

Java

12

ÍNDICE

1. Instala Eclipse en castellano

1.1. Eclipse

2. Aplicación Bienvenido

2.1. Java

2.2. Clases y objetos

2.3. Propiedades y métodos

2.4. Función Main()

3. Aplicaciones con un botón

3.1. Elementos gráficos

3.2. Marcos y distribución de objetos en la ventana

3.3. Botones y ActionListener

4. Aplicación con dos botones

4.1. Corrección de errores

4.2. Botones y ActionListener

5. Aplicación Cambio de moneda

5.1. Etiquetas

5.2. Conversión a numéricos y símbolos decimales

5.3. Control de errores: try-catch

6. Aplicación Concentración

6.1. Etiquetas

6.2. Color y tipo de fuente

6.3. Tipos de datos y operadores aritméticos

6.4. Instrucción IF

7. Aplicación Dividir

7.1. Títulos y bordes

7.2. Caracteres y cadenas de caracteres

7.3. Expresiones y condiciones

7.4. If..., else e if anidados

8. Aplicación Poliedros

8.1. Listas desplegables

8.2. Estructura de control: switch

Curiosidades

Multimedia 09Multimedia

Iconos para aplicaciones

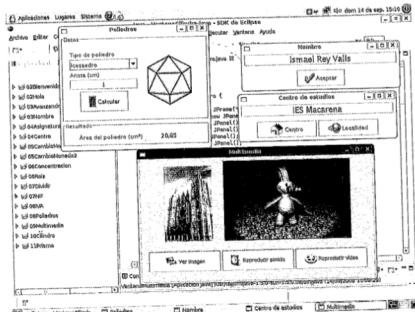
Taller de investigación

Crea una aplicación ejecutable .jar: 10Cilindro y Cilindro.jar

Java e Internet

Applets 11Prisma.html

Aplicación de applets



¿Qué es y para qué sirve Java?

Java es un lenguaje de programación de alto nivel y orientado a objetos, que sirve para desarrollar aplicaciones de propósito general de ámbito empresarial, educativo y lúdico.

Las aplicaciones creadas con **Java** se caracterizan por ser multplataforma, es decir, que sus aplicaciones pueden ser ejecutadas en **Linux** o **Windows**.

Eclipse es un entorno de software libre para el desarrollo de aplicaciones, muy extendido entre los programadores de **Java**.

Programas
Eclipse 3.2.2 en castellano para **Linux**.

Página web
<http://archive.eclipse.org>

Organización

En el **DVD Utilidades** hay una carpeta que se llama **12Java**; debes copiarla en tu carpeta **TIC** para hacer algunas de las actividades de esta unidad.

1. Instala Eclipse en castellano

1. INSTALA ECLIPSE EN CASTELLANO

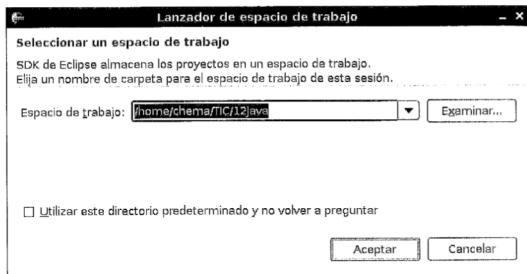
Mueve a tu carpeta personal el programa **Eclipse**.

- Copia del **DVD Utilidades** la carpeta **12Java** en tu carpeta personal **TIC**
- Mueve de tu carpeta personal **TIC/12Java** la carpeta **eclipse** a tu **Carpeta personal** o a la que quieras.
- Crea, en el escritorio, un lanzador a la aplicación situada en la carpeta **eclipse**
- Cierre todas las ventanas.

2. ENTORNO ECLIPSE

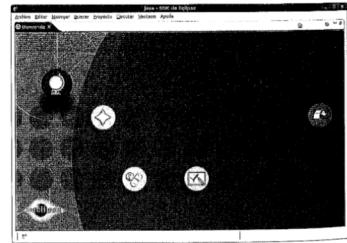
Familiarízate con el entorno de trabajo de **Eclipse**.

- Inicia
- En **Espacio de trabajo** elige tu carpeta **TIC/12Java** y no actives la opción **Utiliza este directorio predeterminado y no volver a preguntar**. Puedes pulsar



c) Haz clic en **Entorno de trabajo**.

- Maximiza la ventana.
- A la izquierda, cierra los paneles **Jerarquía** y **Explorador de paquetes** con
- Elige **Ventana/Mostrar vista/Jerarquía**.
- Elige **Ventana/Mostrar vista/Explorador de paquetes**.
- Arrastra el panel **Jerarquía** a la derecha, junto a **Esquema**.
- Arrastra el panel **Jerarquía** a la parte inferior, junto a **Problemas**.
- Vuelve a poner el panel **Jerarquía** a la izquierda, junto a **Explorador de paquetes**.



1.1. ECLIPSE

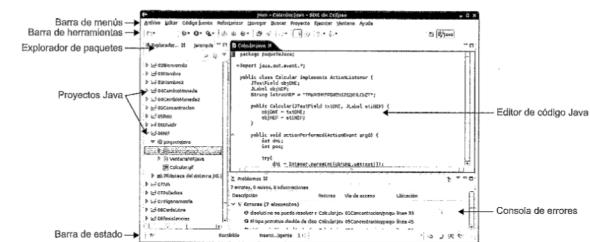
Eclipse es un entorno integrado de desarrollo en **Java**, independiente de la plataforma, que permite crear aplicaciones mediante programación para **Linux**, **Windows**, **Mac**, etc.

La última versión en castellano, en agosto de 2008, y con la que trabajaremos es la **versión 3.2.2**.

Eclipse es software libre y por tanto se puede copiar y distribuir sin ningún problema.

La primera vez que iniciamos **Eclipse** indicamos la carpeta de trabajo. Cada usuario debe tener su propia carpeta, cuyo nombre tendrá la forma **[Usuario]/TIC/12Java**. Es importante no activar la opción **Utiliza este directorio predeterminado y no volver a preguntar** para que cada usuario pueda seleccionar su espacio de trabajo al iniciar **Eclipse**.

En el caso de que algún usuario, al iniciar **Eclipse**, active la opción **Utiliza este directorio predeterminado y no volver a preguntar**, podemos cambiarlo en **Archivo/Comutar espacio de trabajo...**. Para acceder al área de trabajo hacemos clic en **Entorno de trabajo**.



RESUELVE

1 Ayuda

Consulta la ayuda de **Eclipse**.

- En la barra de menús elige **Ayuda/Bienvenida**.
- Haz clic en **Visión general**.
- Revisa los **Conceptos básicos del entorno de trabajo**.
- Al finalizar cierra la ayuda, cierra el panel **Bienvenida** con y cierra **Eclipse**.
- Comprueba el directorio de tu espacio de trabajo, tiene que haber creado la carpeta **.metadata**

EXPERIMENTA

APRENDE

2. Aplicación Bienvenido**1. PRIMERA APLICACIÓN: 02Bienvenido**

Realiza una aplicación utilizando el lenguaje Java que muestre el mensaje **Bienvenido a Java** en un cuadro de texto.

