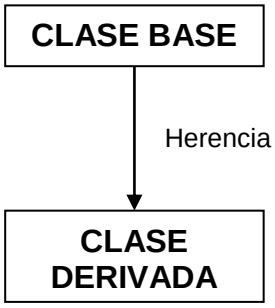
EJERCICIO GUIADO. JAVA: POO. HERENCIA

Concepto de Herencia

El concepto de Herencia consiste en crear una nueva Clase a partir de otra. La nueva Clase es una *mejora* de la anterior. O dicho de otra manera, es una *especialización* de la primera Clase.

Concretamente, la Herencia consiste en tomar una Clase inicial, y, a partir de ella, crear otra Clase que posee las mismas propiedades y métodos que la Clase inicial, además de unas nuevas propiedades y métodos que la Clase inicial no poseía. La nueva Clase creada puede incluso redefinir métodos de la Clase inicial.



La Clase inicial se denomina *Clase Base*, la Clase creada a partir de la clase base se denomina *Clase Derivada*. La *Clase Derivada* contiene propiedades y métodos de la *Clase Base* más unas propiedades y métodos añadidos.

La Herencia es una técnica muy útil que nos permite reutilizar código, es decir, que nos permite usar de nuevo lo que ya teníamos programado añadiendo simplemente algunos cambios adecuar el código al proyecto actual.

La Herencia se puede aplicar tanto a Clases Propias, como a Clases propias del lenguaje de programación Java.

En esta explicación guiada, veremos un ejemplo de uso de la Herencia con clases propias del lenguaje Java.

EJERCICIO GUIADO

Planteamiento Inicial

"Botones Contadores"

Supongamos que en los proyectos cotidianos se plantea la necesidad de usar botones que guarden el número de veces que son pulsados. Estos botones funcionarían exactamente igual que los botones normales (JButton) y tendrían su mismo aspecto, pero sería interesante que además tuvieran los siguientes métodos:

Método *setPulsaciones*.
Permitirá asignar un número de pulsaciones al botón.

```
btnContar.setPulsaciones(3);
//Se asignan 3 pulsaciones al botón btnContar
```

Método *getPulsaciones.*
Permitirá saber cuantas pulsaciones tiene un botón.

```
int p = btnContar.getPulsaciones(3);
//Se recoge el número de pulsaciones del botón btnContar
//en la variable p
```

Método *incrementa*.
Permite incrementar en uno las pulsaciones que tiene un botón.

```
btnContar.incrementa();
//Incrementa en uno las pulsaciones del botón btnContar
```

Método *decrementa.*
Permite decrementar en uno las pulsaciones que tiene un botón

```
btnContar.decrementa();
//Decrementa en uno las pulsaciones del botón btnContar
```

Método *reiniciar.*
Permite colocar las pulsaciones de un botón a cero.

```
btnContar.reiniciar();
//Sitúa a cero las pulsaciones del botón btnContar
```

Método *aumentar.*
Permite aumentar en una determinada cantidad las pulsaciones del botón.

```
btnContar.aumentar(4);
//Aumenta en cuatro las pulsaciones del botón btnContar
```

Método *disminuir.*
Permite disminuir en una determinada cantidad las pulsaciones del botón.

```
btnContar.disminuir(6);
//Disminuye en 6 las pulsaciones del botón btnContar
```

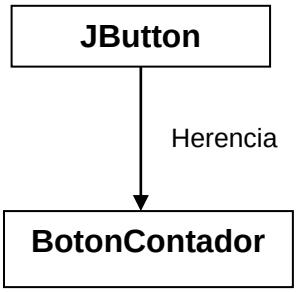
Los botones de este tipo tendrían una propiedad llamada *pulsaciones* de tipo entera que se inicializaría a cero en el momento de construir el botón.

Desgraciadamente **no existe un botón de este tipo en Java**, así que no podemos acudir a la ventana de diseño y añadir un botón como este.

Una posibilidad sería la de programar la clase correspondiente a un botón de este tipo desde cero, pero esto es un trabajo tremendamente complejo.

La solución a este problema es la Herencia. Básicamente, lo que necesitamos es *mejorar* la Clase JButton, la cual define botones normales y corrientes, de forma que estos botones admitan también los métodos indicados antes.

