Java 2

(Manual FV)

Antes de empezar:

- **Manual F.V.**
  Significa "manual **práctico** de informática", pero realmente realmente PRÁCTICO.

- En el texto me refiero continuamente a *TuCarpeta*, esto es un subdirectorio de tu ordenador donde es conveniente que grabes todos los programas de este "tu manual".

- Todos los programas y ejercicios del manual han sido probados sin problema en el **Java 2 SDK (Standard Edition 1.2.2)**

- Mi consejo, es que escribas de nuevo tal como te indico en el texto, cada uno de los programas utilizando el **Bloc de Notas** del Windows. Sólo en el caso de que el programa no te funcione y no descubras cuál es el error, cópialo a partir del Word en el Bloc de Notas y grábalo con otro nombre; para poder compararlo con el tuyo y así descubrir el error.
  Piensa que se aprende más descubriendo los propios errores, que avanzar sin hacer ningún error.

ÍNDICE

I.- Introducción

Origen del Java

La empresa **Sun Microsystems** decidió introducirse en el mercado de la electrónica de consumo para desarrollar programas para pequeños dispositivos electrónicos (tostadoras, microondas, TV interactiva, etc.), para ello "Sun" creó una filial denominada **FirstPerson Inc**.

James Gosling de "First Person Inc", a partir del **C++** crea un nuevo lenguaje de programación (1991), que llamó **Oak** para solucionar el gran problema de "programación" en la electrónica de consumo:

- En la electrónica de consumo, los chips electrónicos correspondientes cambian muy rápidamente: una pequeña diferencia en el precio de un chip, por ejemplo.
- Al utilizar el lenguaje "C++", si cambiamos el chip, es necesario rehacer todos los programas, para adaptarlos al nuevo dispositivo electrónico.
- Un programa escrito en "OAK" no necesita rehacerse para compilarse de nuevo, al cambiar el chip.

En definitiva, si programamos en "Oak" no es necesario cambiar el programa, si varia el chip donde está implementado.

En 1995 tras un cambio de nombre y mejoras en su diseño se presentó el "Java" en sociedad.

Características del Java

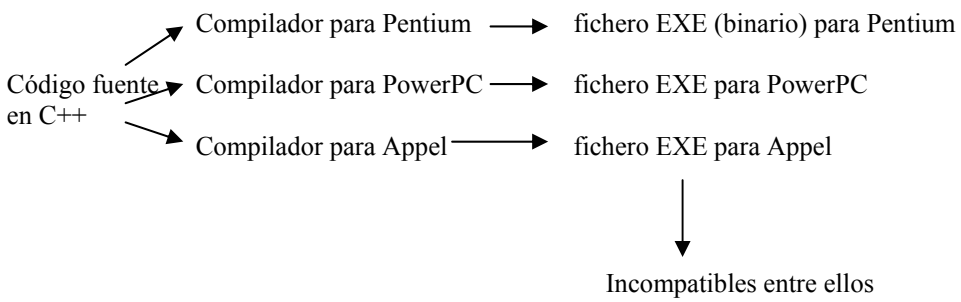
- El "Java" es un lenguaje de programación completo orientado a objetos.

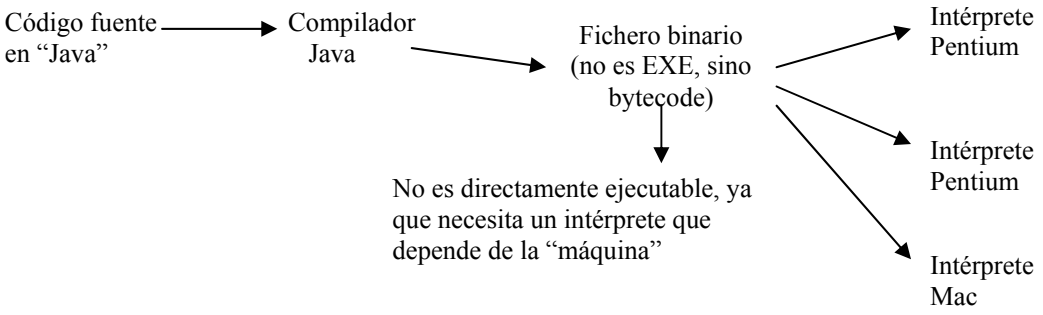
- El "Java" se diseñó a partir del "C++" con el propósito de:
  - Ser reducido
  - Sencillo
  - Rápido
  - Eficiente
  - Transportable (en diversas plataformas y sistemas operativos)

- Con "Java" podemos crear dos tipos de programas:
  - Aplicaciones Completas
  - Applets

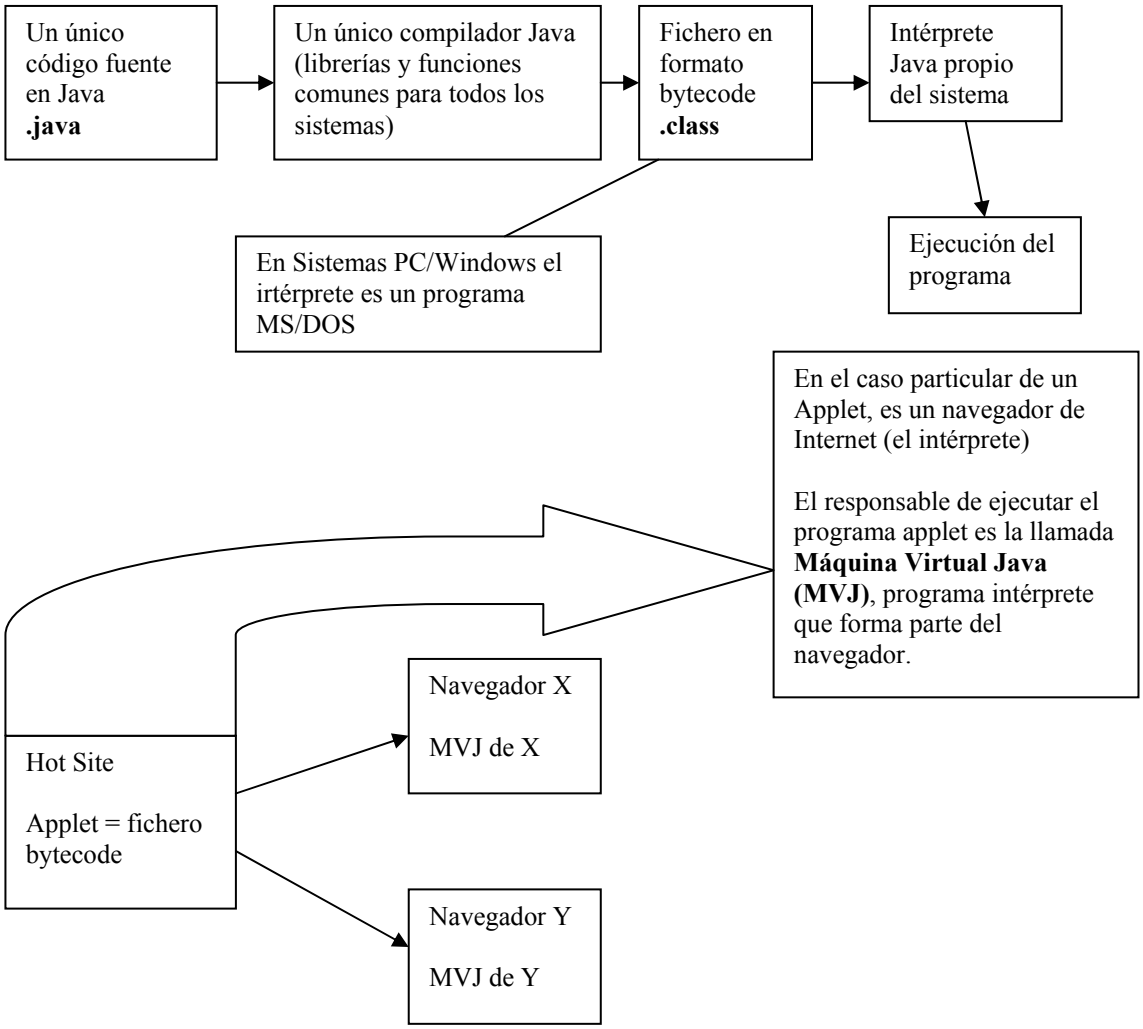
Un "applet" (pequeña aplicación) es un programa java, que se ejecuta en una página web para generar animaciones, gráficos con efectos interactivos, juegos, etc.

- La principal característica del Java es su "independencia de la plataforma":





En definitiva:



Java y JavaScript

- **JavaScript** es una versión de Java directamente interpretada, que se incluye como parte de una página HTML, lo que lo hace muy fácil y cómodo para aplicaciones muy pequeñas, pero que en realidad tiene muchas limitaciones. La principal limitación es que no todos los browsers lo soportan completamente (Explorer, por ejemplo, no soporta las últimas adiciones de Netscape, como las imágenes animadas).

- **JavaScript** es un lenguaje creado por Netscape y Sun.

- **VBScript** (Visual Basic Script) es una copia mejorada de JavaScript creada por Microsoft (en continua guerra con Netscape), que sólo funciona con su navegador (Microsoft Internet Explorer).

- **JavaScript** se le ha llamado el hermano pequeño de Java, pero es bastante menos que eso, ya que Java es un lenguaje de programación general a diferencia de **JavaScript**, que sólo sirve para añadir interactividad a las páginas web.

- **JavaScript** sólo se puede comparar (salvando las distáncias) con los applets de Java.

Veamos con un ejemplo la relación entre JavaScript y un Applet de Java:

- Crea utilizando el Bloc de Notas de Windows, el siguiente fichero:

```
<html>
<script language="JavaScript">
document.write("Esto es un mensaje de JavaScript");
</script>
<body>
<br><br>
Esto forma parte de la página Web
</body>
</html>
```

- graba el fichero anterior en *TuCarpeta* con el nombre **j001.htm**

- Ejecuta el fichero anterior.

- Crea, utilizando el Bloc de Notas, el siguiente fichero:

```
//Applet j002
import java.awt.Graphics;
import java.applet.Applet;
public class j002 extends Applet{
        public void paint(Graphics g){
                g.drawString("Esto es Java",25,25);
        }
}
```

- Graba el fichero anterior en *TuCarpeta*, con el nombre **j002.java**

- Crea el siguiente fichero:

```
<html>
<applet code=j002.class width=100 heigt=100>
</applet>
<body>
```

```
<br><br>
Esto forma parte de la página Web
</body>
</html>
```

- Graba el fichero anterior con el nombre **j003.htm**

- Sitúate en **MS/DOS**, es decir:

  Clic en [Inicio]
  Cursor en "Programas"
  Clic en MS-DOS

- Sitúate en *TuCarpeta*

- Escribe:
  javac j002.java [Return]

Acabamos de compilar el programa java. Es decir, a partir del código fuente en java "j002.java" se ha creado el fichero compilado (binario bytecode) **j002.class**

- Ejecuta el fichero **j003.htm** desde tu navegador.

Compiladores de Java

Hay muchos en el mercado:
- Visual Café de Symantec
- Visual J++ de Microsoft
- Jbuilder de Borland
- Etc.

Nosotros utilizaremos el **Java 2 SDK (Standard Edition 1.2.2)**, que forma parte del kit de desarrollo de java (JDK), que es un regalo gratuito de Sun a la comunidad que programa en Java.

Probablemente lo tendrás en algún CD-ROM de los que acompaña a las revistas de informática. De todas formas puedes bajártelo de Internet en la dirección www.javasoft.com

Al instalar el **JDK de Sun** se crea en nuestro disco duro la carpeta **C:\JDK1.2.2** en cuya carpeta **BIN** tenemos los tres programas básicos:
- Javac.exe, que es el compilador.
- Java.exe, el intérprete para sistemas PC/Windows
- Appletviewer.exe, un visualizador de applets

Si queremos ejecutar los programas "java" desde cualquier carpeta, deberemos incluir en el fichero **Autoexec.bat**, la línea: **PATH=C:\JDK1.2.2\BIN;%PATH%**

El funcionamiento básico del **JDK** es:

- **En una aplicación Java:**
  - Escribimos utilizando un editor de texto (Bloc de Notas del Windows, por ejemplo) el código fuente de nuestro programa Java. Hemos de grabarlo con la extensión **java**, por ejemplo **nombre1.java**
  - Desde MS/DOS y en la carpeta donde tenemos **nombre1.java**, escribimos: **javac nombre1.java** [Return].

De esta forma compilaremos nuestro programa y se creará el fichero binario (bytecode) **nombre1.class**

- Desde MS/DOS y en la carpeta donde tenemos **nombre1.java** y **nombre1.class**, escribimos: **java nombre1** [Return]

De esta forma se "interpreta" el fichero **nombre1.class** en nuestro ordenador PC/Windows, es decir se ejecuta.

- **En un Applet**
  - Igual que en el caso anterior, pero para ejecutar el **applet**, hemos de incluir la llamada correspondiente en una página Web. Es decir, en lugar de utilizar el intérprete **java.exe** de antes, utilizamos el navegador que contiene la MVJ como intérprete.

Los primeros programas

- Escribe utilizando el bloc de notas:

```
class j004 {
        public static void main(String [] args) {
                System.out.println("Bienvenido a Java!");
        }
}
```

- Grábalo en *TuCarpeta* con el nombre **j004.java**

- Compila y ejecuta el programa anterior, es decir:
  - javac j004.java
  - java j004

- Supongo que has deducido para qué sirve la sentencia: **System.out.println("mensaje");**

- Investiga la utilidad de "\n", estudiando el siguiente programa:

```
class j005 {
        public static void main(String [] args) {
                System.out.println("Hola\nQue tal");
                System.out.println("Uno\nDos\nTres\nCuatro");
        }
}
```

- Grábalo con el nombre **j005.java**

- Compílalo y ejecútalo.

- ¿Qué utilidad tiene "\n"?

- Investiga la diferencia entre **print** y **println**, estudiando el siguiente programa:

```
class j006 {
        public static void main(String [] args) {
                System.out.println("Hola");
                System.out.println("Adios");
           System.out.println("");
                System.out.print("Pues vale");
                System.out.print("Eso");
                System.out.print("Vale");
```

```
        }
    }
```

- Grábalo con el nombre **j006.java**

- Compílalo y ejecútalo.

- ¿Qué diferencia hay entre **print** y **println**?

  println("mensaje") después de escribir el "mensaje" se produce un cambio de línea equivale a pulsar [Return]

  print("mensaje") después de escribir el "mensaje", el cursor se sitúa al final del mensaje y en la misma línea.

  Para recordarlo piensa en la siguiente "fórmula":

  println = print + ln (línea nueva)

- Escribe el siguiente programa:

```
// j007.java = dibuja 2 circunferencias
import java.awt.*;
import java.awt.event.*;
class j007 extends Frame {
    public j007() {
        setTitle("Dos circunferencias");
    }
    public static void main(String [] args) {
        Frame f=new j007();
      f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    f.setSize(300,200);
    f.setVisible(true);
    }
    public void paint(Graphics g){
        g.setColor(Color.red);
        g.drawOval(10,30,30,30);
        g.setColor(Color.blue);
        g.drawOval(35,30,30,30);
        g.drawString("Dos circunferencias",40,100);
    }
}
```

- Grábalo con el nombre **j007.java**

- Compílalo y ejecútalo.