- Abre el Explorador de paquetes y crea un nuevo proyecto.
- En **Espacio de trabajo**, elige tu carpeta **TIC/12Java** y no actives la opción **Utiliza este directorio predeterminado y no volver a preguntar**.
- Maximiza la ventana y elige **Archivo/Nuevo/Proyecto...**
- Selecciona **Proyecto Java** y pulsa **Crear**.
- En **Nombre de proyecto** escribe **02Bienvenido** y pulsa **Crear**.
- Elige **Archivo/Nuevo/Clase**.
- En **Paquete**, escribe **paqueteJava** comenzando en minúsculas; en **Nombre** escribe **VentanaBienvenido** y pulsa **Crear**.
- Cierra la ventana de la parte derecha **Bienvenido**.
- En el **editor de código**, entre las líneas ya existentes, escribe el siguiente código de **Java**:

```
package paqueteJava;
import java.awt.*;
import javax.swing.*;
public class VentanaBienvenido {
```

- Dentro de la clase **VentanaBienvenido**, escribe el código de declaración de los objetos que vas a utilizar: una ventana, un marco y un cuadro de texto.

```
public class VentanaBienvenido {
    JFrame miVentana = new JFrame("Bienvenido!");
    JPanel marPrincipal = new JPanel();
    JTextField txtBienvenido = new JTextField(20);
}
```

- Dentro de la clase **VentanaBienvenido**, a continuación de la declaración de objetos y antes de la llave que la cierra, escribe el código del constructor de la clase **VentanaBienvenido**.

```
public VentanaBienvenido() {
    miVentana.setSize(400, 100);
    miVentana.setLocationRelativeTo(null);
    miVentana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    miVentana.setLayout(new BorderLayout());
    txtBienvenido.setText("Bienvenido a Java");
    txtBienvenido.setFont(new Font("Arial", 0, 20));
    txtBienvenido.setForeground(Color.BLUE);
    txtBienvenido.setBackground(Color.YELLOW);
    txtBienvenido.setHorizontalAlignment(JTextField.CENTER);
    marPrincipal.add(txtBienvenido);
    miVentana.add("North", marPrincipal);
    miVentana.setVisible(true);
}
```

- Dentro de la clase **VentanaBienvenido**, a continuación del constructor, después de cerrar su llave y antes de cerrar la llave de la clase, escribe el código de la función **main**.

```
public static void main(String[] args) {
    new VentanaBienvenido();
}
```

m) Haz clic en Guardar.

- En el **Explorador de paquetes**, selecciona **VentanaBienvenido.java** y en su menú **Contextual** elige **Ejecutar como/1 Aplicación Java**.
- Cierra la ventana **1Bienvenido**, y cierra Eclipse.

2.1. JAVA

Java es el lenguaje de programación más extendido en el ámbito de **Internet** y en el mundo empresarial, con el que podemos realizar aplicaciones informáticas. Las aplicaciones de **Java** pueden ejecutarse en cualquier plataforma, ya sea **Windows, Linux, etc.**

2.2. CLASES Y OBJETOS

Java es un lenguaje de programación orientado a objetos. Cada objeto es la representación de un concepto en una aplicación, como ventanas, marcos, textos, botones, etc. Para definir un objeto utilizamos las **clases**, que definen su forma y comportamiento. Para definir una **clase** utilizaremos la siguiente sintaxis:

```
public class VentanaBienvenido {
```

La clase es solo la definición del objeto. El **objeto** lo creamos cuando lo declaramos en la aplicación. Para declararlo utilizaremos el comando **new**. Observa los ejemplos.

```
JFrame miVentana = new JFrame("Bienvenido!");
 JPanel miMarco = new JPanel();
 new VentanaBienvenido();
```

Las clases se agrupan en **paquetes**. Podemos utilizar clases definidas en otros paquetes, importar una clase concreta o utilizar el símbolo *****, que significa todas las clases del paquete.

```
import java.awt.*;
import javax.swing.*;
```

2.3. PROPIEDADES Y MÉTODOS

Los objetos tienen propiedades, como son el color, el tamaño, la alineación, etc. Para modificar las propiedades utilizaremos los métodos, que son subrutinas que actúan sobre las propiedades y los datos:

```
miVentana.setSize(400, 100);
txtBienvenido.setText("Bienvenido a Java");
```

Los métodos utilizados en **Experimenta** son **setSize**, para establecer el tamaño de la ventana; **setText** para el contenido del cuadro de texto; **setFont**, para el tipo y tamaño de letra; **setForeground**, para el color del texto; **setBackground**, para el color de fondo. Algunos de estos métodos son comunes a varios objetos, como botones, cuadros de texto, etiquetas, etc. Existe un método especial, denominado **constructor**, que es el que se ejecuta cuando creamos el objeto, al utilizar el comando **new**. El **constructor** se denomina con el mismo nombre que la clase.

```
public VentanaBienvenido() {
```

Dentro del constructor de la ventana ordenamos la colocación de unos objetos en otros, mediante el método **add**.

2.4. FUNCIÓN MAIN()

La función **main** es la primera función que ejecuta una aplicación **Java**. Si no existe, el sistema operativo no puede ejecutar la aplicación. La función **main** tiene siempre la siguiente sintaxis:

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

En **Java** todas las instrucciones finalizan con el símbolo **;**; podemos escribir una instrucción en una sola línea o en varias, la instrucción no finaliza hasta que escribimos el carácter **,**. Para añadir comentarios en el código utilizamos los símbolos **//**

RESUELVE**1. Crea una nueva aplicación: 02Hola**

Realiza una aplicación que muestre la palabra **Hola** en el título de la ventana, que escriba tu nombre en el mensaje en color rojo (**RED**) sobre fondo naranja (**ORANGE**) y cuya ventana tenga de tamaño **500 x 200** pixels. Muestra el mensaje en la parte inferior ("South") y ejecútalo. Copia de la ventana **VentanaBienvenido.java** todos los trozos de código que necesites y haz los cambios oportunos.



EXPERIMENTA

APRENDE

3. Aplicaciones con un botón**1. CREA UNA APLICACIÓN CON UN BOTÓN: 03Avanzando**

Realiza una aplicación Java que al pulsar un botón, muestre en un cuadro de texto Avanzando con Java. Puedes copiar el código de otras aplicaciones anteriores y hacer los cambios oportunos; es una práctica muy extendida entre los programadores.

- Crea un nuevo Proyecto Java llamado **03Avanzando**. Crea una nueva clase **paquetejava** llamada **VentanaAvanzando**. Asegúrate de crearla en el proyecto **03Avanzando**.
- En el Explorador de paquetes selecciona **03Avanzando/PaqueteJava** y en su menú Contextual elige **Importar...**. Despliega General, selecciona **Sistema de archivos** y pulsa **[Avanzando]**. En el directorio pulsa **[VentanaAvanzando]**, selecciona tu carpeta **TIC/12Java/03Avanzando** y pulsa **[Abrir]**. En la lista de la derecha, activa **Aceptar.gif** y pulsa **[Abrir]**.
- Escribe el código de importación de paquetes **java.awt** y **javax.swing**.
- Define los objetos que vas a utilizar dentro de la clase **VentanaAvanzando**:

```
JFrame miVentana = new JFrame("Avanzando");
JPanel marMenaje = new JPanel();
JPanel marBotones = new JPanel();
JTextField txtMenaje = new JTextField(20);
JButton botAceptar = new JButton();
```

- Crea el constructor de la clase **VentanaNombre** con el siguiente código:

```
public VentanaAvanzando() {
    miVentana.setSize(400, 120);
    miVentana.setLayout(new BorderLayout(null));
    miVentana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    miVentana.setLayout(new BorderLayout());
    txtMenaje.setText("");
    txtMenaje.setFont(new Font("Arial", 0, 20));
    txtMenaje.setForeground(Color.BLUE);
    txtMenaje.setBackground(Color.YELLOW);
    txtMenaje.setHorizontalAlignment(JTextField.CENTER);
    botAceptar.setIcon(new ImageIcon(getClass().getResource("Aceptar.gif")));
    botAceptar.setText("Aceptar");
    botAceptar.setFont(new Font("Arial", 0, 14));
    botAceptar.setPreferredSize(new Dimension(120, 40));
    botAceptar.addActionListener(new Aceptar(txtMenaje));
    marBotones.add(txtMenaje);
    marBotones.add(botAceptar);
    miVentana.add("North", marMenaje);
    miVentana.add("South", marBotones);
    miVentana.setVisible(true);
}
```

- Condifica la función **main** con la siguiente instrucción:

```
new VentanaAvanzando();
```

- Haz clic sobre el icono situado a la izquierda del código y haz doble-clic en **Crear clase 'Aceptar'**. Pulsa **[ok]**. Completa el código:

```
package paquetejava;
import java.awt.event.*;
import javax.swing.*;
public class Aceptar implements ActionListener {
    JTextField objMenaje;
    public Aceptar(JTextField txtMenaje) {
        objMenaje = txtMenaje;
    }
    public void actionPerformed(ActionEvent arg0) {
        objMenaje.setText("Avanzando con Java");
        // TODO Apéndice de método generado automáticamente
    }
}
```

- En esta ventana, haz clic en **Guardar**, y en la ficha **VentanaAvanzando.java**.
- Ejecuta **VentanaAvanzando.java**.
- Pulsa el botón y cierra la aplicación.