La idea es crear una nueva clase a partir de la clase JButton. A esta nueva clase la llamaremos **BotonContador** y haremos que herede de la clase JButton. Por tanto, la clase *BotonContador* tendrá todas las características de los JButton y además le añadiremos los métodos mencionados antes.



Gracias a la herencia, nuestra clase BotonContador poseerá todos los métodos de los JButton (setText, getText, setForeground, setToolTipText, etc.) sin que tengamos que programar estos métodos.

Por otro lado, añadiremos a la clase BotonContador nuestros propios métodos (setPulsaciones, getPulsaciones, etc) y propiedades para que la clase esté preparada para nuestras necesidades.

Resumiendo: programaremos un botón propio, con características necesarias para nuestro trabajo, aprovechando las características de un botón JButton.

Programación de la Clase BotonContador

Creación de la clase derivada

La programación de la clase derivada *BotonContador* se hará en un fichero aparte, al igual que la programación de las clases de creación propia.

Se comenzará definiendo el comienzo de la clase y añadiendo las llaves que limitan su contenido:

```
public class BotonContador extends JButton {


}
```

Observa el código añadido: `extends JButton` antes de la llave de apertura. Este es el código que le permite indicar a Java que nuestra clase *BotonContador* derivará de la clase JButton.

Creación de las propiedades de la clase derivada

La clase *BotonContador* es una clase derivada de la clase JButton, y tendrá las mismas propiedades que la clase JButton, pero a nosostros nos interesa añadir nuestras propias propiedades. En nuestro caso, necesitaremos una variable que contenga el número de pulsaciones del botón en todo momento.

```
public class BotonContador extends JButton {

      int pulsaciones;


}
```

Inicialización de las propiedades de la clase derivada

Nuestra propiedad *pulsaciones* debe ser inicializada en el constructor de la clase. Para ello crea el constructor de la clase:

```
public class BotonContador extends JButton {

      int pulsaciones;

      public BotonContador() {
            pulsaciones=0;
      }

}
```

Añadir métodos propios a la clase derivada

Se añadirán los nuevos métodos que queremos que la clase BotonContador posea. Estos son los siguientes:

setPulsaciones	- asigna un número de pulsaciones al botón.
getPulsaciones	- devuelve el número de pulsaciones del botón.
incrementa	- suma uno a las pulsaciones del botón
decrementa	- resta uno a las pulsaciones del botón
reiniciar	- pone a cero las pulsaciones del botón
aumentar	- aumenta en una cantidad las pulsaciones del botón.
disminuir	- disminuye en una cantidad las pulsaciones del botón.

Estos métodos trabajan con la propiedad *pulsaciones*. Una vez programados estos métodos, la clase quedará de la siguiente forma (observa la programación de los distintos métodos para entenderlos):

```
public class BotonContador extends JButton {

	//propiedades
	int pulsaciones;

	//constructor
	public BotonContador() {
		pulsaciones=0;
	}

	//métodos

	//asigna una cantidad de pulsaciones
	public void setPulsaciones(int p) {
		pulsaciones=p;
	}

	//devuelve las pulsaciones del botón
	public int getPulsaciones() {
		return pulsaciones;
	}

	//incrementa en uno las pulsaciones
	public void incrementa() {
		pulsaciones++;
	}

	//decrementa en uno las pulsaciones
	public void decrementa() {
		pulsaciones--;
	}

	//pone las pulsaciones a cero
	public void reiniciar() {
		pulsaciones=0;
	}

	//aumenta las pulsaciones en una cantidad c
	public void aumenta(int c) {
		pulsaciones=pulsaciones+c;
	}

	//disminuye las pulsaciones en una cantidad c
	public void disminuye(int c) {
		pulsaciones=pulsaciones-c;
	}

}
```

Como has podido observar, la creación de una clase heredada es exactamente igual que la creación de una clase propia. La única diferencia es que hacemos que dicha clase herede de otra clase ya existente, dándole más posibilidades sin necesidad de programar nada:

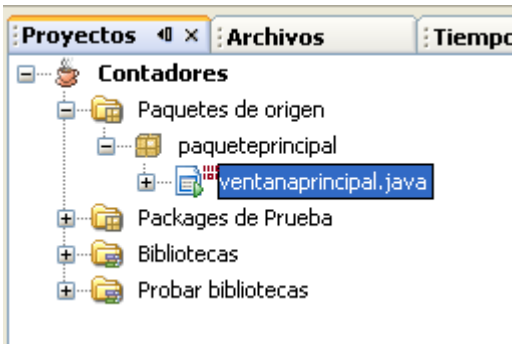
```
		public class BotonContador extends JButton {
```

Uso de la clase BotonContador

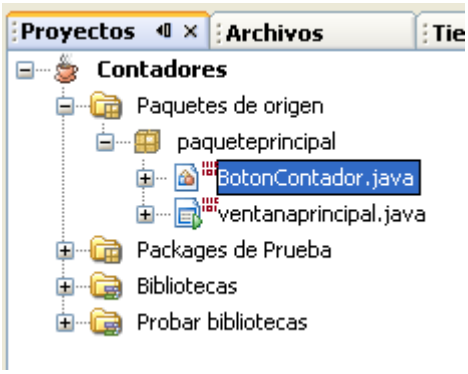
Usaremos la clase BotonContador en un simple proyecto de prueba para que puedas observar sus posibilidades.

Para ello, sigue los pasos que se indican a continuación:

1. Crea un proyecto llamado *Contadores*, que tenga un paquete principal llamado *paqueteprincipal* y un JFrame llamado *ventanaprincipal*:



2. Añadiremos la Clase *BotonContador*. Para ello debes hacer clic con el botón derecho sobre el *paqueteprincipal* y activar la opción *Nuevo* → *Clase Java*. El nombre de la clase es *BotonContador*:

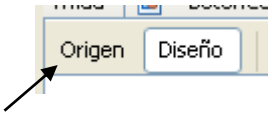


3. Antes de empezar con el diseño de la ventana principal programa la clase BotonContador. Debes hacer doble clic sobre el fichero *BotonContador.java* e introducir en él la clase BotonContador que hemos diseñado anteriormente usando herencia.

   El fichero *BotonContador.java* debería quedar así:

```
/*
 * BotonContador.java
 *
 * Created on 6 de agosto de 2007, 10:59
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package paqueteprincipal;
import javax.swing.JButton;

/**
 *
 * @author didact
 */
public class BotonContador extends JButton {

      //propiedades
      int pulsaciones;

      //constructor
      public BotonContador() {
            pulsaciones=0;
      }

      //asigna una cantidad de pulsaciones
      public void setPulsaciones(int p) {
            pulsaciones=p;
      }

      //devuelve las pulsaciones del botón
      public int getPulsaciones() {
            return pulsaciones;
      }

      //incrementa en uno las pulsaciones
      public void incrementa() {
            pulsaciones++;
      }

      //decrementa en uno las pulsaciones
      public void decrementa() {
            pulsaciones--;
      }

      //pone las pulsaciones a cero
      public void reiniciar() {
            pulsaciones=0;
      }

      //aumenta las pulsaciones en una cantidad c
      public void aumenta(int c) {
            pulsaciones=pulsaciones+c;
      }

      //disminuye las pulsaciones en una cantidad c
      public void disminuye(int c) {
            pulsaciones=pulsaciones-c;
      }
}
```

4. Ahora programaremos el diseño de la ventana. Lo haremos desde código. Entra en el módulo *ventanaprincipal.java* y pasa a la zona de código:



5. En la zona de código programaremos el típico método *CreacionVentana*, llamado desde el constructor, donde se programará el diseño de la ventana:

```
public class ventanaprincipal extends javax.swing.JFrame {

    /** Creates new form ventanaprincipal */
    public ventanaprincipal() {
        initComponents();
        CreacionVentana();
    }

    public void CreacionVentana() {


    }
```

6. Nuestro programa tendrá dos botones del tipo BotonContador. Declara estos botones como variables globales de la clase:

```
public class ventanaprincipal extends javax.swing.JFrame {

    BotonContador btnBotonA;
    BotonContador btnBotonB;

    /** Creates new form ventanaprincipal */
    public ventanaprincipal() {
        initComponents();
        CreacionVentana();
    }
```