- **f.setSize(600,400)**

Determina el tamaño de la ventana
1ª coordenada = 600 = anchura en píxels
2ª coordenada = 400 = altura en píxels.

- **g.setColor(Color,red)**

Establece el rojo como color de dibujo.

- **g.drawOval(10,30,30,30)**

Dibuja una circunferencia cuyo "vértice" superior izquierdo se encuentra en el punto 10,30. La tercera coordenada es el diametro horizontal y la cuarta el diámetro vertical. Como los dos diámetros son iguales, resulta una circunferencia. Si fueran distintos, seria una elipse.

- **g.drawString("Dos circunferencias",40,100)**

"Dibuja" el texto que hay entre comillas en el punto de coordenadas 40,100 (1ª coordenada= distáncia horizontal desde el borde izquierdo, 2ª coordenada = distáncia vertical desde el borde superior).

- A partir del **j007.java** escribe el siguiente programa:

```
// j008.java = dibuja 2 elipses
import java.awt.*;
import java.awt.event.*;
class j008 extends Frame {
        public j008() {
                setTitle("Dos Elipses");
        }
        public static void main(String [] args) {
                Frame f=new j008();
         f.addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
        f.setSize(300,200);
        f.setVisible(true);
        }
        public void paint(Graphics g){
                g.setColor(Color.green);
                g.drawOval(10,30,250,150);
                g.setColor(Color.yellow);
         g.drawOval(40,55,200,125);
                g.setColor(Color.black);
                g.drawString("Dos Elipses",100,100);
        }
}
```

- Grábalo con el nombre **j008.java**

- Compílalo y ejecútalo.

- A partir de los ejercicios anteriores escribe el siguiente programa:

```
// j009.java = Títulos gráficos
import java.awt.*;
import java.awt.event.*;
class j009 extends Frame {
        public j009() {
                setTitle("Dos Elipses");
        }
        public static void main(String [] args) {
                Frame f=new j009();
         f.addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
```

```
                    });
                f.setSize(400,300);
                f.setVisible(true);
                }
                public void paint(Graphics g){
                    g.setColor(Color.red);
                    g.drawString("Hola",50,50);
                    g.setColor(Color.blue);
                    g.drawString("Hola",100,100);
                    g.setColor(Color.green);
                    g.drawString("Hola",200,150);
                    g.setColor(Color.yellow);
                    g.drawString("Hola",300,200);
                }
        }
```

Fundamentos de la Programación Orientada a Objetos

- La programación orientada a objetos **POO** es una forma de programar que se basa en el uso de clases.

- Un programa Java consta de una o más clases.

Una clase consta de **miembros**:

- Propiedades = campos o variables
- Funciones = métodos o habilidades

Ejemplo:

Clase: Arboles

Miembros de "Arboles": propiedades (color, altura etc.) y funciones (crecimiento, etc)

A partir de la clase "Arboles" podemos definir una serie de objetos (elementos o ejemplares): pino, palmera, acacia, roble, etc.

- Creación de objetos:

Sea "Arboles" una clase

Arboles pino = new Arboles();

Acabamos de crear un objeto de la clase "Arboles" de nombre "pino"

Analicemos el programa **j004.java**:

```
class j004 {
    public static void main(String [] args) {
        System.out.println("Bienvenido a Java!");
    }
}
```

- Definimos una clase de nombre **j004** (el fichero debe grabarse con el mismo nombre)

Que consta de un único método = **main**

- **System** es una clase ya definida en "Java" que contiene todas las posibles propiedades y métodos de **entrada-salida**

- **out** es un objeto de la clase "System" que contiene las propiedades y métodos de salida.

- **println** es un método del objeto "out" que imprime en pantalla el mensaje y a continuación hace un [Return]

- **print** es otro método del objeto "out" que imprime en pantalla el mensaje.

- Escribe el siguiente programa:

```
// j010.java=Contador
public class j010 {
        int x;
        public j010() {
                x=0;
        }
        public int incCuenta(){
                x++;
                return x;
        }
        public int getCuenta() {
                return x;
        }
}
```

- Grábalo con el nombre **j010.java**

- Compílalo:
  Javac j010.java y creará el fichero **j010.class**

- No lo ejecutes, porque no hará nada. Veamos:
  - Definimos una clase de nombre **j010**
  - Los miembros de dicha clase son:

Una propiedad:

int x;
No es mas que la declaración de una variable **entera** (int) de nombre **x**

Tres métodos o funciones:

public j010() {x=0;}
Es una función especial llamada **constructor** (observa que tiene el mismo nombre de la clase). Su principal utilidad está en inicializar variables, en nuestro caso **x=0.**

public int incCuenta() {
x++;
return x;
}
Es una función que devuelve el valor de "x" incrementado en una unidad (x++ es equivalente a **x=x+1**).

public int getCuenta() {return x;}
Es una función que devuelve el valor actual de "x".

Lo que acabamos de hacer en **j010.java**, que tenemos compilado en **j010.class** es construir una **CLASE**, que no es mas que un **contador**

La clase o programa **j010** aislado no sirve para nada, vamos a hacer un nuevo programa que utilice la clase **"j010.class"**…

- Escribe el siguiente programa:

```
// j011.java= programa que utiliza la clase j010
import java.io.*;
public class j011 {
        static int n;
        static j010 cuenta;
        public static void main(String [] args) {
                System.out.println("Cuenta...");
                cuenta=new j010();
                System.out.println(cuenta.getCuenta());
                n=cuenta.incCuenta();
                System.out.println(n);
                cuenta.incCuenta();
                System.out.println(cuenta.getCuenta());
                System.out.println(cuenta.incCuenta());
        }
}
```

- Grábalo con el nombre **j011.java**

- Compílalo y ejecútalo.
  Si todo funciona correctamente aparecerá en pantalla:

```
Cuenta …
0
1
2
3
```

  Básicamente lo que hemos hecho es:
  - En **j010.class** tenemos una clase que no es mas que un contador:
    Inicializado a 0 y con dos métodos: **incCuenta** (incrementa en 1 el contador) y **getCuenta** (muestra el valor del contador).
  - En **j011** tenemos un programa **Java** (es decir, otra clase), donde:
    Definimos un objeto de la clase **j010** de nombre **"cuenta"**
    Cuenta = new j010()
    cuenta.getCuenta = 0
    cuenta.incCuenta = 1
    cuenta.incCuenta = 2
    cuenta.incCuenta = 3

- Vamos a hacer un **Applet** que haga lo mismo que el programa anterior:

Escribe:

```
// Applet j012
import java.applet.*;
import java.awt.*;
public class j012 extends Applet {
        static int n;
        static j010 laCuenta;
        public j012() {
                laCuenta= new j010();
        }
        public void paint(Graphics g) {
                g.drawString("Cuenta...",20,20);
            g.drawString(String.valueOf(laCuenta.getCuenta()),20,35);
                n=laCuenta.incCuenta();
```

```
                g.drawString(String.valueOf(n),20,50);
                laCuenta.incCuenta();
            g.drawString(String.valueOf(laCuenta.getCuenta()),20,65);
            g.drawString(String.valueOf(laCuenta.incCuenta()),20,80);
        }
    }
```

- Grábalo con el nombre **j012.java** en *TuCarpeta*

- Compílalo (no lo ejecutes), es decir tendremos el fichero **j012.class**

- Escribe:

```
<html>
<head>
<title>j013.htm</title>
</head>
<body>
<applet code="j012.class" width=170 height=150>
</applet>
</body>
</html>
```

- Graba el fichero anterior en *TuCarpeta*, con el nombre **j013.htm**

- Ejecuta el programa anterior:
  - Podríamos utilizar cualquier navegador.
  - O utilizar el visualizador de applets del SDK:
    Desde MS/DOS:
    appletviewer j013.htm

Continuando con la POO

El modificador "static"
Cada vez que creamos un objeto, se hace una copia de todos los miembros de la clase.

Veamos:
Sea **Arboles** una clase, y **noDePreguntas** un método de **Arboles**.
Sea **pino** y **sauce** dos objetos de la clase **Arboles**.
El método **noDePreguntas** del objeto **pino** es completamente distinto del método **noDePreguntas** del objeto **sauce**.
Supongamos que por la razón que sea, un método (o propiedad) de una clase, nos interesa que sea el mismo para todos los objetos de la clase. En este caso necesitamos un **método (o propiedad) de clase** no **un método (o propiedad) de objeto**.
El modificador **static** determina un método (o propiedad) de clase.

- Analicemos el siguiente programa (es el más sencillo posible, es tan simple que no hace nada):

```
class HacerNada {
        public static void main(String[] args) {
        }
}
```

Se trata de una clase de nombre "HacerNada", que sólo tiene un método de nombre **main**, se trata de un método **void** (no devuelve nada) y tiene un modificador **static**, es decir es un método de clase, que será llamado de la siguiente forma: **HacerNada.main**

Cuando se ejecuta un programa, hemos de dar el nombre de una clase, "Java" busca en dicha clase un método llamado **main** y empieza ejecutando el programa desde allí, como "Java" debe acceder a dicho método, éste debe tener un modificador **public**.

Comentarios

- Escribe el siguiente programa:

```
// j014.java
class j014 {
/* Programa que mostrará en pantalla
   un mensaje de error */
        public static void main(String [] args) {
                System.out.println("----------------------------");
                System.out.println("|                          |");
                System.out.println("|       A V I S O          |");
                System.out.println("|Posible virus destructivo |");
                System.out.println("|                          |");
                System.out.println("----------------------------");
                System.out.println();
                System.out.println("\nEs broma");
        }
}
```

- Grábalo en *TuCarpeta* con el nombre **j014.java**

- Compílalo y ejecútalo.

- Los métodos **println()** y **print()** son métodos de objeto, no de clase, por lo tanto hemos de indicar el objeto correspondiente que es **out**

- Comentarios para el programador:

```
// de una línea
/* podemos utilizarlo en varias líneas */
```

- **System.out.println()**
  Línea en blanco.

Tipos. Variables. Constantes

Java es un lenguaje con **tipos y clases estrictos**, esto es: en una expresión sólo podemos utilizar elementos del mismo tipo o clase, es decir no podemos mezclar cadenas y números, ni árboles ni pájaros.

Java tiene 8 tipos predefinidos conocidos como **tipos primitivos**, pero los tres más usados son:

int : números enteros hasta 2.000.000.000
long : enteros más grandes o para resultados de cualquier operación entera.
double : números reales o para resultados de expresiones en las que interviene como mínimo un número real.
Los otros tipos primitivos son: **boolean, byte, char, float, short**

Declaración de variables

Hay tres posibilidades:

Tipo nombre;
Tipo nombre1, nombre2, nombre3;
Tipo nombre=valor;

Por ejemplo:

```
int temperatura, edad;
double iva=16.5;
```

Declaración de constantes

static final tipo nombre = valor

por ejemplo:

```
static final double pi=3.141592;
```

Muchas veces nos interesará "inicializar" una variable. Por omisión, en casos numéricos, es 0 (por esta razón la inicialización se usa poco en Java).

Expresiones

$\frac{1}{2.5}$	1/2.5
6(7-3)	6*(7-3)
$\frac{100 - 9.99}{15 + 0.1}$	(100-9.99)/(15+0.1)

System.out.println(9*15/5+32) dará como resultado 59, es decir: $\frac{9x15}{5} + 32$

Resto de la división entera: %

23 % 2	1
6 % 6	0
81 % 11	4

seno(x)	Math.sin(x)
x^2	Math.pow(x,2)
	Math.round(6.6) = 7
	Math.round(6.3) = 6

```
static final double kmporMilla = 1.609;
System.out.println(1000+" millas son "+(1000*kmporMilla)+" km.");
```

Dará por resultado: **1000 millas son 1609 km.**

Observa la forma de **concatenar**: y pueden ser números y textos!

- Escribe el siguiente programa:

```
// j015.java
/* Dadas dos tasas de interés, calcula la diferencia en
 el interés simple para el resto del año a partir de
 un mes dado */
public class j015 {
        static final double c=1000;
        static final int m=4; // ABRIL
        static final double tasaAntigua=12.5; // tanto por ciento
        static final double tasaNueva=13.00; // tanto por ciento
        public static void main(String [] args) {
                System.out.println("Cálculo del interés");
                System.out.println("====================");
                // Cálculos preliminares
                double cPrelim=c*(12-m)/12/100;
                // imprimir resultado
                System.out.println("Dado un cambio en la tasa de "+
                        "interés del "+tasaAntigua+"% al "+tasaNueva+
                        "% en el mes "+m+" ,");
                System.out.println("sobre un capital de "+c+
                        ", el interés para el resto del año");
                System.out.print("cambiará en: ");
                System.out.println(cPrelim*tasaNueva-cPrelim*tasaAntigua);
        }
}
```

- Grábalo con el nombre **j015.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Impresión de enteros y reales:

Número	Se imprime como	
10	10	
650	650	
14.75	14.75	
1.6213	1.6213	
1/1.6213	0.6167889964843027	
0.000000001	1.0E-9	
1.0/2	0.5	
1/2	0	Java, si divide dos enteros, automáticamente realiza la "división entera"

Cuando estudiemos el paquete "java.text" ya veremos formas de "controlar" la salida de números por pantalla.

Paquetes Java

Java es un lenguaje muy simple, pero suple su brevedad con muchos paquetes de clases ya escritas que podemos utilizar en nuestros programas.

Un paquete es una colección de clases que encajan lógicamente y que pueden interaccionar entre ellas.

Algunos de los paquetes "Java" más importantes:

- **lang:** funciones del lenguaje
- **util:** utilidades
- **io:** entrada, salida
- **text:** formateo especializado
- **awt:** gráficos e interfaz gráfica
- **awt.event:** gestión de eventos
- **applet:** programas para la "web"
- **net:** redes

Para conseguir acceso a cualquiera de los paquetes (excepto **lang**, que siempre está presente):

import.java.nombreDelPaquete.*;

El "asterisco" indica que todas las clases del paquete han de estar disponibles.

- Escribe el siguiente programa:

```
// j016.java
/* Programa para ver la diferencia entre una constante
   y una variable */
public class j016 {
        static final double pi=3.141592;
        public static void main(String [] args) {
                double radio=5;
                System.out.println("radio= "+radio);
                radio=45.34;
                System.out.println("Ahora el radio mide= "+radio);
                System.out.println("Longitud circunferencia= "+
                        2*pi*radio);
        }
}
```

- Grábalo con el nombre **j016.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- Observa:
  - Constante: no varía su valor durante el programa.
    Declarada antes del **main()**, al poner el modificador **static**
  - Variable: puede variar su valor durante la ejecución del programa.
    Declarada **en** el **main**: al no escribir ningún modificador.

Autoevaluación I

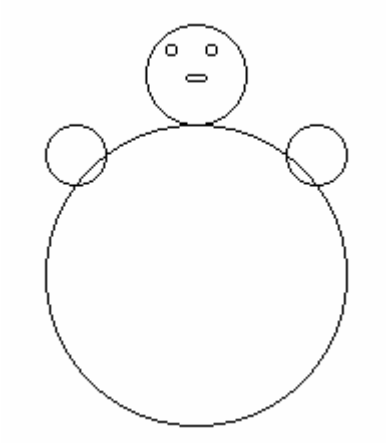
1) Escribe un programa de nombre **EVALJ1A** que dibuje un círculo azul en el centro de una ventana de 300x300 píxeles.