3.1. ELEMENTOS GRÁFICOS

Para utilizar imágenes, sonidos y videos en nuestras aplicaciones, tenemos que importarlos. Para ello, seleccionamos el paquete y en su menú Contextual elegimos **Importar...**. Desplegamos **General/Sistema de archivos** y los importamos al proyecto.

3.2. MARCOS Y DISTRIBUCIÓN DE OBJETOS EN LA VENTANA

Al diseñar una aplicación, debemos prestar atención a la distribución de los objetos en la ventana. Los objetos se distribuyen utilizando marcos. Dentro de la ventana principal creamos uno o varios marcos; y dentro de estos, distribuimos el resto de objetos mediante otros marcos. Los objetos se colocan en un marco utilizando **add**. Observa el código:

```
marMenaje.add(txtMenaje);
miVentana.add("North", marMenaje);
miVentana.add("South", marBotones);
```

Al utilizar **add** los objetos se agrupan de izquierda a derecha. Para apilarlos en vertical, utilizamos los modificadores **North**, **Center** o **South**. Los objetos en los que vamos a utilizar estos modificadores, tienen que declararse en el constructor de la ventana:

```
miVentana.setLayout(new BorderLayout());
```

Conviene nombrar los objetos de la misma forma: **mar** para marcos, **bot** para botones, **eti** para etiquetas, etc.; seguidos de un nombre descriptivo.

3.3. BOTONES Y ACTIONLISTENER

Los botones son controles muy utilizados en la programación, útiles para realizar una tarea al completar los datos de una ventana. Los métodos más importantes son **setIcon** para mostrar una imagen en el botón, **setText** para mostrar texto, **setFont** para determinar el tipo de letra. Utilizamos **setPreferredSize** para establecer el tamaño del botón. Para controlar el **clic** sobre el ratón, utilizamos el método **addActionListener**, el cual invoca la llamada de una clase, en este caso **Aceptar**, y le pasa como parámetro los objetos que va a utilizar:

```
botAceptar.addActionListener(new Aceptar(txtMenaje));
```

La instrucción anterior espera a que se pulse el botón y ejecuta el constructor de la clase **Aceptar**. A este constructor enviamos el objeto **txtMenaje** como parámetro para modificar su contenido. En el código utilizamos un objeto de la misma clase, llamada **objTexto**.

```
public class Aceptar implements ActionListener {
    JTextField objMenaje;
    public Aceptar(JTextField txtMenaje) {
        objMenaje = txtMenaje;
    }
    public void actionPerformed(ActionEvent arg0) {
        objMenaje.setText("Avanzando con Java");
    }
}
```

RESUELVE

1. Aplicación con un botón: 03Nombre

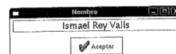
Crea una nueva aplicación llamada **03Nombre** con un cuadro de texto y un botón. Haz que al pulsar el botón aparezcan tu nombre y apellidos.

- Crea el proyecto **03Nombre** y la clase **VentanaNombre**. Puedes copiar partes del código del proyecto **03Avanzando** y modificar lo que sea necesario.

- Define un cuadro de texto **txtMenaje** y un botón **botNombre**.

- Realiza el código del constructor y de la función **main**.

- Crea el **ActionListener** del botón, con el nombre. Implementa la clase **Nombre** para mostrar tu nombre.



EXPERIMENTA

APRENDE

4. Aplicación con dos botones

1. CREA UNA APLICACIÓN CON DOS BOTONES: 04Centro

Realiza una aplicación Java con dos botones. Al pulsar el primero de ellos debe aparecer el nombre de tu centro de estudios y al pulsar el segundo, su localidad. Para el desarrollo de esta aplicación únicamente te indicamos los fragmentos de código que introducen nuevos conceptos. El resto has de programarlos.

- Crea un nuevo **Proyecto Java** llamado **04Centro**
- Crea el paquete **paqueteJava** y la clase **VentanaCentro**
- De tu carpeta **12Java/04Centro** importa las imágenes **Centro** y **Localidad**.

d) Escribe el código de importación de paquetes **java.awt** y **javax.swing**

e) Define los objetos que vas a utilizar dentro de la clase:

```
JPane miVentana = new JPanel("Centro de estudios");
JPanel marMensaje = new JPanel();
JPanel marBotones = new JPanel();
JTextField txtMensaje = new JTextField(20);
JButton botCentro = new JButton();
JButton botLocalidad = new JButton();
```

f) Crea el constructor de la clase **VentanaCentro**. El siguiente texto es únicamente para los botones, completa el resto del código. Puedes copiar fragmentos de otros proyectos. Observa los métodos **setIcon** y **addActionListener** de cada botón.

```
botCentro.setIcon(new ImageIcon(getClass().getResource("Centro.gif")));
botCentro.setText("Centro");
botCentro.setLayout(new Font("Arial", 0, 14));
botCentro.setPreferredSize(new Dimension(150, 40));
botCentro.addActionListener(new Centro(txtMensaje));
botLocalidad.setIcon(new ImageIcon(getClass().getResource("Localidad.gif")));
botLocalidad.setText("Localidad");
botLocalidad.setLayout(new Font("Arial", 0, 14));
botLocalidad.setPreferredSize(new Dimension(150, 40));
botLocalidad.addActionListener(new Localidad(txtMensaje));
```

g) Crea la composición de los objetos en la ventana y codifica la función **main()**

h) Haz clic sobre el icono  situado a la izquierda del código y haz doble-clic en  **Crear clase 'Centro'**. Pausa 

Escribe el código:

```
package paqueteJava;
import java.awt.event.*;
import javax.swing.*;
public class Centro implements ActionListener {
    JTextField objMensaje;
    public Centro(JTextField txtMensaje) {
        objMensaje = txtMensaje;
    }
    public void actionPerformed(ActionEvent e) {
        objMensaje.setText("Colegio Retamar");
    }
}
```

i) Vuelve al código de **VentanaCentro.java** con  y haz doble-clic en  **Crear clase 'Localidad'**. Pausa 

Codifica el código de esta clase. Puedes utilizar partes del código de la clase **Centro**. Haz que se muestre el nombre de la localidad.

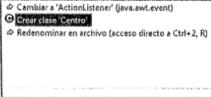
- Haz clic sobre  y corrige los posibles errores del código.
- Guarda el proyecto y ejecútalo.
- Pulsa los botones y cierra la aplicación.



4.1. CORRECCIÓN DE ERRORES

Cuando existen errores en el código, aparece el icono  en el margen izquierdo. Haciendo clic sobre él nos sugiere posibles correcciones; pueden ser por errores de sintaxis, por ausencia de importación de paquetes, por falta de implementación de clases, etc. Es conveniente prestar atención a todas las sugerencias.

Cuando existen errores en el proyecto, estos aparecen marcados en el **Explorador de proyectos**. Una vez corregidos todos los errores de los distintos .java, debemos pulsar  **Guardar** para eliminar los mensajes de error del proyecto. En ocasiones los errores de unas clases son provocados por errores en otras y al guardar el código quedan solucionados.