7. En el método *CreacionVentana* definiremos características de la ventana y construiremos estos botones y los situaremos en la ventana:

```
public void CreacionVentana() {

    this.setTitle("Ejercicio de Herencia");
    this.setSize(250,300);

    //construimos y situamos los botones contadores.
    btnBotonA = new BotonContador();
    btnBotonA.setText("Boton A");
    btnBotonA.setBounds(10,10,100,30);
    this.getContentPane().add(btnBotonA);

    btnBotonB = new BotonContador();
    btnBotonB.setText("Boton B");
    btnBotonB.setBounds(130,10,100,30);
    this.getContentPane().add(btnBotonB);

}
```

8. Como puedes ver, los botones contadores se construyen igual que los JButton, y, de hecho, tienen los mismos métodos que los JButton, ya que derivan de ellos. Así pues, un botón contador tiene un método *setText*, *setBounds*, etc…

9. Ejecuta el programa y observa el resultado. Verás que en la ventana aparecen los dos botones contadores, sin embargo, observarás que tienen el mismo aspecto que los JButton. Se vuelve a insistir que esto es debido a que la clase BotonContador deriva de la clase JButton.

10. Ahora se añadirá un *actionPerformed* a cada botón contador, para controlar sus pulsaciones. Añade al final de *CreacionVentana* el siguiente código ya conocido, para la asignación de eventos *actionPerformed* a los botones:

```
btnBotonA.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        PulsacionBotonA(evt);
    }
});

btnBotonB.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        PulsacionBotonB(evt);
    }
});
```

11. Se tendrán que programar los procedimientos de respuesta a los eventos, que aquí se han llamado PulsacionBotonA y PulsacionBotonB. Prográmalos (fuera de *CreacionVentana*, claro está) de la siguiente forma:

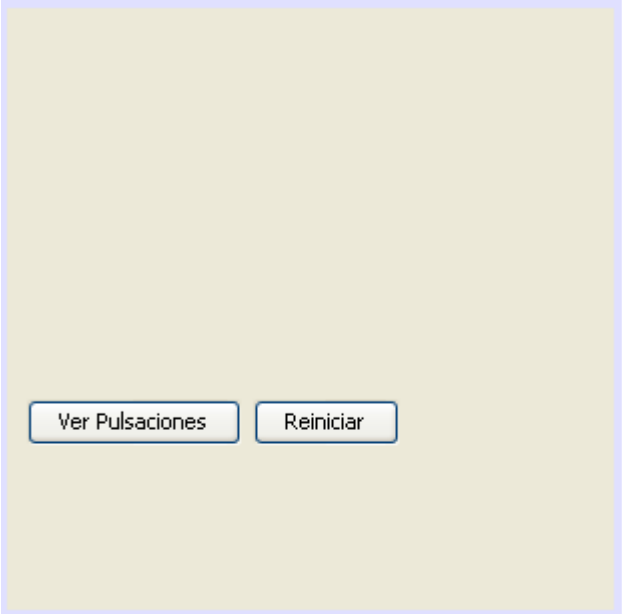
```
public void PulsacionBotonA(ActionEvent evt) {

}

public void PulsacionBotonB(ActionEvent evt) {

}
```

12. Cada vez que se pulse el botón A, debería aumentar su contador interno de pulsaciones en uno. Lo mismo debería pasar con el botón B. Esto se hace fácilmente usando el método *incrementa*, propio de los botones contadores:

```
public void PulsacionBotonA(ActionEvent evt) {
        btnBotonA.incrementa();
}

public void PulsacionBotonB(ActionEvent evt) {
        btnBotonB.incrementa();
}
```

13. Resumiendo, cada vez que se pulsa el botón A, se le da la orden al botón A de que se incremente su contador interno. Lo mismo sucede con el botón B.

14. Ahora programaremos dos botones más en la ventana, pero estos serán botones normales y los crearemos desde la ventana de diseño, para facilitar la tarea. Estos botones se llamarán *btnVerPulsaciones* y *btnReiniciar*.



15. Cuando se pulse el botón *Ver Pulsaciones* debería aparecer un JOptionPane indicando cuantas veces se ha pulsado el botón A y cuantas el B. Esto se hará simplemente pidiendo a cada botón su número de pulsaciones almacenadas.