2) Escribe un programa de nombre **evalj1b**, que sirva para calcular las siguientes expresiones:
   52 % 100
   9000009
   Math.pow(4,3)
   21/7
   22/7
   4-3/4-2

3) La clase "Math" tiene un método "sqrt", que sirve para calcular raices cuadradas.
   Escribe un programa de nombre **evalj1c**, que sirva para resolver la ecuación: $x^2 - 10x + 16 = 0$

4) Un coche gasta 8 litros de gasolina cada 100 km en carreteras normales y un 15% más en carreteras desiguales. Escribe un programa de nombre **evalj1d**, que calcule la distáncia que el coche puede viajar con un depósito de 40 litros lleno en carreteras normales y cuál es la distancia que podría recorrer en carreteras desiguales.

5) Haz un programa de nombre **evalj1e**, que dibuje lo siguiente:



6) Haz un programa de nombre **evalj1f**, que calcule el área de un trapecio concreto, de forma que las bases del trapecio estén declaradas como constantes y la altura como una variable.

7) Haz un programa de nombre **evalj1g**, para calcular:
   - La división entera de 7 y 2
   - La división exacta entre 7 y 2

   Revisa el resultado del programa **evalj1d**

8) Crea un programa de nombre **evalj1h**, que calcule la hipotenusa del triángulo rectángulo de catetos 3.27 t 5.03 cm.

9) Crea un **Applet** de nombre **evalj1i**, que presente la palabra HOLA de color rojo y la palabra ADIOS de color verde. Crea una página web con el mismo nombre (evalj1i) para poder visualizar el applet.

10) Crea una nueva clase java de nombre **evalj1j**, que represente un contador, inicializado a 25 y que sirva para decrecer 3 unidades.

11) Crea un programa de nombre **evalj1k**, que utilice la clase **evalj1j** para mostrar el valor inicial y los dos siguientes valores del contador.

12) Crea un applet de nombre **evalj1l**, que haga lo mismo que el ejercicio anterior. Deberás crear una página web (con el mismo nombre), para comprobar el funcionamiento del applet.

13) ¿Cuál era el nombre primitivo del Java y de qué lenguaje de programación deriva?

14) ¿Qué diferencias hay entre un fichero binario en C++ o en Java?

15) ¿Porqué un programa Java necesita un intérprete?

16) ¿Quién es el "intérprete" en el caso particular de un applet de Java?

17) ¿Qué es un applet?. Relación entre un applet y un programa Java completo.

18) ¿Qué diferencia fundamental hay entre un applet de Java y un programa JavaScript?

19) ¿En qué consiste la Máquina Virtual Java?

20) ¿Qué relación hay entre un fichero con extensión **class** y un fichero con extensión **java**?

21) Si "pepe" es una clase Java, escribe las sentencias para crear tres objetos de "pepe" de nombres: pepito, pep y pepit.

22) Explica la diferencia entre **print** y **println**

23) Explica lo que sepas sobre un "constructor"

24) ¿Qué es el appletviewer?

25) ¿Qué diferencia hay en Java entre división entera y exacta?

Autoevaluación I (Soluciones)

1) Escribe un programa de nombre **EVALJ1A** que dibuje un círculo azul en el centro de una ventana de 300x300 píxeles.

```
// EVALJ1A.java
import java.awt.*;
import java.awt.event.*;
class evalj1a extends Frame {
        public evalj1a() {
                setTitle("EvalJ1A");
        }
        public static void main(String [] args) {
                Frame f=new evalj1a();
         f.addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
        f.setSize(300,300);
        f.setVisible(true);
        }
        public void paint(Graphics g){
                g.setColor(Color.blue);
                g.drawOval(0,0,300,300);
         g.drawOval(30,30,240,240);
        }
}
```

2) Escribe un programa de nombre **evalj1b**, que sirva para calcular las siguientes expresiones:

```
52 % 100
9000009
Math.pow(4,3)
21/7
22/7
4-3/4-2
```

```
// EVALJ1B
public class evalj1b {
        public static void main(String [] args) {
                System.out.println(52%100);
                System.out.println(9000009);
                System.out.println(Math.pow(4,3));
                System.out.println(21/7);
                System.out.println(22/7);
                System.out.println(4-3/4-2);
        }
}
```

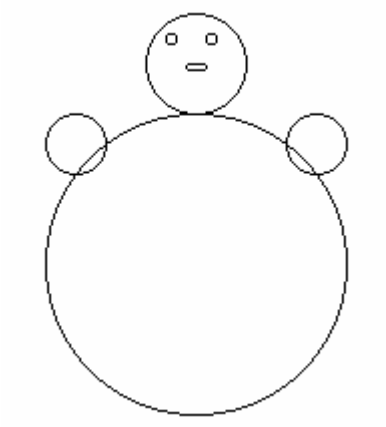
3) La clase "Math" tiene un método "sqrt", que sirve para calcular raices cuadradas.
Escribe un programa de nombre **evalj1c**, que sirva para resolver la ecuación: $x^2 - 10x + 16 = 0$

```
// EVALJ1C
public class evalj1c {
        static final double b=-10;
        static final double c=16;
        /* No sería necesario los "static final"
          ya que no tienen por qué ser constantes */
        public static void main(String [] args) {
                System.out.print("x1= ");
                System.out.println((-b+Math.sqrt(b*b-4*c))/2);
                System.out.print("x2= ");
                System.out.println((-b-Math.sqrt(b*b-4*c))/2);
        }
}
```

4) Un coche gasta 8 litros de gasolina cada 100 km en carreteras normales y un 15% más en carreteras desiguales. Escribe un programa de nombre **evalj1d**, que calcule la distáncia que el coche puede viajar con un depósito de 40 litros lleno en carreteras normales y cuál es la distancia que podría recorrer en carreteras desiguales.

```
// EVALJ1D.java
public class evalj1d {
        static final double kmlnormal=100.0/8.0;
        static final double kmldesig=100.0/(8+15*8.0/100.0);
        public static void main(String [] args) {
                System.out.print("Km/l carret. normal = ");
                System.out.println(kmlnormal);
                System.out.print("Km/l carret. desig. = ");
                System.out.println(kmldesig);
                System.out.print("Km con 40 l carret. normal = ");
                System.out.println(kmlnormal*40.0);
                System.out.print("Km con 40 l carret. desigual = ");
                System.out.println(kmldesig*40.0);
        }
}
```

5) Haz un programa de nombre **evalj1e**, que dibuje lo siguiente:



```
// EVALJ1E
import java.awt.*;
import java.awt.event.*;
class evalj1e extends Frame {
        public evalj1e() {
                setTitle("Dibujito");
        }
        public static void main(String [] args) {
                Frame f=new evalj1e();
         f.addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
        f.setSize(300,300);
        f.setVisible(true);
        }
        public void paint(Graphics g){
                g.drawOval(80,100,150,150);
                g.drawOval(80,100,30,30);
                g.drawOval(130,50,50,50);
                g.drawOval(200,100,30,30);
                g.drawOval(140,60,5,5);
                g.drawOval(160,60,5,5);
                g.drawOval(150,75,10,3);
        }
}
```

6) Haz un programa de nombre **evalj1f**, que calcule el área de un trapecio concreto, de forma que las bases del trapecio estén declaradas como constantes y la altura como una variable.

```
// EVALJ1F
public class evalj1f {
        static final double basMayor=15.7;
        static final double basMenor=7.81;
        public static void main(String [] args) {
                double altura=2.35;
                System.out.println("Trapecio= "+
               (basMayor+basMenor)*altura/2);
        }
}
```

7) Haz un programa de nombre **evalj1g**, para calcular:
   - La división entera de 7 y 2
   - La división exacta entre 7 y 2

   Revisa el resultado del programa **evalj1d**

```
// EVALJ1G
public class evalj1g {
        public static void main(String [] args) {
                System.out.println("División entera de 7 y 2= "+(7/2));
                System.out.println("División exacta de 7 y 2= "+(7.0/2.0));
        }
}
```

8) Crea un programa de nombre **evalj1h**, que calcule la hipotenusa del triángulo rectángulo de catetos 3.27 t 5.03 cm.

```
// EVALJ1H
public class evalj1h{
        public static void main(String [] args) {
                System.out.println("Hipotenusa= "+
                Math.sqrt(3.27*3.27+5.03*5.03));
        }
}
```

9) Crea un **Applet** de nombre **evalj1i**, que presente la palabra HOLA de color rojo y la palabra ADIOS de color verde. Crea una página web con el mismo nombre (evalj1i) para poder visualizar el applet.

```
// Applet EVALJ1I
import java.awt.*;
import java.applet.Applet;
public class evalj1i extends Applet {
        public void paint(Graphics g) {
                g.setColor(Color.red);
                g.drawString("Hola",25,25);
                g.setColor(Color.green);
                g.drawString("Adios",35,35);
        }
}
```

```
<html>
<applet code=evalj1i.class width=100 height=100>
</applet>
</html>
```

10) Crea una nueva clase java de nombre **evalj1j**, que represente un contador, inicializado a 25 y que sirva para decrecer 3 unidades.

```
//EVALJ1J
public class evalj1j {
        int x;
        public evalj1j() {
                x=25;
        }
        public int incCuenta() {
                x=x-3;
```

```
                return x;
        }
        public int getCuenta() {
                return x;
        }
}
```

11) Crea un programa de nombre **evalj1k**, que utilice la clase **evalj1j** para mostrar el valor inicial y los dos siguientes valores del contador.

```
// evalj1k.java= programa que utiliza la clase evalj1j
import java.io.*;
public class evalj1k {
        static int n;
        static evalj1j cuenta;
        public static void main(String [] args) {
                System.out.println("Cuenta...");
                cuenta=new evalj1j();
                System.out.println(cuenta.getCuenta());
                n=cuenta.incCuenta();
                System.out.println(n);
                cuenta.incCuenta();
                System.out.println(cuenta.getCuenta());
        }
}
```

12) Crea un applet de nombre **evalj1l**, que haga lo mismo que el ejercicio anterior. Deberás crear una página web (con el mismo nombre), para comprobar el funcionamiento del applet.

```
// Applet evalj1l
import java.applet.*;
import java.awt.*;
public class evalj1l extends Applet {
  static int n;
  static evalj1j laCuenta;
  public evalj1l() {
        laCuenta= new evalj1j();
  }
  public void paint(Graphics g) {
        g.drawString("Cuenta...",20,20);
       g.drawString(String.valueOf(laCuenta.getCuenta()),20,35);
        n=laCuenta.incCuenta();
        g.drawString(String.valueOf(n),20,50);
        laCuenta.incCuenta();
       g.drawString(String.valueOf(laCuenta.getCuenta()),20,65);
  }
 }
```

```
<html>
<head>
<title>evalj1l.htm</title>
</head>
<body>
<applet code="evalj1l.class" width=170 height=150>
</applet>
</body>
```

</html>

13) ¿Cuál era el nombre primitivo del Java y de qué lenguaje de programación deriva?

Oak. C++

14) ¿Qué diferencias hay entre un fichero binario en C++ o en Java?

En C++ es directamente ejecutable (extensión EXE). En Java (extensión CLASS) es bytecode, esto es: necesita un intérprete para ejecutarse.

15) ¿Porqué un programa Java necesita un intérprete?

Porque el fichero binario correspondiente es bytecode: es decir no ejecutable.

16) ¿Quién es el "intérprete" en el caso particular de un applet de Java?

Un navegador de Internet

17) ¿Qué es un applet?. Relación entre un applet y un programa Java completo.

Una aplicación java que se ejecuta en una página web. En un applet, el intérprete es el navegador. En un programa java, el intérprete depende de la máquina.

18) ¿Qué diferencia fundamental hay entre un applet de Java y un programa JavaScript?

El intérprete correspondiente a un applet es precisamente el navegador de Internet

19) ¿En qué consiste la Máquina Virtual Java?

Es el programa que forma parte de un navegador de Internet que permite "interpretar", por lo tanto "ejecutar" un applet.

20) ¿Qué relación hay entre un fichero con extensión **class** y un fichero con extensión **java**?

Fichero.java: es el archivo que escribimos nosotros utilizando el lenguaje de programación java
Fichero.class: es el fichero anterior, pero compilado (binario = bytecode)

21) Si "pepe" es una clase Java, escribe las sentencias para crear tres objetos de "pepe" de nombres: pepito, pep y pepit.

pepe pepito=new pepe(); **pepe pep=new pepe();** **pepe pepit=new pepe();**

22) Explica la diferencia entre **print** y **println**

Println = print + cambio de línea

23) Explica lo que sepas sobre un "constructor"

Es una función especial de una clase (mismo nombre de la clase), cuya utilidad es inicializar variables.

24) ¿Qué es el appletviewer?

Es es visualizador de applets que incorpora el JDK

25) ¿Qué diferencia hay en Java entre división entera y exacta?

La división entre enteros, en java, nos da la división entera (cociente de la división sin decimales).
La división exacta en java, es la división entre números no enteros.

II.- Entrada y Salida de Datos

Entrada de datos por el teclado

Teclado = flujo "in"

1) Necesitamos el paquete **java.io**, es decir:
   Primera línea del programa: **import java.io.*;**
2) Establecer el flujo de entrada, el teclado (podría ser un fichero u otro dispositivo de entrada):
   Primera línea del **main**:
   BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
3) Si en el proceso de entrada de datos hay algo que va mal, el programa puede hacer cosas incontroladas, para evitarlo se usa una **excepción**, de la siguiente forma:
   public static void main(String[] args) **throws IOException** {

Leyendo cadenas de caracteres

String nombreVariable = in.readLine();

La sentencia anterior, inmoviliza la ejecución del programa hasta que escribamos algo por teclado y pulsemos [Return].
Lo que escribimos se guarda en la variable de texto: **nombreVariable**

- Escribe el siguiente programa java:

```
// j017.java
import java.io.*;
public class j017 {
        public static void main(String [] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Escribe tu nombre: ");
                String nom=in.readLine();
                System.out.println("Adios "+nom);
        }
}
```

- Grábalo con el nombre **j017.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Leyendo números

Java puede leer fácilmente números y otros tipos de datos si están en forma binaria y en un archivo. Pero en el caso de números entrados por teclado, debe leer una cadena y hacer **la conversión** en el programa.
Las funciones de conversión se encuentran en el paquete **java.text**, por lo tanto:
1ª línea de programa: **import java.text.*;**
Para leer un número real, deberemos escribir:
double x=Double.valueOf(in.readLine().trim()).doubleValue();

- Escribe el siguiente programa:

```
// j018.java
import java.text.*;
import java.io.*;
public class j018 {
        public static void main(String [] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Escribe un número: ");
                double x=Double.valueOf(in.readLine().trim()).doubleValue();
                System.out.println("El triple es = "+3*x);
        }
}
```

- Grábalo con el nombre **j018.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Booleanos

Las condiciones controlan las decisiones que se toman en los programas en cuanto a caminos alternativos a seguir. Una condición produce un valor **true** o **false**.