4.2. BOTONES Y ACTIONLISTENER

Cuando realizamos aplicaciones con varios botones, debemos escribir el código por separado de cada botón. En primer lugar debemos realizar su definición:

```
JButton botLocalidad = new JButton();
```

En el constructor debemos implementar los métodos necesarios, **setIcon** para mostrar la imagen, **setText** para el texto, **setFont** para el tipo de letra, **setPreferredSize** para el tamaño y **addActionListener** para indicar la clase invocada al hacer clic sobre él. Observa el código:

```
botLocalidad.setIcon(new ImageIcon(getClass().getResource("Localidad.gif")));
botLocalidad.setText("Localidad");
botLocalidad.setFont(new Font("Arial", 0, 14));
botLocalidad.setPreferredSize(new Dimension(150, 40));
botLocalidad.addActionListener(new Localidad(txtMensaje));
```

Por último, debemos especificar cada una de las clases. Observa la clase **Localidad**:

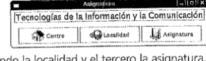
```
package paqueteJava;
import java.awt.event.*;
import javax.swing.*;
public class Localidad implements ActionListener {
    JTextField objMensaje;
    public Localidad(JTextField txtMensaje) {
        objMensaje = txtMensaje;
    }
    public void actionPerformed(ActionEvent e) {
        objMensaje.setText("Sevilla");
    }
}
```

RESUELVE

1. Aplicación con dos botones: 04Asignatura

Crea una nueva aplicación llamada **04Asignatura** con un cuadro de texto y tres botones. El primero muestra el nombre de tu centro de estudios, el segundo la localidad y el tercero la asignatura.

- Crea el proyecto **04Asignatura** y la clase **VentanaAsignatura**.
- Define los marcos, el cuadro de texto y los tres botones. Al tercero llámalo **botAsignatura**.
- Modifica el tamaño de la ventana a **500 x 120** y el cuadro de texto a **25**.
- Realiza el código del constructor y de la función **main**.
- Codifica los métodos **addActionListener**. Invoca las clases **Centro**, **Localidad** y **Asignatura**.
- Para el botón **Asignatura**, utiliza laImagen **Asignatura** de tu carpeta **12Java/04Asignatura**.



5. Aplicación Cambio de moneda

1. APPLICACIÓN: 05CambioMoneda

Crea una aplicación Java para convertir importes de euros a dólares. Para el desarrollo de esta aplicación únicamente te indicamos los fragmentos de código que introducen nuevos conceptos. El resto has de programarlos.

- Crea un nuevo **Proyecto Java** llamado **05CambioMoneda**
- Crea el paquete **paqueteJava** y la clase **VentanaCambioMoneda**
- De carpeta **12Java/05CambioMoneda**, importa la imagen **Derecha.gif**
- Escribe el código de importación de paquetes **java.awt** y **javax.swing**
- Define y programa los marcos necesarios, como en la imagen de la derecha.

- f) En la aplicación, observa las etiquetas **Euros (€)** y **Dólares**

(5). Las etiquetas se declaran con código:

```
JLabel etiEuros = new JLabel();
```

```
JLabel etiDolares = new JLabel();
```

- g) Crea el constructor de **VentanaCambioMoneda**. Crea la ventana del tamaño **400 x 100** píxeles. Para la etiqueta **etiEuros** utiliza el siguiente código.

```
etiquetas.setText("Euros (€)");  
etiquetas.setPoint(new Point("Arial", 0, 14));  
etiquetas.setForeground(Color.BLUE);  
etiquetas.setPreferedSize(new Dimension(80, 25));  
etiquetas.setHorizontalAlignment(javax.swing.JTextField.CENTER);  
etiquetas.setVerticalAlignment(javax.swing.JTextField.TOP);
```

De igual forma escribe el código para la etiqueta **etiDolares**

- h) Codifica el **ActionListener** del botón. Envía **txtEuros** y **txtDolares** como parámetros.

```
botCalcEuroDolar.addActionListener(new CalcEuroDolar(txtEuros, txtDolares));
```

- i) Crea la composición de los objetos en la ventana y codifica la función **main()**

j) En el margen izquierdo, haz clic sobre y corrige los posibles errores del código.

- k) Codifica la clase **CalcEuroDolar** con el código que aparece en líneas inferiores. Importa los paquetes necesarios.

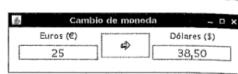
Debes modificar el valor 1.54 por el valor actual.

```
public class CalcEuroDolar implements ActionListener {  
    JTextField objEuros;  
    JTextField objDolares;  
    double valor;  
  
    public CalcEuroDolar(JTextField txtEuros, JTextField txtDolares) {  
        objEuros = txtEuros;  
        objDolares = txtDolares;  
    }  
  
    public void actionPerformed(ActionEvent arg0) {  
        try{  
            valor = Double.parseDouble(objEuros.getText().replace(',', '.'));  
            DecimalFormat miFormato = new DecimalFormat("#,##0.00");  
            objDolares.setText(miFormato.format((valor * 1.54)));  
        } catch(Exception e){  
            JOptionPane.showMessageDialog(null, "Introduce un valor correcto");  
        }  
    }  
}
```

- l) En el margen izquierdo, haz clic sobre y corrige los posibles errores del código. Algunos de ellos serán por falta de importación de paquetes de clases.

- m) Guarda el proyecto y ejecuta **VentanaCambioMoneda.java**

- n) Ejecuta la aplicación, introduce un importe en **Euros** y pulsa



5.1. ETIQUETAS

Las etiquetas son objetos que sirven para mostrar información en la aplicación. A diferencia del cuadro de texto, el contenido de la etiqueta no puede ser modificado mediante el teclado. Podemos utilizarlas como textos indicativos o para mostrar resultados de operaciones. Los métodos más importantes son **setText**, **setFont**, **setForeground**, **setBackground** y **setPreferredSize**. Con **setVerticalAlignment** y **setHorizontalAlignment** establecemos la alineación.

5.2. CONVERSIÓN A NUMÉRICOS Y SÍMBOLOS DECIMALES

Cuando realizamos operaciones con números, debemos tener en cuenta que Java utiliza el punto (.) como separador decimal. Es necesario convertir los números introducidos por el teclado en números con los que Java pueda trabajar. Adicionalmente, cuando aceptamos un número por el teclado, lo hacemos mediante cadenas de texto; y Java, para operar con ellos, necesita que sean de un tipo numérico, ya sea entero o decimal. Observa el código siguiente, antes de convertir el valor a numérico, reemplaza las comas por puntos.

```
valor = Double.parseDouble(objEuros.getText().replace(',', '.'));
```

Una vez realizada la operación debemos desarrollar los pasos a la inversa, es decir, pasar el dato numérico a texto y formatearlo utilizando la coma como símbolo decimal y dos decimales:

```
DecimalFormat miFormato = new DecimalFormat("#,##0.00");
```

```
objDolares.setText(miFormato.format((valor * 1.54))));
```

5.3. CONTROL DE ERRORES: try-catch

Utilizamos **try-catch** para controlar los errores de ejecución. Cuando se produce un error en el código comprendido dentro de **try...** finaliza la ejecución de ese código y ejecuta las instrucciones comprendidas dentro de **catch...**. En nuestra aplicación, esto ocurre cuando hemos introducido un valor erróneo o letras, y no se puede realizar la conversión numérica; situación que aprovechamos para mostrar un mensaje. Observa el código:

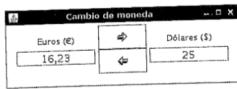
```
try{  
    valor = Double.parseDouble(objEuros.getText().replace(',', '.'));  
    DecimalFormat miFormato = new DecimalFormat("#,##0.00");  
    objDolares.setText(miFormato.format((valor * 1.54)));  
}  
catch(Exception e){  
    JOptionPane.showMessageDialog(null, "Introduce un valor correcto");  
}
```

RESUELVE

1. Aplicación: 05CambioMoneda2

Crea una aplicación Java para convertir importes de euros a dólares y viceversa. Puedes copiar fragmentos de código del proyecto **05CambioMoneda**.

- Introduce los controles necesarios como marcos, etiquetas, cuadros de texto, etc.; como en la imagen adjunta.
- De carpeta **12Java/05CambioMoneda2** utiliza las imágenes **Derecha.gif** e **Izquierda.gif**.
- Crea dos botones, **botCalcEuroDolar** y **botCalcDolarEuro**.
- Codifica el método **addActionListener** de los dos botones.
- Codifica las clases **CalcEuroDolar** y **CalcDolarEuro**.
- Controla los posibles errores del código.



