperCase();
```

5. Sintaxis: estructuras de control

```
while ( 'condición' ) {  ... }

do { ... } while ( 'condición' );

for ( int x= vmin ; x< vmax; x++) { ... }

if ( 'condición' ) { ... } else { ... }

switch ( ' expresion' ) {

    case 'valor' : ...; break;

     case 'valor':  ...; break;

     default:  ...  }
```


5. Ejemplo: Biblioteca




```
class Usuario {
      String carnet=""; String nombre=""; String domicilio="";
      String localidad=""; String provincia="";
      String fecha_nac="";
      String tlfn=""; String email=""; String foto="";

     public void ver_usuario () { ... }
     public void leer_usuario (String s) throws IOException{ ... }
     public void guardar_usuario () throws IOException{ ..}
     public void mod_nombre ()throws IOException{ .. }
     ...
}  // fin clase
```