Las expresiones booleanas usan los siguientes operadores de comparación:

== igualdad	**& y**
!= distinto	**| o**
> mayor	**^ o exclusivo**
< menor	**! No**
>= mayor o igual	
<= menor o igual	

Java también tiene una versión adicional de “y” y “o” que son los operadores “&&” y “||”

Caracteres

Los caracteres en Java se escriben con una única comilla, por ejemplo: ‘a’

Caracteres de escape:

\b	retroceso
\t	tabulador
\n	cambio de línea
\f	cambio de página
\r	volver al principio de la línea.
\u	permite accerder a los 10.000 caracteres Unicode.

Tipos Numéricos

		Valor máximo
byte	entero con signo	127
short	“	32.767
int	“	2.147.483.647
long	“	
float	punto flotante	
double	“	

i++ equivale a i=i+1 y equivale a i+=1

total += 5 será equivalente a "total=total+5"

Conversión entre tipos

Conversión automática:
byte → short → int → long → float → double

Math.round(un número "double") dará por resultado un número "long".

```
float kgr
double x
kgr=(float)(x*1.2)
```

(int) 6.3 dará como resultado 6

A diferencia de otros lenguajes, no hay en Java posibilidad para convertir caracteres y booleanos a números y viceversa.

Bucle "for"

```
for (int i=0;i<5;i++) {
        ----------;
        ----------;
}
```
En el caso particular de una única sentencia, no es necesario encerrarla entre llaves.

- Escribe el siguiente programa:

```
// j019.java
import java.text.*;
import java.io.*;
public class j019 {
        public static void main(String [] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Número de etiquetas: ");
                double etiq=Double.valueOf(in.readLine().trim()).doubleValue();
                System.out.println();
                for (int i=0;i<etiq;i++) {
                        System.out.println("================");
                        System.out.println("|   H O L A   |");
                        System.out.println("================");
                        System.out.println("\n\n");
                }
        }
}
```

- Grábalo con el nombre **j019.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Programa que dibuja un cuadrado de asteriscos:

- Escribe el siguiente programa:

```
// j020.java
import java.text.*;
import java.io.*;
public class j020 {
        public static void main(String [] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Lado: ");
                double x=Double.valueOf(in.readLine().trim()).doubleValue();
                System.out.println();
                for (int i=0;i<x;i++) {
                        System.out.print((i+1));
                        for (int j=0;j<x;j++)
                                System.out.print("*");
                        System.out.println();
                }
        }
}
```

- Grábalo con el nombre **j020.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- Observa que escribimos **i+1** entre paréntesis en **System.out.print((i+1))**
  Ya que en caso contrario nos escribiría: 01
  11
  21
  …

- Escribe el siguiente programa:

```
// j021.java
public class j021 {
/* Tabla de temperaturas Celsius Farenheit */
        public static void main(String [] args) {
                System.out.println("Tabla de conversion de Temperaturas");
                System.out.println("====================================");
                System.out.println();
                System.out.println("C\tF");
                for (int c=5;c<=20;c++) {
                        System.out.print(c+"\t");
                        System.out.println(Math.round(c*9/5+32));
                }
        }
}
```

- Grábalo con el nombre **j021.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Construyendo Métodos

Un método es un grupo de declaraciones e instrucciones a las que se da un nombre y que puede llamarse por este nombre para llevar a cabo una acción concreta.

El modelo es:

```
modificadores  grupo nombre(parámetro) {
        intrucción1;
        instrucción2;
        return expresión;  // sólo métodos con resultado
}
```

Ejemplos:

1)
```
static void caja() {
        System.out.println("----------------------------");
        System.out.println("|                          |");
        System.out.println("|                          |");
        System.out.println("|                          |");
        System.out.println("----------------------------");
}
```

2)
```
static int Fahrenheit(int Celsius) {
        return Celsius*9/5+32;
}
```

- Si un método se declara como **void**, se le llama dando su nombre y los parámetros (si hay alguno).

  Ejemplo 1: **caja();**
  cuyo efecto será imprimir:
  ```
  ----------------------------
  |                          |
  |                          |
  |                          |
  ----------------------------
  ```

- System.out.println(30+"C es "+Fahrenheit(30)+"F");
  Imprimirá:
  30C es 86F

Los métodos pueden hacerse más potentes permitiendo que el efecto sea distinto cada vez que llamemos al método. Esto se llama generalizar o parametrizar un método y los valores que van a ser diferentes se conocen como parámetros.

Lo que queremos es una forma de poder escribir:

caja(12,16)

Siendo 12 el ancho y 16 el alto.
Para ello deberíamos declarar: **static void caja(int ancho,int alto)**

Entonces:
```
int tamaño=60;
caja(tamaño,(int)tamaño/2);
```
No sería correcto: caja("pepe") o caja(12.5,10);

- Escribe el siguiente programa:

```
// j022.java
public class j022 {
        static final int colsPorLinea=5;
        static final int maxNoLinea=10;
        static final String hueco="\t";
        public static void main(String[]args) {
                System.out.println("\tTabla conversion temperaturas");
                System.out.println("\t============================");
                System.out.println();
                for(int col=0;col<colsPorLinea;col++)
                        System.out.print("C   F"+hueco);
                System.out.println();
                for(int linea=0;linea<maxNoLinea;linea++)
                        impLinea(linea);
        }
        static void impLinea(int estalinea) {
                for(int col=0;col<colsPorLinea;col++) {
                        int c=estalinea*colsPorLinea+col;
                        System.out.print(c+"  ");
                        System.out.print(fahrenheit(c)+hueco);
                }
                System.out.println();
        }
        static int fahrenheit(int Celsius) {
                return Math.round(Celsius*9/5+32);
        }
}
```

- Grábalo con el nombre **j022.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- Escribe el siguiente programa:

```
// j023.java
/* Muestra un histograma del interes simple de 5 a 15 anios */
public class j023 {
  static final double c=1000;
  static final double tasa=12.5;
  public static void main(String[] args) {
        System.out.println("Plan Crecimiento PepesBanc");
        System.out.println("=========================");
        System.out.println("Capital de "+c+" a una tasa del "+tasa);
        System.out.println();
        System.out.println("Anios");
        for(int anio=5;anio<=15;anio++)
                barra(anio,c*1.2*anio*tasa/100+c);
        eje();
  }
  static void barra(int etiqueta,double h) {
        System.out.print(etiqueta+"\t");
        int stop=(int)(h/100);
        for(int estrella=0;estrella<stop;estrella++)
                System.out.print('*');
```

```
                System.out.println(" "+h);
        }
        static void eje() {
                int ticks=5;
                System.out.print('\t');
                for(int barra=0;barra<ticks*10;barra++)
                        System.out.print('=');
                System.out.println('=');
                System.out.print('\t');
                for(int n=0;n<ticks;n++)
                        System.out.print("+\t");
                System.out.println('+');
                System.out.print('\t');
                for(int n=0;n<=ticks;n++)
                        System.out.print(n*1000+"\t");
                System.out.println();
                System.out.println("\t\t\t\tpts");
        }
}
```

- Grábalo con el nombre **j023.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Devolviendo valores desde un método

Los valores pueden pasarse a los métodos vía parámetros, pero no sacarse de ellos. Para obtener un valor de un método, usamos el proceso "return".

Pero ¿qué ocurre si queremos devolver más de un valor?: mediante un objeto "x" de la clase "Obj" declarada fuera del método de la siguiente forma:

```
class Obj {
        int a;
        double b;
}
Obj x=new Obj();
```

1ª posibilidad:

```
static void m1(Obj pepe) {
        pepe.a=valor1;
        pepe.b=valor2;
}
m1(x); //llamada
```

"x" pasa a ser "pepe" dentro de "m1", es decir los cambios a "pepe" son realmente cambios a "x".

2ª posibilidad:

```
static void m2() {
    x.a=valor1;
    x.b=valor2;
}
m2(); // llamada
```

Hay otras posibilidades pero son menos comunes.

- Vamos a crear una clase que nos permita imprimir etiquetas.

Escribe el siguiente programa:

```
// j024.java
public class j024 {
    public j024(int a) {
            atraves=a;
    }
    public void impHorizontal() {
            for(int caja=0;caja<atraves;caja++)
                    System.out.print("----------\t");
            System.out.println();
    }
    void impVertical() {
            for(int caja=0;caja<atraves;caja++)
                    System.out.print("|         |\t");
            System.out.println();
    }
    private int atraves;
}
```

- Grábalo con el nombre **j024.java** en *TuCarpeta*

- Compílalo pero no lo ejecutes.

- Observa que el parámetro del **constructor** lo copiamos en una variable **private**, que pueden usar los métodos de la clase, pero es inaccesible desde fuera de la clase.

- Vamos a hacer un programa que utilice la clase anterior:

Escribe:

```
// j025.java
public class j025 {
        static final int nEtiq=4;
        static final int nfilas=4;
        public static void main(String[] args) {
                j024 etiqueta=new j024(nEtiq);
                etiqueta.impHorizontal();
                for(int fila=0;fila<nfilas;fila++)
                        etiqueta.impVertical();
                etiqueta.impHorizontal();
        }
}
```

- Grábalo con el nombre **j025.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- Vamos a hacer otra clase para imprimir otro tipo de etiquetas:

Escribe:

```
// j026.java
class j026 {
    private char hori,verti,precio;
    private int altura,anchura;
    j026(char h,char v,int l,int a) {
            hori=h;
            verti=v;
            altura=l;
            anchura=a;
    }
    void cajaVariable(char precio) {
            unaLinea(hori,hori,hori);
            for(int l=2;l<altura;l++)
                    unaLinea(verti,precio,verti);
            unaLinea(hori,hori,hori);
            System.out.println();
    }
    void unaLinea(char izquierda,char centro,char derecha) {
            System.out.print(izquierda);
            for(int a=2;a<anchura;a++)
                    System.out.print(centro);
            System.out.println(derecha);
    }
}
```

- Grábalo con el nombre **j026.java** en *TuCarpeta*

- Compílalo y no lo ejecutes.

- Vamos a hacer un programa que utilice la clase anterior, escribe:

```
// j027.java
public class j027 {
    public static void main(String[] args) {
            System.out.println("Vales para la fiesta");
            j026 pequenio=new j026('-','|',5,11);
            pequenio.cajaVariable('2');
            pequenio.cajaVariable('1');
            j026 grande=new j026('=','=',7,15);
            grande.cajaVariable('5');
            grande.cajaVariable('8');
    }
}
```

- Grábalo con el nombre **j027.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- Programa que nos pregunta nuestro nombre y edad y da como resultado los días de vida que tenemos

Escribe:

```
// j028.java
import java.io.*;
import java.text.*;
public class j028 {
```

```
        public static void main(String[] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Escribe tu nombre: ");
                String nom=in.readLine();
                System.out.println();
                System.out.print("Escribe tu edad: ");
                double edad=Double.valueOf(in.readLine().trim()).doubleValue();
                System.out.println("\n\n");
                System.out.println("En estos momentos tienes "+(365*edad)+" dias de vida");
        }
}
```

- Grábalo con el nombre **j028.java** en *TuCarpeta*.

- Compílalo y ejecútalo.

- Programa que escribe los múltiplos de 13 menores de 1000

Escribe:

```
// j029.java
public class j029 {
        public static void main(String[] args) {
                for(int mult=13;mult<1000;mult=mult+13)
                        System.out.print(mult+"\t");
        }
}
```

- Grábalo con el nombre **j029.java** en *TuCarpeta*.

- Compílalo y ejecútalo.

- Programa que escribe los múltiplos de 11 menores de 1500 y escribe la suma y producto de todos ellos.

Escribe:

```
// j030.java
public class j030 {
        public static void main(String[] args) {
                double sum,pro;
                sum=0;pro=1;
                for(double mult=11;mult<1500;mult=mult+11) {
                        System.out.print(mult+"\t");
                        sum=sum+mult;
                        pro=pro*mult;
                }
                System.out.println("\n\nSuma= "+sum);
                System.out.println("\nProducto= "+pro);
        }
}
```

- Grábalo con el nombre **j030.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- Supongo que está claro lo que significa **Producto = Infinity**

Vamos a crear una clase especial que utilizaremos a partir de este momento en las "entradas por teclado" (habrás observado que es horrible la forma de programar una entrada en Java).

- Escribe:

```
// Text.java

import java.io.*;
import java.util.*;
import java.text.*;

public class Text {

 public Text () {};

 private static StringTokenizer T;
 private static String S;

 public static BufferedReader open (InputStream in)  {
  return new BufferedReader(new InputStreamReader(in));
 }

 public static BufferedReader open (String filename)
    throws FileNotFoundException {
  return new BufferedReader (new FileReader (filename));
 }

 public static PrintWriter create
     (String filename) throws IOException {
  return new PrintWriter (new FileWriter (filename));
 }

 public static void prompt (String s) {
  System.out.print(s + " ");
  System.out.flush();
 }

 public static int readInt (BufferedReader in) throws IOException {
    if (T==null) refresh(in);
    while (true) {
      try {
        return Integer.parseInt(T.nextToken());
      }
      catch (NoSuchElementException e1) {
        refresh (in);
      }
      catch (NumberFormatException e2) {
        System.out.println("Error in number, try again.");
      }
    }
  }

 public static char readChar (BufferedReader in) throws IOException {
```

```
    if (T==null) refresh(in);
    while (true) {
      try {
        return T.nextToken().trim().charAt(0);
      }
      catch (NoSuchElementException e1) {
        refresh (in);
      }
    }
  }

public static double readDouble (BufferedReader in) throws IOException {
    if (T==null) refresh(in);
    while (true) {
      try {
        String item = T.nextToken();
        return Double.valueOf(item.trim()).doubleValue();
      }
      catch (NoSuchElementException e1) {
        refresh (in);
      }
      catch (NumberFormatException e2) {
        System.out.println("Error in number, try again.");
      }
    }
  }

 public static String readString (BufferedReader in) throws IOException {
   if (T==null) refresh (in);
   while (true) {
     try {
       return T.nextToken();
     }
     catch (NoSuchElementException e1) {
       refresh (in);
     }
   }
 }

 private static void refresh (BufferedReader in) throws IOException {
   S = in.readLine ();
   if (S==null) throw new EOFException();
   T = new StringTokenizer (S);
 }

 //  Metodos Escritura
 //  -----------------

 private static DecimalFormat N = new DecimalFormat();
 private static final String spaces = "                  ";

 public static String writeDouble (double number, int align, int frac) {
   N.setGroupingUsed(false);
   N.setMaximumFractionDigits(frac);
   N.setMinimumFractionDigits(frac);
   String num = N.format(number);
   if (num.length() < align)
     num = spaces.substring(0,align-num.length()) + num;
   return num;
 }
```

```
  public static String writeInt (int number, int align) {
    N.setGroupingUsed(false);
    N.setMaximumFractionDigits(0);
    String num = N.format(number);
    if (num.length() < align)
      num = spaces.substring(0,align-num.length()) + num;
    return num;

   }
  }
```

- Graba el fichero con el nombre **Text.java** en *TuCarpeta*

- Compílalo, pero no lo ejecutes.