6. Aplicación Concentración

1. APLICACIÓN: 06Concentración

Crea una aplicación Java para calcular la concentración de una disolución según los gramos de soluto y el volumen de la disolución. Para la codificación únicamente te indicamos los nuevos conceptos. El resto los programarás.

- a) Crea un proyecto llamado **06Concentración** y la clase **VentanaConcentración**. De tu carpeta **12Java/06Concentración** importa las imágenes **Calcular**, **Concentración** y **Formula**.

- b) Escribe el código de importación de paquetes **awt y swing**.
c) Define los marcos necesarios, las etiquetas, los cuadros de texto y los botones. Para mostrar el resultado utiliza también una etiqueta, llámala **etlResultado**
d) Las imágenes se declaran como etiquetas. Utiliza el siguiente código:



- e) Codifica el constructor de la clase **VentanaConcentración**. El tamaño que hemos utilizado para la ventana es **500 x 250** pixels. Codifica las etiquetas, cuadros de texto, botones, etc. Para la imagen **imgFormula** utiliza el código siguiente:

```
imgFormula.setIcon(new ImageIcon(getClass().getResource("Formula.gif")));
```

```
imgFormula.setPreferredSize(new Dimension(90, 30));
```

```
imgFormula.setVerticalAlignment(JLabel.CENTER);
```

```
f) De igual manera, introduce el código para la imagen imgConcentracion, tamaño 140 x 120
```

- g) Programa la etiqueta **etlResultado** con el siguiente código:

```
etlResultado.setText("");
```

```
etlResultado.setFont(new Font("Arial", 1, 14));
```

```
etlResultado.setForeground(Color.BLUE);
```

```
etlResultado.setBackground(Color.ORANGE);
```

```
etlResultado.setHorizontalAlignment(JLabel.CENTER);
```

```
etlResultado.setPreferredWidth(new Dimension(100, 30));
```

```
etlResultado.setOpaque(true);
```

- h) Codifica el método **addActionListener** del botón y llama a la clase **Calcular**. Envía los parámetros **txtSoluto**, **txtDisolucion** y **etlConcentracion** para modificar su valor.

```
botCalcular.addActionListener(new Calcular(txtSoluto, txtDisolucion, etlResultado));
```

- i) Crea la composición de los objetos en la ventana y codifica la función **main()**

- j) Codifica la clase **Calcular**. Añade el siguiente código. Tienes que añadir la importación de paquetes y definir los objetos utilizados y las variables **soluto** y **disolucion** del tipo **double**.

```
public Calcular(JTextField txtSoluto, JTextField txtDisolucion, JLabel lblConcentracion) {
    objSoluto = txtSoluto;
    objDisolucion = txtDisolucion;
    objConcentracion = lblConcentracion;
}

public void actionPerformed(ActionEvent e) {
    try{
        soluto = Double.parseDouble(objSoluto.getText().replace(',', '.'));
        disolucion = Double.parseDouble(objDisolucion.getText().replace(',', '.'));

        catch (Exception e1){
            JOptionPane.showMessageDialog(null, "Introduce valores numéricos");
            return;
        }
        if (soluto <= 0){
            JOptionPane.showMessageDialog(null, "Soluto: escribe un valor positivo");
            return;
        }
        if (disolucion <= 0){
            JOptionPane.showMessageDialog(null, "Disolucion: escribe un valor positivo");
            return;
        }
        DecimalFormat miFormato = new DecimalFormat("#,##0.00");
        objConcentracion.setText(miFormato.format((soluto/disolucion)));
    }
}
```

- k) Corrige los posibles errores. Guarda y ejecuta el proyecto.

6.1. ETIQUETAS

Además de utilizar las etiquetas como texto indicativo del contenido introducido en cuadros de texto, las utilizamos para mostrar resultados, ya que en ellas podemos introducir datos. Cuando utilizamos una etiqueta para este fin, programaremos el método **setOpaque**. Observa el código:

```
etlResultado.setText("");
etlResultado.setFont(new Font("Arial", 1, 14));
etlResultado.setForeground(Color.BLUE);
etlResultado.setBackground(Color.ORANGE);
etlResultado.setHorizontalAlignment(JLabel.CENTER);
etlResultado.setPreferredWidth(new Dimension(100, 30));
etlResultado.setOpaque(true);
```

6.2. COLOR Y TIPO DE FUENTE

Podemos variar los colores en nuestras aplicaciones mediante el objeto **Color**. Utilizamos las siguientes variantes: **Color.BLACK**, **Color.DARK_GRAY**, **Color.GRAY**, **Color.LIGHT_GRAY**, **Color.WHITE**, **Color.MAGENTA**, **Color.RED**, **Color.PINK**, **Color.ORANGE**, **Color.YELLOW**, **Color.GREEN**, **Color.CYAN** y **Color.BLUE**. Su utilización en el código es el siguiente:

```
botCalcular.setBackground(Color.MAGENTA);
```

Para definir un tipo de fuente, utilizamos el siguiente código, donde indicamos el tipo de fuente, si es o no negrita, y el tamaño:

```
txtSoluto.setFont(new Font("Arial", 0, 17));
```

Si vamos a utilizar varias veces el mismo tipo de fuente en una aplicación es conveniente definir una variable con el tipo de fuente para usarla a continuación donde sea necesario:

```
Font fuente = new Font("Helvetica", 0, 14);
```

```
txtSoluto.setFont(fuente);
```

6.3. TIPOS DE DATOS Y OPERADORES ARITMÉTICOS

Para realizar operaciones matemáticas, debemos trabajar con tipos numéricos. Existen distintos tipos, en función del número que van a almacenar. Son los siguientes:

Operadores aritméticos

byte	-128 a +127	+	Suma y resta
short	-32 768 a 32 767	*	Multiplicación y división
int	-2 147 483 648 a 2 147 483 648	%	Resto de la división
float	+/- 3,4 x 10 ³⁸	Math.pow(a, b)	Devuelve a^b
double	+/- 3,4 x 10 ³⁰⁸	Math.sqrt(x)	Devuelve la raíz cuadrada de x

6.4. INSTRUCCIÓN IF

Utilizamos **if** para tomar decisiones dentro de la aplicación. Sirve para decidir si ejecutamos, o no, un bloque de código, en función de la veracidad de una condición. En el código siguiente, evaluamos el valor introducido en el soluto, si es igual o inferior a cero, mostramos un mensaje:

```
if (soluto <= 0){
    JOptionPane.showMessageDialog(null, "Escribe un valor positivo!");
    return;
}
```

La instrucción **return**, abandona la ejecución de **Calcular** y la devuelve a la función **main()**

RESUELVE

1. Aplicación: 06Raíz

Crea una aplicación para calcular la raíz cuadrada de un número. Si el número es menor que **0**, indica que no tiene raíz real. Al pulsar el botón, si no se ha introducido ningún número muestra un mensaje de error. Redondea el resultado con dos decimales. Utiliza las imágenes de tu carpeta **12Java/06Raíz**

7. Aplicación Dividir

1. APLICACIÓN DIVIDIR: 07Dividir

Crea una aplicación Java para calcular el cociente de dos números. Si el dividendo y el divisor son 0, el resultado es **Indeterminado**; si el divisor es 0 y el dividendo no, el resultado es **Infinito**. Redondea el resultado en dos decimales.

- Define el proyecto llamado **07Dividir** y la clase **VentanaDividir**. De tu carpeta **12Java/07Dividir** incorpora la imagen **Calcular**.
- Escribe el código de importación de paquetes **awt** y **swing**. Define los marcos necesarios, las etiquetas, los cuadros de texto y los botones. Utiliza una etiqueta para mostrar el resultado.
- Para definir un marco con borde y título, como **Operandos**, define el marco y define una fuente, un borde y un título. Observa el código:



```
 JPanel marOperandos = new JPanel();
 Font fuente = new Font("Arial", 0, 14);
 Border linea = BorderFactory.createLineBorder(Color.GRAY);
 TitledBorder titOperando = BorderFactory.createTitledBorder(linea,
 "Operando", TitledBorder.LEFT, TitledBorder.TOP, fuente, Color.RED);
```

d) Codifica el constructor de la clase **VentanaDividir**. El tamaño que hemos utilizado para la ventana es **310 x 175** pixels. Codifica las etiquetas, cuadros de texto, botones, etc. Para añadir el borde y el título al marco, utiliza el siguiente código en el constructor:

```
 marOperando.setBorder(titOperando);
```

e) Codifica el método **addActionListener** del botón y llama a la clase **Calcular**. Envía los parámetros **txtDividendo**, **txtDivisor** y **etarResultado**, para modificar su valor.

f) Crea la composición de los objetos en la ventana y codifica la función **main()**

g) Codifica la clase **Calcular**. Añade el siguiente código. Define las variables **dividendo** y **divisor** del tipo **double**.

```
 public class Calcular extends JPanel {
 JTextField txtDividendo, txtDivisor;
 JLabel etiResultado;
 JButton btnCalcular;
 
 public Calcular(JTextField txtDividendo, JTextField txtDivisor, JLabel etiResultado) {
     objDividendo = txtDividendo;
     objDivisor = txtDivisor;
     objResultado = etiResultado;
 }
 
 public void actionPerformed(ActionEvent arg0) {
     dividendo = Integer.parseInt(objDividendo.getText().replace(',', '.'));
     divisor = Integer.parseInt(objDivisor.getText().replace(',', '.'));
     if (divisor == 0) {
         if (dividendo == 0)
             objResultado.setText("Indeterminado");
         else
             objResultado.setText("Infinito");
     } else {
         DecimalFormat miFormato = new DecimalFormat("#,##0.00");
         objResultado.setText(miFormato.format((dividendo/divisor)));
     }
 }
 }
```

h) Antes de la instrucción **try**, incluye el código para comprobar si el cuadro de texto **txtDividendo** está vacío.

```
 if (objDividendo.getText().equals("")) {
     JOptionPane.showMessageDialog(null, "Introduce el dividendo");
     return;
 }
```

i) Corrige los posibles errores de código.

j) Guarda y ejecuta el proyecto.

7.1. TÍTULOS Y BORDES

Utilizamos marcos para agrupar y destacar algunos objetos. Para ello definimos el marco, la línea, el tipo de letra y, si es necesario, el color. Para definirlos utilizamos el siguiente código:

```
JPanel marOperandos = new JPanel();
Font fuente = new Font("Arial", 0, 14);
Border linea = BorderFactory.createLineBorder(Color.GRAY);
TitledBorder titOperando = BorderFactory.createTitledBorder(linea,
"Operando", TitledBorder.LEFT, TitledBorder.TOP, fuente, Color.RED);
```

Para establecer el título y la línea del marco en el constructor, utilizamos el siguiente código:

```
marOperando.setBorder(titOperando);
```

7.2. CARACTERES Y CADENAS DE CARACTERES

En Java, además de los tipos básicos numéricos existe el tipo **char**, que sirve para almacenar un carácter. Para almacenar cadenas de caracteres utilizamos objetos de la clase **String**. También podemos utilizar matrices de cadenas. Observa la definición de cada una de ellas:

```
char caracter = 'a';
String nombre = "Me llamo Ismael Rey";
```

Para definir caracteres usamos comillas simples ''; y para cadenas utilizamos comillas dobles "".

7.3. EXPRESIONES Y CONDICIONES

Una **expresión** está formada por operandos (datos) y operadores (operaciones).

Una **asignación** es una expresión para realizar un cálculo o manipular caracteres.

```
i = 0;
i + j - 10;
nombre = "Me llamo Ismael Rey";
```

Una **condición** es una expresión que se evalúa como verdadera o falsa.

```
num > 0;
nombre == "Sonia";
```

7.4. IF ... ELSE E IF ANIDADOS

Utilizamos **if ... else** cuando queremos ejecutar códigos diferentes según el resultado de la expresión que se evalúa. Si es verdadero, se ejecuta el código entre **if** y **else**; si resulta falso, se ejecuta el código a continuación del **else**. Es posible anidar condiciones. Observa el ejemplo.

```
if (num > 0) {
    obj.setText("num es mayor que 0");
} else {
    if (num < 0) {
        obj.setText("num es menor que 0");
    } else {
        obj.setText("num es igual a 0");
    }
}
```

1 Crea una nueva aplicación: 07NIF

Crea una aplicación para calcular la letra del **NIF**. Divide el número del **DNI** entre **23**; si el resto es **0**, la letra es **T**; si es **1**, la letra es **R**, etc., según la lista: **TRWAGMYFPDXBNJZSQVHLCKET**. Utiliza el siguiente código:

```
try{
    String letrasNIF = "TRWAGMYFPDXBNJZSQVHLCKET";
    int dni = Integer.parseInt(objDNI.getText());
    int pos = dni % 23;
    objNIF.setText(objDNI.getText() + '-' + letrasNIF.charAt(pos));
}
```



RESUELVE

EXPERIMENTA

APRENDE

8. Aplicación Poliedros

1. APLICACIÓN: 08Poliedros

Crea una aplicación Java para calcular el área de los poliedros regulares. Utiliza una lista desplegable para seleccionar el tipo de poliedro y muestra la imagen correspondiente.

- a) Crea el proyecto **Poliedros** y la clase **VentanaPoliedros**. De tu carpeta **12Java/08Poliedros** importa las imágenes existentes.
 - b) Escribe el código de importación de paquetes **awt** y **swing**. Define los marcos necesarios, fuentes, títulos, etiquetas, cuadros de texto y botones. Muestra el resultado con una etiqueta.
 - c) Para definir la lista desplegable o **ComboBox**, utiliza el código siguiente:
- ```
String[] lista = {"Piramíde", "Cubo o hexaedro", "Octaedro", "Dodecaedro", "Icosaedro"};
JComboBox cmbPoliedro = new JComboBox(lista);
```

- d) Codifica el constructor de la clase **VentanaDividir**. El tamaño que hemos utilizado para la ventana es **380 x 315** píxeles. Codifica las etiquetas, cuadros de texto, botones, etc.
- e) Codifica las características de la lista desplegable y el método **addActionListener**. Utiliza el siguiente código:

```
cmbPoliedro.setEditable(true);
cmbPoliedro.setBackground(Color.YELLOW);
cmbPoliedro.setForeground(Color.BLUE);
cmbPoliedro.addActionListener(new Seleccionar(cmbPoliedro, imgPoliedro));
```

- f) Codifica el método **addActionListener** del botón y llama a la clase **Calcular**.
- botoncular.addActionListener(new Calcular(cmbPoliedro, txtArista, etiResultado));

- g) Crea la composición de los objetos en la ventana y codifica la función **main()**
- h) Codifica la clase **Seleccionar** con el siguiente código. Importa los paquetes necesarios y define las variables que se utilizan.

```
public void actionPerformed(ActionEvent arg0) {
 String seleccion = objComboBox.getSelectedItem() + ".gif";
 objImagen.setIcon(new ImageIcon(getClass().getResource(seleccion)));
}
```

- i) Codifica la clase **Calcular** con el siguiente código, importa los paquetes y define las variables. Añade el código que falta.

```
public void actionPerformed(ActionEvent arg0) {
 try{
 arista = Double.parseDouble(objArista.getText().replace(',','.'));

 switch (objComboBox.getSelectedIndex()+1){
 case 1:
 area = (Math.pow(arista,2) * Math.sqrt(3));
 break;
 case 2:
 area = (6 * Math.pow(arista,2));
 break;
 case 3:
 area = (2 * Math.pow(arista,2) * Math.sqrt(3));
 break;
 case 4:
 area = (5 * Math.pow(arista,2) * Math.sqrt(3));
 break;
 case 5:
 area = (3 * Math.pow(arista,2) * Math.sqrt(25 + 10 * Math.sqrt(5)));
 break;
 }
 DecimalFormat miFormato = new DecimalFormat("#,##0.00");
 objResultado.setText(miFormato.format((area)));
 }
 catch(Exception e1){
 JOptionPane.showMessageDialog(null, "Introduce una arista válida");
 }
}
```

- j) Añade el código para comprobar que se ha introducido la arista y que es positiva.
- k) Guarda y ejecuta el proyecto.