Como puedes observar los métodos de la clase **Text** son:

void prompt(String s)
int readInt(BufferedReader in)
double readDouble(BufferedReader in)
String readString(BufferedReader in)
char readChar(BufferedReader in)
String writeInt(int number,int align)
String writeDouble(double number,int align,int frad)
BufferedReader open(InputStream in)
BufferedReader open(String filename)
printWriter creater(String filename)

prompt es un método que imprimirá una cadena, pero mantiene el cursor en la misma línea para que si hay una respuesta, ésta pueda aparecer a continuación.

readInt y **readDouble** devuelven números

readChar lee un único carácter.

El parámetro **align** especifica el número mínimo de caracteres
Ejemplo: **align=6** el número 123 tendrá 3 espacios delante

writeInt y **writeDouble** tienen la propiedad de que si el número no cabe en el espacio dado, éste se expandirá hacia la derecha, y se imprimirán por completo los dígitos antes del punto decimal.

Para números reales es necesario el parámetro **frac**:

Por ejemplo:
System.out.println(writeDouble(x),10,4);
Tendremos para diferentes valores de "x":

X	**en pantalla**
-1234.5678	-1234.5678
1234.56789	1234.5678
45.67	45.6700
4	4.0000
4.56789	4.5678
0	0.0000
123456789	123456789.0000
777777.88888	777777.8888

- Vamos a comenzar a utilizar la clase **Text**...

Escribe:

```
// j031.java
import java.io.*;

class j031 {

 /* Lee numeros interactivamente y muestra su suma
    utilizando la nueva clase Text
    No confundir la nueva clase Text, con mayusculas
    con el paquete incorporado java "text" en
    minuscula.   */

 public static void main(String[] args) throws IOException {

  int count;
  double total = 0;
  double number;

  BufferedReader in = Text.open(System.in);

  System.out.println("****** Suma de Numeros ******");

  Text.prompt("Cuantos numeros?");
  count = Text.readInt(in);

  for (int i = 1; i <= count; i++) {
   System.out.print(i+"> ");
   number = Text.readDouble(in);
   total += number;
  }
  System.out.println("Ya es suficiente, gracias.");
  System.out.println("La suma es "+total);
 }
}
```

- Grábalo con el nombre **j031.java** en *TuCarpeta*

- Compílalo y ejecútalo.

La clase **Text** si detecta números erróneos, nos permitirá escribirlos de nuevo. Permite (ignora) líneas en blanco y espacios entre los datos: pero no los permite dentro de las cadenas...

- Escribe el siguiente programa:

```
// j032.java
import java.io.*;
class j032 {
        public static void main(String[] args) throws IOException {
                BufferedReader in= Text.open(System.in);
                Text.prompt("¿Cómo se llama ?");
                String nombre=Text.readString(in);
                System.out.print("Felicidades "+nombre);
/* Al escribir tu nombre, escribe también tu apellido
```

```
        observaras que no aparece el apellido, debido al espacio
        en blanco que escribes entre el nombre y el apellido */
            }
        }
```

- Grábalo con el nombre **j032.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Para solucionar el problema anterior, escribe el siguiente programa:

```
// j033.java
import java.io.*;
class j033 {
    public static void main(String[] args) throws IOException {
        BufferedReader in= Text.open(System.in);
        Text.prompt("¿Cómo se llama ?");
        String nombre=Text.readString(in);
        String apellido=Text.readString(in);
        System.out.print("Felicidades "+nombre+" "+apellido);
/* Al escribir tu nombre, escribe también tu apellido */
    }
}
```

- Grábalo con el nombre **j033.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Vamos a ver cómo funciona la entrada de datos en un programa, no por teclado sino por un archivo.

- Escribe el siguiente fichero:

```
35
4
17.3
5
1.2
73
```

- Grábalo con el nombre **j034.txt** en *TuCarpeta*.

- Escribe el siguiente programa:

```
// j035.java
import java.io.*;

class j035 {

  public static void main (String [] args) throws IOException {

    int count;
    double total = 0;
    double number;
```

```
    BufferedReader in = Text.open(System.in);
    BufferedReader fin = Text.open("j034.txt");

    System.out.println("****** Suma de numeros desde un fichero ******");

    Text.prompt ("Cuantos numeros hay?");
    count = Text.readInt(in);

    for (int i = 1; i <= count; i++) {
      number = Text.readDouble(fin);
      total += number;
    }
    System.out.println("Ya es suficiente, gracias.");
    System.out.println("El total es "+total);
  }
}
```

- Grábalo con el nombre **j035.java** en *TuCarpeta*

- Compílalo y ejecútalo.

- De la misma forma que podemos entrar los datos desde un archivo, también podemos enviar la salida de un programa a un fichero:

Escribe el siguiente programa:

```
// j036.java
import java.io.*;
public class j036 {
    public static void main(String[] args) throws IOException {
            PrintWriter fout=Text.create("j036.txt");
            System.out.println("Imprimiendo la etiqueta en j036.txt");
            fout.println("-----------------------");
            fout.println("|                     |");
            fout.println("|      pepe           |");
            fout.println("|                     |");
            fout.println("-----------------------");
            fout.close();
            System.out.println("Programa terminado");
    }
}
```

- Grábalo con el nombre **j036.java** en *TuCarpeta*

- Compílalo y ejecútalo.

La estructura de programación IF-ELSE

Modelo:

```
if (condición)
        instrucción;
else instrucción;
```

En general:

```
if(condición) {
        ..............;
        ..............;
}
else {
        ..............;
        ..............;
}
```

- Escribe el siguiente programa:

```
// j037.java
import java.io.*;

class j037 {
 public static void main(String[] args) throws IOException {

  BufferedReader in = Text.open(System.in);

  int count;
  double total = 0;
  double posTotal = 0;
  double negTotal = 0;
  double number;

  System.out.println("****** SUMAR ******");
  Text.prompt("¿Cuántos números?");
  count = Text.readInt(in);

  for (int i = 1; i <= count; i++) {
   Text.prompt(i+">");
   number = Text.readDouble(in);
   total = total + number;
   if (number > 0)
     posTotal += number;
   else
     negTotal += number;
   }

  System.out.println("Esto es todo,gracias.");
  System.out.println("La suma total es "+total);
  System.out.println("La suma de los positivos es "+posTotal);
  System.out.println("La suma de los negativos "+negTotal);
 }

}
```

- Grábalo en *TuCarpeta* con el nombre **j037.java**

- Compílalo y ejecútalo.

- Escribe el siguiente programa:

```
// j038.java
import java.io.*;
```

```
class j038 {

  public static void main(String[] args) throws IOException {

    BufferedReader in = Text.open(System.in);

    System.out.println("*****  Localizo el numero mas alto *****");

    Text.prompt("Cuantos numeros?");
    int n = Text.readInt(in);

    System.out.println("Escribelos:");
    Text.prompt("1>");
    int highest = Text.readInt(in);

    int number;
    for (int i = 2; i <= n; i++) {
      Text.prompt(i+">");
      number = Text.readInt(in);
      if (number > highest)
        highest = number;
    }
    System.out.println("Esto es todo, gracias");
    System.out.println("El numero mas alto es "+highest);
  }

}
```

- Grábalo con el nombre **j038.java**

- Compílalo y ejecútalo

Introducción a las excepciones

Una excepción es un objeto que avisa que ha ocurrido alguna condición inusual. Java tiene muchos objetos de excepción predefinidos, y también podemos crear nuestros propios.

Una de las excepciones más utiles es la que avisa del final de los datos. En el teclado, el usuario teclea datos y cuando termina, pulsa el carácter que el sistema usa para el final del flujo. Este puede ser [CTRL][Z] o [ESC] por ejemplo. La presencia de este carácter especial es detectada por la clase **Text**, que la pasa al método lanzando la **IOException**

Veamoslo con un ejemplo:

- Escribe el siguiente programa:

```
// j039.java
import java.io.*;

class j039 {

  public static void main (String [] args) throws IOException {

    int count = 0;
```

```
    double total = 0;
    double number;

    BufferedReader in = Text.open(System.in);

    System.out.println("****** Suma de n numeros ******");
    System.out.println("Escribe los números, para acabar pulsa control-D (unix)"+
      " o control-Z (Windows)");

    try {
     for (count = 1; ; count++) {
       Text.prompt(count+">");
       number = Text.readDouble(in);
       total += number;
     }
    } catch (EOFException e) {
       System.out.println("Esto es todo, gracias.");
       System.out.println("La suma total es "+total);
    }
  }
}
```

- Grábalo en *TuCarpeta* con el nombre **j039.java**

- Compílalo y ejecútalo.

Autoevaluación II

1) Haz un programa de nombre **evalj2a**, que funcione de la siguiente forma:

- El programa nos pregunta nuestro nombre
- El programa escribe 15 veces "Hola" y a continuación el nombre introducido

Haz el programa sin utilizar la clase **Text**.

2) Haz un programa de nombre **evalj2b**, que funcione igual que el anterior pero utilizando la clase **Text**.

3) Haz un programa de nombre **evalj2c**, que funcione de la siguiente forma:

- El programa nos pide un número
- El programa nos escribe el doble, triple, cuadrúple y quintuple del número introducido (utiliza un "for").

Haz el programa sin utilizar la clase **Text**.

4) Haz un programa de nombre **evalj2d**, que haga lo mismo que el anterior, pero utiliza la clase **Text**.

5) Haz un programa de nombre **evalj2e**, que dibuje un rectángulo de asteriscos a partir de la base y la altura.

6) Haz un programa de nombre **evalj2f**, que calcule los múltiplos de 7 menores de 100, su número, su suma y producto de todos ellos.

7) Haz un programa de nombre **evalj2g**, que escriba la tabla de valores de la función $y=3x^2-5x+1$ entre dos valores que hemos de entrar por teclado y un incremento de la "x", que también hemos de entrar por teclado.

   Utiliza la clase **Text**

8) Haz un programa de nombre **evalj2h**, que sirva para calcular la hipotenusa de un triangulo rectángulo a partir de la medida de los dos catetos.

Autoevaluación II (Soluciones)

1) Haz un programa de nombre **evalj2a**, que funcione de la siguiente forma:

- El programa nos pregunta nuestro nombre
- El programa escribe 15 veces "Hola" y a continuación el nombre introducido

Haz el programa sin utilizar la clase **Text**.

```
// evalj2a.java
import java.io.*;
public class evalj2a {
        public static void main(String[] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Escribe tu nombre: ");
                String nom=in.readLine();
                for(int i=0;i<15;i++)
                        System.out.println("Hola "+nom);
        }
}
```

2) Haz un programa de nombre **evalj2b**, que funcione igual que el anterior pero utilizando la clase **Text**.

```
// evalj2b.java
import java.io.*;
public class evalj2b {
        public static void main(String[] args) throws IOException {
                BufferedReader in=Text.open(System.in);
                Text.prompt("Escribe tu nombre: ");
                String nom=Text.readString(in);
                for(int i=0;i<15;i++)
                        System.out.println("Hola "+nom);
        }
}
```

3) Haz un programa de nombre **evalj2c**, que funcione de la siguiente forma:

- El programa nos pide un número
- El programa nos escribe el doble, triple, cuadrúple y quintuple del número introducido (utiliza un "for").

Haz el programa sin utilizar la clase **Text**.

```
// evalj2c.java
import java.io.*;
import java.text.*;
public class evalj2c {
        public static void main(String[] args) throws IOException {
                BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Escribe un numero: ");
                double x=Double.valueOf(in.readLine().trim()).doubleValue();
                for(int j=2;j<6;j++)
                        System.out.println(j*x);
```

```
	}
}
```

4) Haz un programa de nombre **evalj2d**, que haga lo mismo que el anterior, pero utiliza la clase **Text**.

```
// evalj2d.java
import java.io.*;
public class evalj2d {
	public static void main(String[] args) throws IOException {
		BufferedReader in=Text.open(System.in);
		System.out.print("Escribe un numero: ");
		double x=Text.readDouble(in);
		for(int i=2;i<6;i++)
			System.out.println(i*x);
	}
}
```

5) Haz un programa de nombre **evalj2e**, que dibuje un rectángulo de asteriscos a partir de la base y la altura.

```
// evalj2e.java
import java.io.*;
public class evalj2e {
	public static void main(String[] args) throws IOException {
		BufferedReader in=Text.open(System.in);
		System.out.print("Escribe el numero de asteriscos de la base: ");
		int bas=Text.readInt(in);
		System.out.println();
		System.out.print("Escribe el numero de asteriscos de la altura: ");
		int alt=Text.readInt(in);
		System.out.println();
		for(int i=0;i<alt;i++) {
			for(int j=0;j<bas;j++)
				System.out.print("*");
			System.out.println();
		}
	}
}
```

6) Haz un programa de nombre **evalj2f**, que calcule los múltiplos de 7 menores de 100, su número, su suma y producto de todos ellos.

```
// evalj2f.java
public class evalj2f {
	public static void main(String[] args) {
		double sum,pro,n;
		sum=0;pro=1;n=0;
		for(double mult=7;mult<1000;mult=mult+7) {
			n++;
			System.out.print(mult+"-");
			sum=sum+mult;
			pro=pro*mult;
```

```
                }
                System.out.println();
                System.out.println("Número de multiplos= "+n);
                System.out.println("\n\nSuma= "+sum);
                System.out.println("\nProducto= "+pro);
        }
}
```

7) Haz un programa de nombre **evalj2g**, que escriba la tabla de valores de la función $y=3x^2-5x+1$ entre dos valores que hemos de entrar por teclado y un incremento de la "x", que también hemos de entrar por teclado.

Utiliza la clase **Text**

```
// eval2g.java
import java.io.*;
class evalj2g {
    public static void main(String[] args) throws IOException {
            BufferedReader in=Text.open(System.in);
            System.out.println("Valor mínimo de x= ");
            double x1=Text.readDouble(in);
            System.out.println("Valor maximo de x= ");
            double x2=Text.readDouble(in);
            System.out.println("Incremento de la x= ");
            double incre=Text.readDouble(in);
            for(double x=x1;x<=x2;x=x+incre)
                    System.out.println("x= "+x+"\ty= "+(3*x*x-5*x+1)+"\n");
    }
}
```

8) Haz un programa de nombre **evalj2h**, que sirva para calcular la hipotenusa de un triangulo rectángulo a partir de la medida de los dos catetos.

```
// eval2h.java
import java.io.*;
class evalj2h {
        public static void main(String[] args) throws IOException {
                BufferedReader in=Text.open(System.in);
                System.out.println("Cateto= ");
                double cat1=Text.readDouble(in);
                System.out.println("El otro cateto= ");
                double cat2=Text.readDouble(in);

                double hipot=Math.sqrt(cat1*cat1+cat2*cat2);
                System.out.println("Hipotenusa= "+hipot);
        }
}
```

III.- Estructura del Lenguaje

Definición de las Reglas Sintácticas

Las instrucciones, expresiones y bloques

Una **instrucción** es un comando cualquiera cuyo final está marcado por un punto y coma. Por tanto es posible escribir una instrucción en varias líneas.

Ejemplos:

```
import java.applet.Applet;
System.out.println(“pepe”);
```

Ciertas instrucciones tienen el efecto de producir un valor, hablando en este caso de **expresiones**. El valor producido se llama valor de retorno.

Ejemplos:

```
int x=(y+z)/(z+y);
int x=10;
Obj.x=10;
```

Un **bloque** de instrucciones, permite realizar un conjunto de instrucciones. Por ello, es necesario declarar el bloque de instrucciones entre llaves.