### 8.1. LISTAS DESPLEGABLES

La lista desplegable, también llamada cuadro combinado o **ComboBox**, es un objeto que permite seleccionar una lista de un conjunto fijo de opciones. Es un objeto muy utilizado en la programación. Para definir un **ComboBox** y llenarlo de opciones utilizamos el siguiente código:

```
String[] lista = {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes"};
JComboBox cmbPoliedro = new JComboBox(lista);
```

Las propiedades más importantes del **ComboBox** son:

- **getSelectedIndex()**: contiene el índice del elemento seleccionado, comenzando por 0. Recuerda que el primer elemento es el 0
- **getSelectedItem()**: contiene el texto de la opción seleccionada.

En nuestra aplicación, aprovechamos que las imágenes mostradas se llaman igual que las opciones de la lista, añadiéndoles la extensión del archivo **.gif**. Utilizamos el siguiente código:

```
String seleccion = objComboBox.getSelectedItem() + ".gif";
objImagen.setIcon(new ImageIcon(getClass().getResource(seleccion)));
```

### 8.2. ESTRUCTURA DE CONTROL: switch

La instrucción **switch** permite ejecutar diferentes bloques en función del resultado de una expresión:

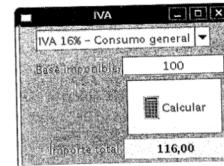
```
switch (opcion){
 case 1:
 area = (Math.pow(arista,2) * Math.sqrt(3));
 break;
 case 2:
 area = (6 * Math.pow(arista,2));
 break;
 case 3:
 area = (2 * Math.pow(arista,2) * Math.sqrt(3));
 break;
 case 4:
 area = (5 * Math.pow(arista,2) * Math.sqrt(3));
 break;
 case 5:
 area = (3 * Math.pow(arista,2) * Math.sqrt(25 + 10 * Math.sqrt(5)));
 break;
}
```

La instrucción **break** es opcional y fuerza que la ejecución se reduzca a continuación del **switch**.

## RESUELVE

### 1. Crea una nueva aplicación: 08IVA

Crea una aplicación para calcular el importe de una factura, en función del **IVA** aplicado a la base imponible, según se trate de **Libros (4%)**, **Alimentación (7%)** o **Consumo general (16%)**. Al pulsar el botón, si no se ha introducido ningún número, muestra un mensaje de error. Redondea el resultado con dos decimales. Utiliza las imágenes de tu carpeta **12Java/08IVA**



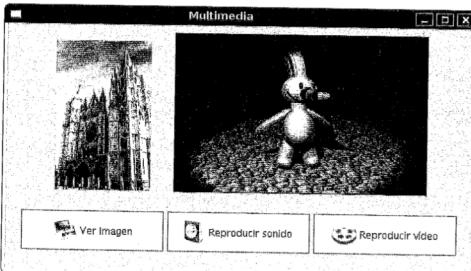
## CURIOSIDADES

## TALLER DE INVESTIGACIÓN

## MULTIMEDIA: 09Multimedia

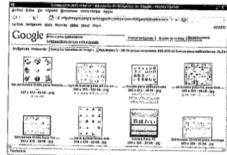
Importa el proyecto **09Multimedia** a tu espacio de trabajo.

- Inicia **Eclipse**.
- Elige **Archivo/Importar...**, selecciona **General/Proyectos existentes en el espacio de trabajo** y pulsa **Siguiente >**
- Pulsa **Explorar...**, busca y selecciona tu carpeta **TIC/12Java** y pulsa **Avanzar**.
- Selecciona el proyecto **09Multimedia** y pulsa **Avanzar**.
- En el **Explorador de paquetes**, selecciona el proyecto y en el menú **Contextual** elige **Propiedades**, selecciona **Vía de construcción Java**, elige la ficha **Bibliotecas**, pulsa **Añadir jar externo...**, selecciona la carpeta **09Multimedia/libJMF** y selecciona los archivos **customizer.jar**, **jmf.jar**, **mediaplayer.jar** y **multplayer.jar**. Pulsa **Avanzar** y **Finalizar**.
- En el **Explorador de paquetes**, despliega **09Multimedia/paqueteJava** y selecciona **VentanaMultimedia.java**. En su menú **Contextual** elige **Ejecutar como/1 Aplicación Java**.



## ICONOS PARA APLICACIONES

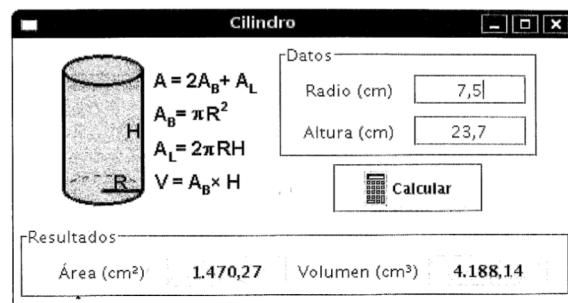
En las aplicaciones que hemos desarrollado hemos utilizado iconos para los botones. Si deseámos encontrar más opciones podemos realizar una búsqueda en **Internet**, a través de **Google**. Una vez descargadas las imágenes se pueden recortar con **GIMP**.



## CREA UNA APLICACIÓN EJECUTABLE .JAR: 10Cilindro y Cilindro.jar

Importa el proyecto **10Cilindro** a tu espacio de trabajo.

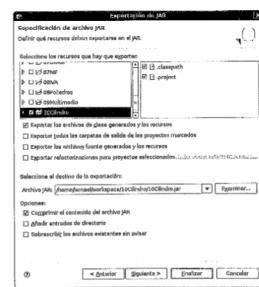
- Inicia **Eclipse**.
- Elige **Archivo/Importar...** e importa el proyecto **10Cilindro** a tu espacio de trabajo.
- Ejecuta **10Cilindro/paqueteJava/VentanaCilindro.java**.



Hasta ahora hemos ejecutado las aplicaciones utilizando el entorno de trabajo de **Eclipse**.

Vamos a construir una aplicación ejecutable, que se pueda exportar a cualquier equipo de cualquier plataforma, ya sea **Windows** o **Linux** y ejecutarse.

- En el **Explorador de paquetes** selecciona el proyecto **10Cilindro** y en su menú **Contextual** elige **Exportar...**
- Elige **Java/Archivo JAR**, pulsa **Siguiente >**
- En **Archivo JAR** pulsa **Navegar...**, en **Nombre** escribe **Cilindro.jar**, selecciona tu carpeta **12Java/10Cilindro**, pulsa **Avanzar** y **Finalizar**.
- Pulsa **Avanzar** y **Finalizar**.
- En la parte inferior de la ventana, en **Class principal** pulsa **Navegar...** y selecciona la clase **VentanaCilindro** y pulsa **Avanzar**.
- Pulsa **Finalizar**.
- Cierra **Eclipse**.
- De tu carpeta **12Java/10Cilindro** haz doble-clic sobre el archivo **Cilindro.jar**.
- Pruéba a ejecutar esta aplicación en otros equipos, incluso en plataformas **Windows**.



## JAVA E INTERNET

### APPLETS: Prisma.html

Un applet de Java es una aplicación Java con las siguientes características:

- Puede incluirse dentro de una página web y ejecutarse desde el navegador de Internet.
- Sus características de seguridad impiden que se pueda escribir en el disco duro del ordenador donde se ejecuta, generando un alto nivel de confianza al ejecutarla desde Internet.
- Son aplicaciones multiplataforma, con lo que puede ejecutarse en Windows, Linux, Mac; y con Firefox, Internet Explorer o cualquier otro navegador.

Ejecuta un applet para calcular el área y el volumen de un prisma.

a) Abre la carpeta TIC/12java/11Prisma y haz doble-clic en la página web 11Prisma.htm



b) Prueba con distintos valores, incluso números negativos y textos, para ver los mensajes.

### DESARROLLO DE APPLETS

Inicia Eclipse e importa el proyecto 11Prisma. Observa las diferencias en la programación:

- Hemos denominado el paquete con el nombre PaqueteApplet.
- Observa la definición de la clase VentanaPrisma, que extiende la clase JApplet.

```
public class VentanaPrisma extends JApplet {
 • El applet se inicia en el navegador, no es necesario definir objetos de la clase JApplet. Al final del constructor, añadimos el marco principal al navegador, con el siguiente código:
 Container navegador = this.getContentPane();
 navegador.add(marPrincipal);
 • El applet no utiliza la función main(). En su lugar utiliza una función equivalente, llamada init(), que es la que busca y ejecuta el navegador para iniciar la aplicación:
 public void init() {
 try {
 new VentanaPrisma();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 • Para invocar el applet desde el navegador, en cualquier página web, que debe estar en la misma carpeta, utilizamos el siguiente código HTML:
 <applet code=paqueteApplet.VentanaPrisma.class width="600" height="200" >
 </applet>
```

### TABLA DEL CÓDIGO ASCII

Para obtener un carácter cualquiera, mantenemos pulsada la tecla [Alt] y tecleamos en el teclado numérico de la derecha el número correspondiente a su código ASCII.

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| 0 =     | 1 = ①   | 2 = ②   | 3 = ③   | 4 = ④   | 5 = ⑤   | 6 = ⑥   | 7 = ⑦   |
| 8 = ⑧   | 9 = ⑨   | 10 = ⑩  | 11 = ⑪  | 12 = ⑫  | 13 = ⑬  | 14 = ⑭  | 15 = ⑮  |
| 16 = ⑯  | 17 = ⑯  | 18 = ⑯  | 19 = ⑯  | 20 = ⑯  | 21 = ⑯  | 22 = ⑯  | 23 = ⑯  |
| 24 = ⑯  | 25 = ⑯  | 26 = ⑯  | 27 = ⑯  | 28 = ⑯  | 29 = ⑯  | 30 = ⑯  | 31 = ⑯  |
| 32 = ⑯  | 33 = ⑯  | 34 = ⑯  | 35 = ⑯  | 36 = ⑯  | 37 = ⑯  | 38 = ⑯  | 39 = ⑯  |
| 40 = ⑯  | 41 = ⑯  | 42 = ⑯  | 43 = ⑯  | 44 = ⑯  | 45 = ⑯  | 46 = ⑯  | 47 = ⑯  |
| 48 = ⑯  | 49 = ⑯  | 50 = ⑯  | 51 = ⑯  | 52 = ⑯  | 53 = ⑯  | 54 = ⑯  | 55 = ⑯  |
| 56 = ⑯  | 57 = ⑯  | 58 = ⑯  | 59 = ⑯  | 60 = ⑯  | 61 = ⑯  | 62 = ⑯  | 63 = ⑯  |
| 64 = ⑯  | 65 = ⑯  | 66 = ⑯  | 67 = ⑯  | 68 = ⑯  | 69 = ⑯  | 70 = ⑯  | 71 = ⑯  |
| 72 = ⑯  | 73 = ⑯  | 74 = ⑯  | 75 = ⑯  | 76 = ⑯  | 77 = ⑯  | 78 = ⑯  | 79 = ⑯  |
| 80 = ⑯  | 81 = ⑯  | 82 = ⑯  | 83 = ⑯  | 84 = ⑯  | 85 = ⑯  | 86 = ⑯  | 87 = ⑯  |
| 88 = ⑯  | 89 = ⑯  | 90 = ⑯  | 91 = ⑯  | 92 = ⑯  | 93 = ⑯  | 94 = ⑯  | 95 = ⑯  |
| 96 = ⑯  | 97 = ⑯  | 98 = ⑯  | 99 = ⑯  | 100 = ⑯ | 101 = ⑯ | 102 = ⑯ | 103 = ⑯ |
| 104 = ⑯ | 105 = ⑯ | 106 = ⑯ | 107 = ⑯ | 108 = ⑯ | 109 = ⑯ | 110 = ⑯ | 111 = ⑯ |
| 112 = ⑯ | 113 = ⑯ | 114 = ⑯ | 115 = ⑯ | 116 = ⑯ | 117 = ⑯ | 118 = ⑯ | 119 = ⑯ |
| 120 = ⑯ | 121 = ⑯ | 122 = ⑯ | 123 = ⑯ | 124 = ⑯ | 125 = ⑯ | 126 = ⑯ | 127 = ⑯ |
| 128 = ⑯ | 129 = ⑯ | 130 = ⑯ | 131 = ⑯ | 132 = ⑯ | 133 = ⑯ | 134 = ⑯ | 135 = ⑯ |
| 136 = ⑯ | 137 = ⑯ | 138 = ⑯ | 139 = ⑯ | 140 = ⑯ | 141 = ⑯ | 142 = ⑯ | 143 = ⑯ |
| 144 = ⑯ | 145 = ⑯ | 146 = ⑯ | 147 = ⑯ | 148 = ⑯ | 149 = ⑯ | 150 = ⑯ | 151 = ⑯ |
| 152 = ⑯ | 153 = ⑯ | 154 = ⑯ | 155 = ⑯ | 156 = ⑯ | 157 = ⑯ | 158 = ⑯ | 159 = ⑯ |
| 160 = ⑯ | 161 = ⑯ | 162 = ⑯ | 163 = ⑯ | 164 = ⑯ | 165 = ⑯ | 166 = ⑯ | 167 = ⑯ |
| 168 = ⑯ | 169 = ⑯ | 170 = ⑯ | 171 = ⑯ | 172 = ⑯ | 173 = ⑯ | 174 = ⑯ | 175 = ⑯ |
| 176 = ⑯ | 177 = ⑯ | 178 = ⑯ | 179 = ⑯ | 180 = ⑯ | 181 = ⑯ | 182 = ⑯ | 183 = ⑯ |
| 184 = ⑯ | 185 = ⑯ | 186 = ⑯ | 187 = ⑯ | 188 = ⑯ | 189 = ⑯ | 190 = ⑯ | 191 = ⑯ |
| 192 = ⑯ | 193 = ⑯ | 194 = ⑯ | 195 = ⑯ | 196 = ⑯ | 197 = ⑯ | 198 = ⑯ | 199 = ⑯ |
| 200 = ⑯ | 201 = ⑯ | 202 = ⑯ | 203 = ⑯ | 204 = ⑯ | 205 = ⑯ | 206 = ⑯ | 207 = ⑯ |
| 208 = ⑯ | 209 = ⑯ | 210 = ⑯ | 211 = ⑯ | 212 = ⑯ | 213 = ⑯ | 214 = ⑯ | 215 = ⑯ |
| 216 = ⑯ | 217 = ⑯ | 218 = ⑯ | 219 = ⑯ | 220 = ⑯ | 221 = ⑯ | 222 = ⑯ | 223 = ⑯ |
| 224 = ⑯ | 225 = ⑯ | 226 = ⑯ | 227 = ⑯ | 228 = ⑯ | 229 = ⑯ | 230 = ⑯ | 231 = ⑯ |
| 232 = ⑯ | 233 = ⑯ | 234 = ⑯ | 235 = ⑯ | 236 = ⑯ | 237 = ⑯ | 238 = ⑯ | 239 = ⑯ |
| 240 = ⑯ | 241 = ⑯ | 242 = ⑯ | 243 = ⑯ | 244 = ⑯ | 245 = ⑯ | 246 = ⑯ | 247 = ⑯ |
| 248 = ⑯ | 249 = ⑯ | 250 = ⑯ | 251 = ⑯ | 252 = ⑯ | 253 = ⑯ | 254 = ⑯ | 255 = ⑯ |