Ejemplo:
Creación de un **thread** (proceso ligero) para hacer una pausa de 2 segundos en la ejecución de un programa:

```
try {
        Thread.sleep(2000);
        System.out.println(“Detiene durante 2 seg.”);
}
cath(InterruptedException e) {
        System.out.println(“Error de ejecución: “+e);
}
```

- Escribe el siguiente programa:

```
// j040.java
class j040 {
        public static void main(String[] args) {
                int x=10;
                System.out.println("x= "+x);
                x=15;
                System.out.println("x= "+x);
                System.out.println("Espero 5 segundos...");
                try {
                        Thread.sleep(5000);
                }
                catch(InterruptedException e) {
                        System.out.println("Error: "+e);
```

```
                    }
                    System.out.println("Ya han pasado 5 segundos");
                }
            }
```

- Grábalo en *TuCarpeta* con el nombre **j040.java**

- Compílalo y ejecútalo

- **try**
  Se utiliza para definir un bloque de instrucciones que puede generar uno o varios errores de excepción, que deberán tratarse en un bloque **catch**.

- **catch**
  Se utiliza para el tratamiento de errores de excepción generados por la **jvm** (Máquina Virtual Java).

Los Identificadores

Un identificador es un nombre que hace referencia a una variable, un método, una clase, una interfaz o a un paquete de programas Java.

Reglas de nomenclatura:
- El primer carácter debe ser una letra, el símbolo de subrayado (_) o el símbolo de dolar ($).
- Tras el primer carácter, puede haber letras, números o cualquier carácter Unicode superior a 0x00C0.
- No puede contener espacios, tabuladores, saltos de línea o de párrafo.
- Java diferencia entre mayúsculas y minúsculas (es Case Sensitive).
- No pueden usarse como identificadores nombres clave (nombres reservados por Java).

Ejemplos:

```
$Cuenta          // correcto
_Cuenta          // correcto
llegó1           es distinto de llego1
cuenta           es distinto de Cuenta
2Empleados       // incorrecto
if          // incorrecto
ça          // incorrecto
número#     // incorrecto, ya que # está por debajo de 0x00C0
```

Además, por **convención**:
- El nombre de una clase o interfaz se escribe con la primera letra en mayúscula (personalmente no sigo esta convención)
- El nombre de un método o variable empieza por minúscula.
- En el nombre de una constante, todas las letras están en mayúsculas.

Los Comentarios

Tres tipos de comentarios:
- Primer tipo: //
  Todo lo que está entre la doble barra y el retorno de línea, se considera un comentario.

Ejemplo:
```
// Declaración de la variable x de tipo entero
int x;
x += 2; // equivale a x=x+2
```

- Segundo tipo: /* y */
  Todo lo que está contenido entre /* y */ es un comentario. Este, nos permite escribir comentarios en varias líneas, así como en el interior de una instrucción.

  Ejemplo:
```
/* Declaración de
variables locales */
int x=20, y=10, z=5;
x=2, y = 5 / Equivale a y=y+5 */, z=10;
```

- Tercer tipo: /** y */
  Todo lo que está contenido entre /** y */ se considera como comentario.
  La única diferencia con el anterior, es que éste es el "oficial" de **Sun**

  Ejemplo:
```
/** El método main()
es el punto de entrada principal de una aplicación Java */
public static void main(String []args) {...}
```

- Escribe, a partir del programa j040.java, el siguiente programa:

```
// j041.java
public class j041 {
    /** A continuacion tenemos el metodo main() */
    public static void main(String[] args) {
            int x=10;
            System.out.println("x= "+x);
            x=15;
            System.out.println("x= "+x);
            System.out.println("Espero 5 segundos...");
            try {
                    Thread.sleep(5000);
            }
            catch(InterruptedException e) {
                    System.out.println("Error: "+e);
            }
            System.out.println("Ya han pasado 5 segundos");
    }
}
```

- Grábalo con el nombre **j041.java** en *TuCarpeta*

- Compílalo y ejecútalo.

En el paquete **JDK** de **Sun**, disponemos de

javac.exe	: compilador
java.exe	: intérprete para MS/DOS
appletviewer.exe	: visualizador de applets

Pero disponemos también del programa **javadoc.exe** que no es más que un generador de documentación del código fuente de un programa Java, en forma de páginas HTML, con la misma presentación que la documentación oficial de las clases Java editadas por **Sun**

Veamos cómo funciona:

- Sitúate en *TuCarpeta*, y desde MS/DOS

- Escribe:
  javadoc j041.java [Return]

- Investiga el contenido de los ficheros generados:
  allclasses-frame.html
  deprecated-list.html
  help-doc.html
  index.html
  index-all.html
  overview-tree.html
  package-list
  packages.html
  serialized-form.html
  stylesheet.css

Las constantes literales

Un literal es una constante que representa un valor no modiicable.
Un literal puede ser un número, un carácter, una cadena de caracteres o un booleano.

- Las literales numéricas

LOS ENTEROS
Pueden escribirse de tres formas:

- Decimal (base 10)
  Nunca debe empezar por 0
- Hexadecimal (base 16)
  Debe precederse por **0x** o **0X**
- Octal (base 8)
  Debe precederse por **0**.

15 en base 10 = 0xF en base 16 = 017 en base 8

0x12FA en base 16 es:
12Fa = $A+F\cdot16+2\cdot16^2+1\cdot16^3$ = $10+15\cdot16+2\cdot16^2+1\cdot16^3$ = 10+240+512+4096 = **4858 en base 10**

03571 en base 8 es:
3571 = $1+7\cdot8+5\cdot8^2+3\cdot8^3$ = 1+56+320+1536 = **1913 en base 10**

- Escribe el siguiente programa:

```
// j042.java
class j042 {
        public static void main(String[] args) {
                int x=15;
                int y=0xF;
                int z=017;
        /* La salida System.out.println de un numero entero
           siempre es en base 10.
```

```
                    Por lo tanto disponemos de un metodo muy facil
                    para transformar un numero en base 16 o 8 a
                    base 10 */
                        System.out.println("x en base 10 es 15, x= "+x);
                        System.out.println("0xF en base 10 es "+y);
                        System.out.println("017 en base 10 es "+z);
                        System.out.println("3571 en base 8 es "+
                                03571+" en base 10");
                        System.out.println("12FA en base 16 es "+
                                0x12FA+" en base 10");
                        System.out.println("Es tope Guapi Cachipirili");
                }
        }
```

- Grábalo con el nombre **j042.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Un entero puede almacenarse en cuatro tipos de datos:
byte (8 bits)
short (16 bits)
int (32 bits)
long (64 bits)

Por defecto, los literales enteros se almacenan en tipo **int** (excepto en el caso de que sobrepase el máximo int, en este caso se considera **long**)

También es posible hacer que un literal entero se almacene **long**: añadiendo **l** o **L** al final del literal

Ejemplos:
long x=4; // 4 puede ser tipo int: Java almacena el valor en x en 32 bits
long x=4L; // forzamos a Java a almacenar x en tipo long, es decir 64 bits

LOS REALES
Un real es un número con coma con o sin exponente.
Los literales reales usan el punto (.) para representar la coma flotante.
Un real puede almacenarse en dos tipos de datos:
float (32 bits)
double (64 bits)
Por defecto Java almacena los literales como tipo **double**.
Podemos forzar el almacenaje en **float**, añadiendo **f** o **F** al final del literal.

Ejemplos:
double x=4.85;
float x=4.85F;

Es posible utilizar la notación exponencial en los literales reales. Para ello debe añadirse “e” o “E” al final del literal seguido del exponente.

Ejemplos:
double x=4.85e10;
double y=4.85E-5;

- Escribe el siguiente programa:

```
// j043.java
class j043 {
        public static void main(String[] args) {
                double x=4.89234567812;
                double y=4.89234567812f;
                double z=4.8567e6;
                double w=4.8567E-12;
                System.out.println("x= "+x);
                System.out.println("y= "+y);
                System.out.println("z= "+z);
                System.out.println("w= "+w);
        }
}
```

- Graba el programa con el nombre **j043.java**

- Compílalo y ejecútalo.

- Observa detenidamente la salida del programa.

Los literales booleanos

Un booleano sólo puede tomar dos valores: **true o false**
En Java no podemos utilizar el 0 como falso y el 1 como verdadero, como sucede en el C/C++

Ejemplos:
boolean estáCansado=false;
boolean continuar=true;

Los literales carácter

Es un único carácter entre comillas simples: ‘A’, ‘0’, ...
Los caracteres se almacenan en 16 bits: 65.536 caracteres = 256 (Ascii) + Unicode

Java, igual que otros lenguajes, utiliza los literales caracter, llamados **caracteres de control** que no pueden visualizarse, pero que perrmiten representar caracteres especiales:

Carácter de control	**Significado**
\\	barra invertida
\	continuación
\’	comilla simple
\”	comillas dobles
\b	borrar atrás
\r	retorno de carro
\t	tabulación
\f	cambio de página
\n	nueva línea
\<valor octal>	carácter octal
\x<valor hexadecimal>	carácter hexadecimal
\u<valor unicode>	carácter unicode

- Escribe el siguiente programa:

```
// j044.java
class j044 {
      public static void main(String[] args) {
            System.out.println("Pepe\t\ttiene\n\t50 pts");
            System.out.println("El caracter 73 en octal es "+"\073");
            System.out.println("El caracter 6 en octal es "+"\006");
      }
}
```

- Grábalo en *TuCarpeta* con el nombre **j044.java**

- Compílalo y ejecútalo.

Los literales cadena de caracteres

En Java, no hay ningún tipo de datos para tratar las cadenas de caracteres, pero existe la clase **String**. En Java, una cadena es, pues, un objeto.

Los nombres clave de Java

abstrac - boolean - break - byte - byvalue - case - catch - char - class - const - continue - default - do - double - else - extends - false - final - finally - float - for - goto - if - implements - import - instanceof - int - interface - long - native - new - null - package - private - protected - public - return - short - static - super - switch - synchronized - this - throw - throws - transient - true - try - void - volatile - while.

Los operadores

- Operadores unarios

+	positivo
-	negación
~	complemento
++	incremento
--	decremento
!	opuesto

Ejemplos:
i++; // añade 1 al valor de i
++i; // añade 1 al valor de i
La diferencia entre estas dos instrucciones es que en el primer caso el operador se evalúa antes del incremento, mientras que en el segundo caso se evalúa tras el incremento.

Veamos el siguiente programa para entenderlo...

- Escribe el siguiente programa:

```
// j045.java
```

```
class j045 {
        public static void main(String[] args) {
                int i;
                i=0;
                do {
                  System.out.println("i= "+i);
                } while (i++<5);
                i=0;
                do {
                  System.out.println("i= "+i);
                } while (++i<5);
        }
}
```

- Grábalo en *TuCarpeta* con el nombre **j045.java**
- Compílalo y ejecútalo.
- Observa la diferencia entre los dos bucles.

- Los operadores binarios

- Operadores Aritméticos

+	suma
-	resta
*	multiplicación
/	división entera
%	módulo (resto de la división entera)

El operador "+" se utiliza además para la concatenación de cadenas de caracteres.

- Escribe el siguiente programa:

```
// j046.java
class j046 {
        public static void main(String[] args){
                int x=7,y=5;
                System.out.println("Suma= "+(x+y));
                System.out.println("Resta= "+(x-y));
                System.out.println("Producto= "+(x*y));
                System.out.println("Division entera= "+(x/y));
                System.out.println("Modulo= "+(x%y));
        }
}
```

- Grábalo con el nombre **j046.java** en *TuCarpeta*
- Compílalo y ejecútalo.

- Operadores de Comparación

==	igual
!=	distinto
<	inferior
>	superior
<=	inferior o igual
>=	superior o igual

No confundas el operador de asignación (=) con el de comparación (==)

Ejemplos:
5<2; // devuelve falso
x==y // prueba la igualdad entre x e y
x !=y // prueba si los valores x, y son distintos

- Operadores de asignación

=	asignación
+=	suma y asignación
-=	resta y asignación
*=	producto y asignación
/=	división y asignación
%=	módulo y asignación
&=	"Y" lógico y asignación
|=	"O" lógico y asignación
^=	"O" exclusivo y asignación
<<=	desplazamiento a la izquierda y asignación
>>=	desplazamiento a la derecha y asignación

Ejemplos:

```
x += 2;  // equivale a x=x+2
x %= 2; // equivale a x=x%2
x <<=2; // equivale a x= x<<2
```

- Operadores lógicos

!	no
&	y
|	o
^	o exclusivo
&&	y lógico
||	o lógico
&=	y con asignación
|=	o con asignación
^=	o exclusivo con asignación

Diferencia entre el operador "&" y el "&&":

- Cuando se utiliza el operador &, Java evalúa el valor de las expresiones situadas a ambos lados del operador, sea cual sea el valor de las expresiones.
- Cuando se utiliza el operador &&, Java evalúa la expresión situada a la izquierda del operador. Si es falsa, no se evalúa la expresión de la derecha, y el conjunto toma el valor falso. Sucede lo mismo con los operadores | y ||

Los Tipos de Datos

LAS VARIABLES

- **Variables de clase**: definen los atributos de la clase. Las variables de clase y sus valores se almacenan en la propia clase, y se aplican a todas sus instancias (objetos).
- **Variables de instancia**: definen los atributos de una única instancia de la clase
- **Variables locales**: se declaran y se utilizan en el interior de la definición de un método, en el interior de un bloque de instrucciones {}

- **La declaración de las variables**:

<tipo de datos> <identificador> [=valor por defecto];

Ejemplos:

```
String apellido;
int x,y,z;
boolean trabaja;
String apellido=”PEREZ”;
int x=5,y=10,z=15;
boolean trabaja=false;
```

Es posible declarar varias variables del mismo tipo en la misma instrucción: int x,y=10,z=3;
Normalmente la declaración de las variables se hace al principio del bloque, justo después de la declaración de la clase o el método.

- **Declaración de una variable con valor constante**

Para declarar una constante en Java, debe precederse la declaración de la variable con la palabra clave “final”. En este caso el valor de la variable no puede ser modificado.

Ejemplo:

```
final float e=166.386;
final float pi=3.141592;
```

- **Ámbito y redefinición de una variable**

En Java, una variable puede verse tan sólo en el interior del bloque en que se ha definido.
Un bloque es un conjunto de instrucciones comprendidas entre los operadores llave ({...}.
Si en un bloque se redefine una variable que ya existia en el bloque anterior, esta nueva variable enmascara la variable superior, pero sólo dentro de este bloque (y también para todos sus sub-bloques).

- Escribe el siguiente programa:

```
// j047.java
class j047 {
        // declaramos una variable global
        static int val=1;
        public static void main(String[] args) {
                System.out.println("La variable en el 'main' es "+
                        val);
                // nueva asignacion
                val=2;
                // llamamos al metodo procedimiento
```

```
                procedimiento();
                //llamamos al metodo funcion
                funcion(val);
        }
        // redefinimos la variable
        static void procedimiento() {
                char val='A';
                System.out.println("Ahora la variable es "+
                        val);
        }
        // vamos a ver el valor 'superior'
        static int funcion(int v) {
                System.out.println("La variable en la funcion es "+
                        v);
                return(v);
        }
}
```

- Grábalo en *TuCarpeta* con el nombre **j047.java**

- Compílalo y ejecútalo

Los tipos de datos simples

Tipo	Tamaño	Intervalo de valor
byte	8 bits	-128 a 127
short	16 bits	-32768 a 32767
int	32 bits	-2147483648 a 2147483647
long	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
float	32 bits	real simple precisión (1.402e-45 a 3.402e+38)
double	64 bits	real doble precisión (4.94e-324 a 1.79e+308)
char	16 bits	codificación Unicode en hexadecimal de 65.536 caracteres.
boolean	1 bit	true o false

- Escribe el siguiente programa:

```
// j048.java
class j048 {
  public static void main(String[] args) {
        System.out.println("El factorial de 25 no sobrepasa "+
                "el maximo numero 'int'");
        System.out.println("Trabajando con 'int' resulta:");
        int x=25,factur=1;
        for(int i=1;i<=x;i++)
                factur=factur*i;
        System.out.println("Factorial de 25= "+factur);
        System.out.println();
        System.out.println("El factorial de 26 sobrepasa "+
                "el maximo numero 'int'");
        System.out.println("Observa el resultado trabajando con 'int'");
        int x1=26,factur1=1;
        for(int i=1;i<=x1;i++)
                factur1=factur1*i;
        System.out.println("Factorial de 26= "+factur1);
        System.out.println();
        System.out.println("En cambio, ahora trabajo con numeros 'long'");
```

```
                System.out.println("El factorial de 26 es "+(long)26*2076180480);
  }
}
```

- Grábalo con el nombre **j048.java** en *TuCarpeta*

- Compílalo y ejecútalo.

El tipo **char** permite almacenar todo tipo de caracteres (letras, cifras, caracteres acentuados, símbolos), pero de forma individual.

- Escribe el siguiente programa:

```
// j049.java
class j049 {
  public static void main(String[] args) {
        int i;
        char x;
        System.out.println("Caracteres Unicode de 32 a 255");
        for(i=32;i<256;i++) {
              x=(char)(i);
              System.out.print(i+" "+x+"\t");
        }
  }
}
```

- Grábalo con el nombre **j049.java** en *TuCarpeta*

- Compílalo y ejecútalo.

Cada tipo simple contiene lo que se llama un **tipo compuesto**
Los tipos simples se aplican a las variables, mientras que los tipos compuestos sólo afectan a los objetos. De hecho, cada tipo compuesto es una clase: Byte, Short, Integer, Long, Float, Double, Char, Boolean.

Ejemplos:
int miEntero;
miEntero=10; o int miEntero=10;

Para declarar un objeto de tipo compuesto debe utilizarse uno de los **constructores** de dicha clase. Un constructor es un método de la clase que lleva el mismo nombre que la clase, y cuya primera letra es mayúscula.

Ejemplo:
Integer miEntero=new Integer(10);

Las variables simples se manipulan con los operadores y palabras clave de Java, encambio los objetos compuestos se manipulan con los métodos del objeto.

Los tipos de datos compuestos

Los tipos de clases

Las variables de Java también pueden poseer una clase por tipo. La variable hace entonces referencia a un objeto de dicha clase.

Ejemplos:
La clase Point, para una variable de tipo objeto de clase Point
Point pt;
La clase Color, para una variable de tipo objeto de clase Color
Color uncolor;

Las cadenas de caracteres

En Java puede declararse y manipularse una cadena de caracteres de dos formas:

- Utilizando una variable objeto de clase String, porque no hay ningún tipo simple que corresponda a las cadenas de caracteres.
- Utilizando una variable de tipo **tabla (array o arreglo o matriz o vector)** de char (más adelante).

La clase **String** forma parte del paquete **java.lang**
Los atributos y métodos de la clase String son directamente accesibles sin que sea necesario importar la clase:
import java.lang.String;
Tampoco es necesario utilizar el operador **new** para crear una instancia de la clase String, ya que Java lo hace implícitamente.
La declaración de un objeto de cadena de caracteres se asemeja, pues, a la declaración de un objeto de datos, con la única diferencia que podrá utilizar el operador punto para acceder al conjunto de atributos y métodos de su variable objeto de tipo String.

La declaración:
String apellido= “PEPE”;
es equivalente:
char[] tabChar={‘P’,’E’,’P’,’E’};
String apellido=new String(tabChar);
y equivalente a:
String apellido=new String(“PEPE”);

Vamos a ver un ejemplo de las distintas formas de declarar una cadena de caracteres, y uso del método “length” de la clase String, para mostrar la longitud de la cadena de caracteres.

- Escribe el siguiente programa:

```
// j050.java
class j050 {
        public static void main(String args[]) {
                char[] apellido={'P','E','P','E'};
                String apellido1=new String(apellido);
                String apellido2=new String("PEPE");
                String apellido3="PEPE";
                System.out.print("apellido: ");
                for(int i=0;i<=(apellido.length-1);i++) {
                        System.out.print(apellido[i]);
                }
                System.out.println();
                System.out.println("longitud apellido: "+
                        apellido.length);
                System.out.println("apellido1: "+apellido1);
```

```
                    System.out.println("longitud apellido1: "+
                            apellido1.length());
                    System.out.println("apellido2: "+apellido2);
                    System.out.println("longitud apellido2: "+
                            apellido2.length());
                    System.out.println("apellido3: "+apellido3);
                    System.out.println("longitud apellido3: "+
                            apellido3.length());
                }
        }
```

- Graba el programa en *TuCarpeta* con el nombre **j050.java**

- Compílalo y ejecútalo.

Las Tablas (o Arrays o Arreglos o Matrices o Vectores o Listas)

Una tabla es una variable que contiene una lista de elementos del mismo tipo de datos simples o de la misma clase. Cada elemento se almacena en su propia casilla, que está indexada (desde 0), lo que permite acceder fácilmente a los diferentes elementos de la tabla.

La variable tabla puede contener cualquier tipo de información: numérica (enteros o reales), booleana, carácter, cadena de caracteres (String), o cualquier objeto. Es posible crear una tabla de tablas, lo que permite obtener una tabla multidimensional.

No puede crearse una tabla que contenga simultáneamente enteros y cadenas de caracteres.

Una vez se ha creado la tabla es imposible modificar el tipo de información que se encuentra almacenada.

La creación de una tabla Java consta de tres etapas:
1º) Declaración de la variable
2º) Creación del objeto tabla y asignación a la variable tabla.
3º) Almacenamiento de la información en la tabla.

Declaración de una variable tabla

<tipo>[] <nombre>;
o bien:
<tipo> <nombre>[];

<tipo>: tipo de datos que se almacenarán en la tabla
<nombre>: nombre de la variable tabla

Ejemplos:

```
// tabla de enteros llamada notas
int[] notas;
// dos tablas de cadenas
String[] apellido, nombre;
// tabla de objetos Color de nombre colores
Color colores[];
// tabla bidimensional
float coordenadas[][];
```

Crear el objeto tabla

Una vez declarada la variable de tabla, debe crearse la tabla y dimensionarla para asignarle un espacio de memoria.
Dado que las tablas son objetos en Java, debe usarse el operador **new** o inicializar directamente el contenido de la tabla con los operadores llaves: {}

Podemos utilizar las tres posibilidades siguientes:
/* Utilizar únicamente si <tipo>[] <nombre> ha sido declarado anteriormente */
<nombre>=new <tipo>[noElementos];

<tipo> <nombre>[] = new <tipo>[noElementos];

<tipo> <nombre> = {elemento1, elemento2,...., elemento3};

Ejemplos:
Color colores[];
colores=new Color[3];

String apellido[]=new String[2];
Color[] colores=new Color[3];

String[] apellido={"Pepe","Paco"};
Color colores[]={Color.red,Color.green,Color.yellow};

En el caso de tablas multidimensionales, funciona igual:

byte[][] boletinDeNotas;
boletinDeNotas = new byte[5][6];

float coordenadas[][]=new float[3][2];
float coordenadas[][]= {{0.0,0.1,0.2},{1.0,1.1}};

Acceso a los elementos de una tabla

Tras haber declarado previamente una tabla y de haberla dimensionado, debe llenarse cada uno de los elementos de la tabla.

Para ello se utiliza:
<nombre>[<índice>] = <valor>;
o
<nombre>[<índice>][<índice>] = <valor>;

Ejemplos:
char[] cNombre= new char[4];
cNombre[0]='P';
cNombre[1]='E';
cNombre[2]='P';
cNombre[3]='E';

float [][]coordenadas=new float[2][3];
coordenadas[0][0] = 0.0;
coordenadas[0][1]=0.1;
...
...

Para leer los datos de una tabla:

```
char c;
for(int i=0;i<=(cNombre.length-1);i++) {
        c=cNombre[i];
        System.out.print(c+" ");
}

System.out.println(coordenadas[0][0]); // muestra 0.0
```

Conocer la dimensión de una tabla

Para conocer la dimensión de una tabla basta con indicar el nombre de la variable de tipo tabla, seguida de un punto y de la propiedad length.

Para conocer la dimensión de una tabla multidimensional...

```
/* Creación de una tabla de enteros de dos dimensiones 10x20 */
int i[][]=new int[10][20];
System.out.println(i[0].length);
// muestra 10 - 1ªtabla o dimensión
System.out.println(i[1].length);
// muestra 20-2ªdimensión.
```

- Escribe el siguiente programa:

```
// j051.java
import java.awt.*;
class j051 {
    public static void main(String []args) {
        Color[] c1={Color.red,Color.green,Color.yellow};
        System.out.println("Tamaño de c1: "+c1.length);
        for(int i=0;i<3;i++)
            System.out.println(c1[i]);
        Color[] c2=new Color[3];
        c2[0]=Color.yellow;
        c2[1]=Color.green;
        c2[2]=Color.red;
        for(int i=0;i<3;i++)
            System.out.println(c2[i]);
        char[] apellido={'p','e','p','e'};
        String apellido1=new String(apellido);
        String apellido2=new String("pacos");
        String apellido3="felipe";
        System.out.print("apellido: ");
        for(int i=0;i<=(apellido.length-1);i++)
            System.out.println(apellido[i]);
        System.out.println();
        System.out.println("longitud apellido: "+
            apellido.length);
        System.out.println("apellido1: "+apellido1);
        System.out.println("longitud apellido1: "+
            apellido1.length());
        System.out.println("apellido2: "+apellido2);
        System.out.println("longitud apellido2: "+
            apellido2.length());
        System.out.println("apellido3: "+apellido3);
        System.out.println("longitud apellido3: "+
            apellido3.length());
    }
```

```
}
```

- Grábalo con el nombre **j051.java** en *TuCarpeta*
- Compílalo y ejecútalo.

Conversión de tipos de datos simples

A veces nos interesará convertir el tipo de datos de una variable para, por ejemplo, pasar un parámetro **byte** a un método que sólo acepta **int**.

Para convertir el tipo de datos de una variable, debe prefijarse la variable con el nuevo tipo entre parentesis:

(<tipo>)<variable>;

- Escribe el siguiente programa:

```
// j052.java
class j052 {
        public static void main(String args[]) {
                char ch='b';
                System.out.println("Valor de ch: "+ch);
                // conversion en entero short
                short sh=(short)ch;
                System.out.println("Valor de sh: "+sh);
                // conversion en real float
                float fl=(float)sh;
                System.out.println("Valor de fl: "+fl);
        }
}
```

- Grábalo con el nombre **j052.java** en *TuCarpeta*
- Compílalo y ejecútalo.

Debe prestarse especial atención al realizar conversiones de tipos de datos.

Por ejemplo, al convertir una variable **float** en una variable **int**, se pierden datos:

```
float jl1=2.12f;
int i=(int)fl1;
System.out.println("Val de i: "+i);
```

Aparecerá: **Val de i: 2**

O al convertir una variable de 64 bits a una variable de 8 bits, se obtiene un resultado erróneo:

```
double d=2e231;
byte i1=(byte)d;
System.out.println("Val de i1: "+i1);
```

Aparecerá: **Val de i1: -1**

Por norma general, deben convertirse tipos del mismo tamaño de memoria, o de un tipo de un cierto tamaño de memoria a un tipo de tamaño mayor.

Las Estructuras de Control

Las instrucciones condicionales

if

Permite ejecutar instrucciones en función de la prueba de una condición. La condición debe devolver obligatoriamente un booleano: true o false.

- Sintaxis 1

```
if (<condición>) <instrucción>;
```

- Sintaxis 2:

```
if (<condición>)
        <instrucción1>;
else
        <instrucción2>;
```

- Sintaxis 3:

```
if (<condición>) {
        <secuencia de instrucciones>;
}
```

- Sintaxis 4

```
if (<condición>) {
        <secuencia de instrucciones>;
}
else  {
        <secuencia de instrucciones>;
}
```

- Sintaxis 5:

```
if (<condición>) {
        <secuencia de instrucciones>;
}
else if(<condición2>) {
        <secuencia de instrucciones>;
}
else if(<condición 3>) {
        <secuencia de instrucciones>;
}
else {
        <secuencia de instrucciones>;
}
```

Ejemplo:

```
if(A>B) {
        System.out.println("A es mayor que B");
}
else if(A==B) {
        System.out.println("A y B son iguales");
}
else {
        System.out.println("B es mayor que A");
}
```

Java posee también una instrucción bajo la forma:

x ? expr1 : expr2;

Si x es verdadera se evaluará “expr1”, en caso contrario “expr2”

x ? expr1 : expr2;

es equivalente a:

```
if (x) expr1;
else expr2;
```

Ejemplo:

```
if(color==”blanco”) {
        color=”negro”;
}
else {
        color=”blanco”;
}
```

es equivalente a:

color=(color==”blanco” ? “negro””: “blanco”);

switch

Antes de encadenar una serie de condiciones if...else...else if..., es más juicioso utilizar la estructura “switch”, que permite ejecutar una de las secuencias de instrucciones especificadas en función del valor de una expresión:

```
switch (<expresión>) {
        case <expr1>:
                <secuencia de instrucciones 1>;
                break;
        case <expr2>:
                <secuencia de instrucciones 2>;
                break;
        case <expr3>:
                <secuencia de instrucciones 3>;
                break;
        default:
                <secuencia de instrucciones 4>
}
```

El **break** es necesario para aislar cada uno de los casos. Por tanto, si un caso cumple la condición, se ejecutan sus instrucciones, y el break hace salir del bucle. Entonces los casos siguientes no se comprueban.

La etiqueta **default** puede utilizarse para ejecutar una secuencia de instrucciones en caso de que no se haya cumplido la condición del test.

Ejemplo:

```
switch (color) {
        case “rojo”:
                peon=Color.red;
                break;
        case “verde”:
                peon=Color.green;
                break;
        case “naranja”:
                peon=Color.orange;
                break;
        default:
```

```
                System.out.println("Color no refenciado");
}
```

Las instrucciones repetitivas

for

Permite repetir un cierto número de veces, mientras se cumpla una condición, la ejecución de una secuencia de instrucciones.

```
for (<contador=inicio>;<test contador>;<incremento>) {
        <secuencia de instrucciones>;
}
```

Ejemplo: Introduce y muestra el valor de cada índice de una tabla.

```
        String []code=new String[4];
        int numero=0;
        for (int i=0;i<4;i++) {
            code[i]=String.valueOf(numero+i);
            System.out.println(code[i]);
        }
```

while

Ejecuta una secuencia de instrucciones mientras sea cierta una condición. La condición se comprueba antes de la ejecución del bucle.
Si desde la primera prueba la condición es falsa, el bucle no se ejecutará jamás.

```
while <condición> {
        <secuencia de instrucciones>;
}
```

Ejemplo: Un bucle sin fin

```
while (true) {
        System.out.println("Está en un bucle sin fin");
}
```

do...while

Ejecuta una secuencia de instrucciones al menos una vez (condición después de la secuencia de instrucciones), mientras que la condición sea verdadera.

```
do {
        <secuencia de instrucciones>;
}
while (<condición>);
```

Ejemplo: Ejecuta el bucle mientras que "i" no sea igual a 10

```
int i;
do {
        i=i+1;
        System.out.println("i: "+i);
}
while(i != 10);
```

Las instrucciones “break” y “continue”

Las instrucciones **break** y **continue** permiten salir de un bloque de instrucciones que se están ejecutando una vez que el test de la condición ha sido realizado y verificado.

break

Utilizado en un bloque de instrucciones ({...}, “break” permite interrumpir la ejecución de la secuencia de instrucciones hasta que se cumple una condición booleana, y pasar a la siguiente instrucción situada tras el final del bloque.

En el caso de un programa con bloques anidados, la instrucción **break**, permite salir del bloque en curso y continuar a partir del bloque situado en el nivel superior.

- Escribe el siguiente programa:

```
// j053.java
class j053 {
    public static void main(String []args) {
            int i=0;
            while(true) {
                    if(i==10) break;
                    System.out.println("Bucle sin fin "+i);
                    i=i+1;
            }
    }
}
```

- Grábalo con el nombre **j053.java** en *TuCarpeta*

- Compílalo y ejecútalo.

La instrucción “break” puede utilizarse también para hacer un desvio hacia una etiqueta situada fuera de un bloque anidado.
Entonces se sale del bloque indicado por la etiqueta.

- Escribe el siguiente programa:

```
// j054.java
class j054 {
    public static void main(String []args) {
            int i=0,j=0;
            etiqueta:
            for(i=0;i<3;i++) {
                    System.out.println("Bucle 1: i= "+i+
                            ", j= "+j);
                    for(j=0;j<3;j++) {
                            System.out.println("Bucle 2: i= "+i+
                                    ", j= "+j);
                            if((i+j)==3) break etiqueta;
                    }
            }
    }
}
```

- Grábalo con el nombre **j054.java** en *TuCarpeta*

- Compílalo y ejecútalo

- Observa que para (i+j==3) “break etiqueta”, hace salir del bloque indicado por “etiqueta:”, es decir el bucle for.

continue

Utilizada en un bloque, “continue” permite prohibir la ejecución de una secuencia de instrucciones al cumplirse una condición booleana, y de reiniciar el tratamiento del bloque en el que estaba situada dicha condición.

- Escribe el siguiente programa:

```
// j055.java
class j055 {
	public static void main(String []args) {
		for(int i=0;i<=4;i++) {
			if(i==2) continue;
			System.out.println("i= "+i);
		}
	}
}
```

- Grábalo con el nombre **j055.java** en *TuCarpeta*

- Compílalo y ejecútalo.

La instrucción **continue** puede también utilizarse para realizar un desvio hacia una etiqueta situada fuera de un bloque anidado. Entonces se ejecuta el bloque de la etiqueta.

- Escribe el siguiente programa:

```
// j056.java
class j056 {
	public static void main(String []args) {
		int i=0,j=0;
		etiqueta:
		for(i=0;i<3;i++) {
			System.out.println("Bucle 1: i= "+i+
				", j= "+j);
			for(j=0;j<3;j++) {
				System.out.println("Bucle 2: i= "+i+
					", j= "+j);
				if((i+j)==3) continue etiqueta;
			}
		}
	}
}
```

- Grábalo con el nombre **j056.java** en *TuCarpeta*

- Compílalo y ejecútalo

- Para ((i+j)==3) “continue etiqueta;”, hace reiniciar el bloque indicado por “etiqueta:”, es decir, el bucle for.

Autoevaluación III

1) Escribe un programa de nombre **evalj3a**, que escriba los números pares inferiores a 30 pero con un intervalo de 3 seg. en la aparición de cada número.

2) Calcula matemáticamente:
   a.- El número 3BE que está en base 16 a base 10.
   b.- El número 277 que está en base 8 a base 10.
   c.- El número BFF que está en base 16 a base 10.
   d.- El número 1111 que está en base 8 a base 10
   e.- Haz un programa de nombre **evalj3b**, que sirva para comprobar los resultados de los apartados anteriores.

3) Escribe el siguiente programa:

```
// evalj3c.java
import java.util.Date;
public class evalj3c {
      public static void main(String args[]) {
            Date x=new Date();
            System.out.println(x);
      }
}
```

- Grábalo en *TuCarpeta* con el nombre **evalj3c.java**, compílalo y ejecútalo.
- Contesta a las siguientes preguntas:
  a.- ¿Qué utilidad tiene **Date()**?
  b.- ¿**Date** es una clase o un objeto?
  c.- ¿**Date()** que es?
  d.- ¿x es una clase o un objeto?

4) Escribe el siguiente programa:

```
// evalj3d.java
public class evalj3d {
  public static void main(String args[]) {
            System.out.println("Math.random()="+Math.random());
            System.out.println("Math.random()="+Math.random());
            System.out.println("Math.random()="+Math.random());
            System.out.println("Math.round(3.4)="+Math.round(3.4));
            System.out.println("Math.round(3.8)="+Math.round(3.8));
            System.out.println();
            for(int i=0;i<100;i++)
                  System.out.print(Math.round(9*Math.random()+1)+"\t");
  }
}
```

- Grábalo en *TuCarpeta* con el nombre **eval3d.java**, compílalo y ejecútalo.
- Contesta a las siguientes preguntas:

a.- ¿Para qué sirven las funciones Math.random() y Math.round()?
b.- ¿Qué relación hay entre "Math" por un lado y "random()", "round()" por otro?
c.- Indica la expresión correspondiente a un número aleatorio entero menor o igual a 5 (que sea positivo y distinto de cero).
d.- Indica la expresión correspondiente a un número aleatorio entero menor o igual a 10 (que sea positivo, y que pueda ser cero).

5) Escribe un programa de nombre **evalj3e.java** que simule la tirada aleatoria de 5 dados de parchís.

6) Escribe el siguiente programa:

```
// evalj3f.java

class evalj3f {

   public static void main( String args[] ) {
      StringBuffer str = new StringBuffer( "Hola" );
      str.append( " Mundo" );

      System.out.println( str );
   }
}
```

- Grábalo en *TuCarpeta* con el nombre **evalj3f.java**, compílalo y ejecútalo.
- Contesta a las siguientes preguntas:
  a.- Qué son los siguientes términos: StringBuffer
  StringBuffer(cadena)
  str
  append(cadena)
  b.- ¿Qué utilidad tienen la clase "StringBuffer" y el método "append".

7) Escribe el siguiente programa:

```
// evalj3g.java

public class evalj3g {

   public static String cadenaInversa( String fuente ) {
      // Se obtiene la longitud de la cadena que se pasa
      int longitud = fuente.length();

      // Se crea un stringbuffer de la longitud de la cadena
      StringBuffer destino = new StringBuffer( longitud );

      // Se recorre la cadena de final a principio, añadiendo
      // cada uno de los caracteres leidos al stringbuffer
      for( int i=(longitud-1); i >= 0; i-- )
         destino.append( fuente.charAt( i ) );
```

```
        // Devolvemos el contenido de la cadena invertida
        return( destino.toString() );
        }


    public static void main( String args[] ) {
        // Imprime el resultado inverit la cadena que se toma por
        // defecto
        System.out.println( cadenaInversa( "Hola Mundo" ) );
        }
}
```

- Grábalo en *TuCarpeta* con el nombre **evalj3g.java**. Compílalo y ejecútalo.
- Contesta a las siguientes preguntas:
  a.- ¿"cadenaInversa" es un método o una propiedad y de qué clase forma parte?
  b.- Indica la función de los términos: "fuente", "length()" y "longitud".

Autoevaluación III

1) Escribe un programa de nombre **evalj3a**, que escriba los números pares inferiores a 30 pero con un intervalo de 3 seg. en la aparición de cada número.

```
// evalj3a.java
public class evalj3a {
        public static void main(String args[]) {
                for(int i=2;i<30;i=i+2) {
                        System.out.print(i+"-");
                        try {
                                Thread.sleep(3000);
                        }
                        catch(InterruptedException e) {
                                System.out.println("Error: "+e);
                        }
                }
        }
}
```

2) Calcula matemáticamente:
   a.- El número 3BE que está en base 16 a base 10.
   b.- El número 277 que está en base 8 a base 10.
   c.- El número BFF que está en base 16 a base 10.
   d.- El número 1111 que está en base 8 a base 10
   e.- Haz un programa de nombre **evalj3b**, que sirva para comprobar los resultados de los apartados anteriores.

```
// evalj3b.java
public class evalj3b {
        public static void main(String args[]) {
                int x=0x3BE,y=0xBFF;
                int a=0277,b=01111;
                System.out.println("3BE (base 16) en base 10 es "+x);
                System.out.println("277 (base 8) en base 10 es "+a);
                System.out.println("BFF (base 16) en base 10 es "+y);
                System.out.println("1111 (base 8) en base 10 es "+b);
        }
}
```

3) Escribe el siguiente programa:

```
// evalj3c.java
import java.util.Date;
public class evalj3c {
        public static void main(String args[]) {
                Date x=new Date();
                System.out.println(x);
        }
}
```

- Grábalo en *TuCarpeta* con el nombre **evalj3c.java**, compílalo y ejecútalo.
- Contesta a las siguientes preguntas:

a.- ¿Qué utilidad tiene **Date()**?
b.- ¿**Date** es una clase o un objeto?
c.- ¿**Date()** que es?
d.- ¿x es una clase o un objeto?

a.- Nos da la fecha y hora actual del sistema
b.- Date es una clase del paquete java.util
c.- Date() es el constructor de la clase Date
d.- Es un objeto (u instancia de Date)

4) Escribe el siguiente programa:

```
// evalj3d.java
public class evalj3d {
  public static void main(String args[]) {
        System.out.println("Math.random()="+Math.random());
        System.out.println("Math.random()="+Math.random());
        System.out.println("Math.random()="+Math.random());
        System.out.println("Math.round(3.4)="+Math.round(3.4));
        System.out.println("Math.round(3.8)="+Math.round(3.8));
        System.out.println();
        for(int i=0;i<100;i++)
              System.out.print(Math.round(9*Math.random()+1)+"\t");
  }
}
```

- Grábalo en *TuCarpeta* con el nombre **eval3d.java**, compílalo y ejecútalo.
- Contesta a las siguientes preguntas:
  a.- ¿Para qué sirven las funciones Math.random() y Math.round()?
  b.- ¿Qué relación hay entre "Math" por un lado y "random()", "round()" por otro?
  c.- Indica la expresión correspondiente a un número aleatorio entero menor o igual a 5 (que sea positivo y distinto de cero).
  d.- Indica la expresión correspondiente a un número aleatorio entero menor o igual a 10 (que sea positivo, y que pueda ser cero).

a.- Math.random(), calcula un número aleatorio entre 0 y 1. Math.round(x) calcula el entero más próximo a "x".
b.- "Math" es una clase y "random(), round()" son dos métodos de la clase "Math".
c.- Math.round(Math.random()*4+1)
d.- Math.round(Math.random()*10)

5) Escribe un programa de nombre **evalj3e.java** que simule la tirada aleatoria de 5 dados de parchís.

```
// evalj3e.java
class evalj3e {
      public static void main(String args[]) {
            for(int i=0;i<5;i++)
                  System.out.print(Math.round(5*Math.random()+1)+"\t");
      }
}
```

6) Escribe el siguiente programa:

```
// evalj3f.java

class evalj3f {

   public static void main( String args[] ) {
      StringBuffer str = new StringBuffer( "Hola" );
      str.append( " Mundo" );

      System.out.println( str );
   }
}
```

- Grábalo en *TuCarpeta* con el nombre **evalj3f.java**, compílalo y ejecútalo.
- Contesta a las siguientes preguntas:
  a.- Qué son los siguientes términos: StringBuffer
  StringBuffer(cadena)
  str
  append(cadena)
  b.- ¿Qué utilidad tienen la clase "StringBuffer" y el método "append".

a.- StringBuffer: una clase StringBuffer(cadena): un constructor de la clase StringBuffer
str: un objeto de la clase StringBuffer
append: un método de la clase StringBuffer
b.- StringBuffer: gestiona las cadenas de texto del "buffer"
append: añade text al "buffer".

7) Escribe el siguiente programa:

```
// evalj3g.java

public class evalj3g {

   public static String cadenaInversa( String fuente ) {
      // Se obtiene la longitud de la cadena que se pasa
      int longitud = fuente.length();

      // Se crea un stringbuffer de la longitud de la cadena
      StringBuffer destino = new StringBuffer( longitud );

      // Se recorre la cadena de final a principio, añadiendo
      // cada uno de los caracteres leidos al stringbuffer
      for( int i=(longitud-1); i >= 0; i-- )
         destino.append( fuente.charAt( i ) );

      // Devolvemos el contenido de la cadena invertida
      return( destino.toString() );
      }


   public static void main( String args[] ) {
      // Imprime el resultado inverit la cadena que se toma por
      // defecto
      System.out.println( cadenaInversa( "Hola Mundo" ) );
```

```
        }
    }
```

- Grábalo en *TuCarpeta* con el nombre **evalj3g.java**. Compílalo y ejecútalo.
- Contesta a las siguientes preguntas:
  a.- ¿"cadenaInversa" es un método o una propiedad y de qué clase forma parte?
  b.- Indica la función de los términos: "fuente", "length()" y "longitud".

a.- cadenaInversa es un método de la clase evalj3g
b.- fuente: argumento del método "cadenaInversa" que en nuestro caso es "Hola Mundo"
length(): es una función que devuelve la longitud de una cadena
longitud: es un número entero que representa la longitud de la cadena a invertir

Hasta aquí, la versión no registrada del manual.

Si deseas la parte que falta, es decir:

Debes adquirir la versión registrada, es decir entera.

Es muy fácil, has de hacer lo siguiente:

1) **Rellena el siguiente formulario con tus datos:**

Nombre y Apellidos:

Dirección:

Código Postal: **Población:**

Versión completa del "Java 2 (Manual FV)"

2) **Envíame el formulario anterior por correo ordinario junto con un "billete" (lo que consideres justo por un disquete, gastos de manipulación, envío y "estímulo por mi parte para que continue colgando en Internet, mis manuales").**

   A mi dirección que es:

   F.V.
   c) Valencia 21-25, 2º , 4ª
   08915 – Badalona (Barcelona)
   España

3) **A vuelta de correo recibirás en tu dirección, un disquete con la versión completa del manual "JavaScript (Manual FV)".**