    Accede al *actionPerformed* del botón *Ver Pulsaciones* y programa lo siguiente:

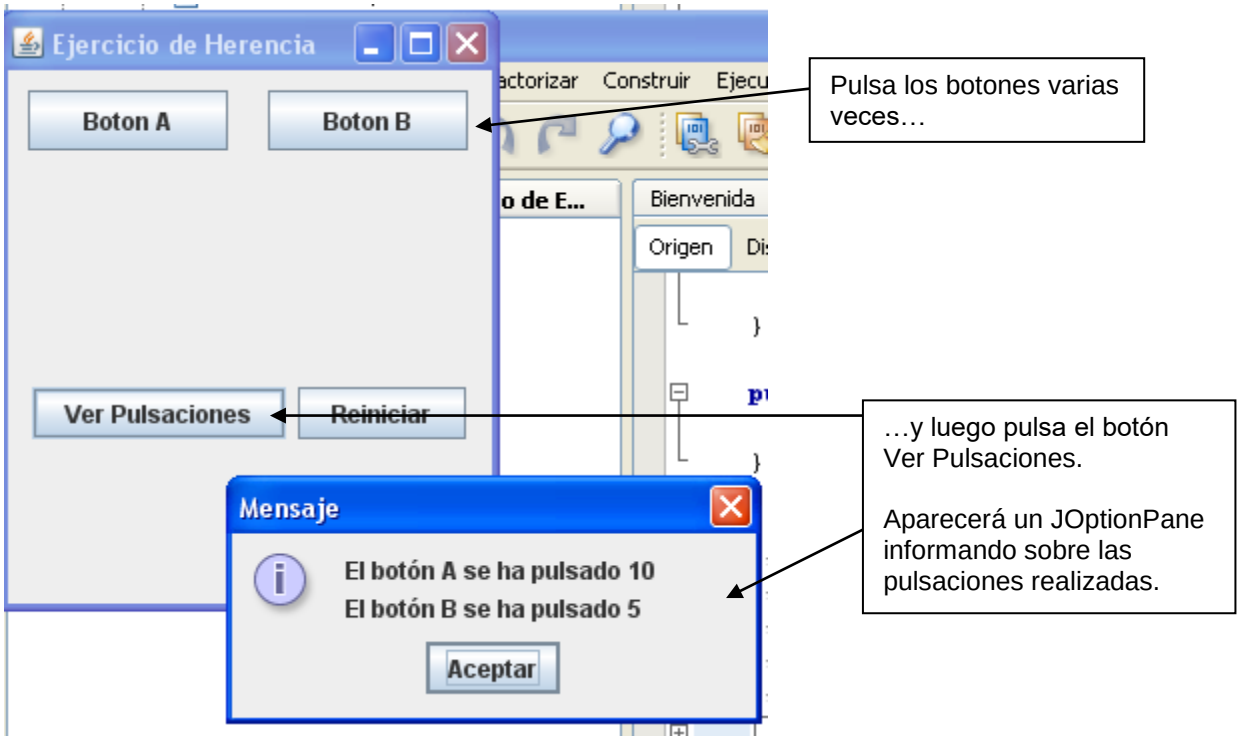
```
    private void btnVerPulsacionesActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String info;

        info = "El botón A se ha pulsado "+btnBotonA.getPulsaciones()+"\n";
        info = info + "El botón B se ha pulsado "+btnBotonB.getPulsaciones()+"\n";
        JOptionPane.showMessageDialog(null,info);

    }
```

    En este botón le pedimos al Botón A que nos diga cuantas pulsaciones tiene anotadas (a través del método *getPulsaciones*) y lo mismo hacemos con el Botón B. Esto es posible ya que ambos botones son del tipo BotonContador, la clase heredada que hemos programado.

16. Ejecuta el programa y prueba lo siguiente:

- Pulsa varias veces el botón A y el B. Verás que no sucede nada (aunque internamente cada uno de los botones está guardando el número de pulsaciones en su propiedad *pulsaciones*)

- Pulsa ahora el botón *Ver Pulsaciones* y comprobarás como este botón te muestra las veces que pulsaste cada botón.

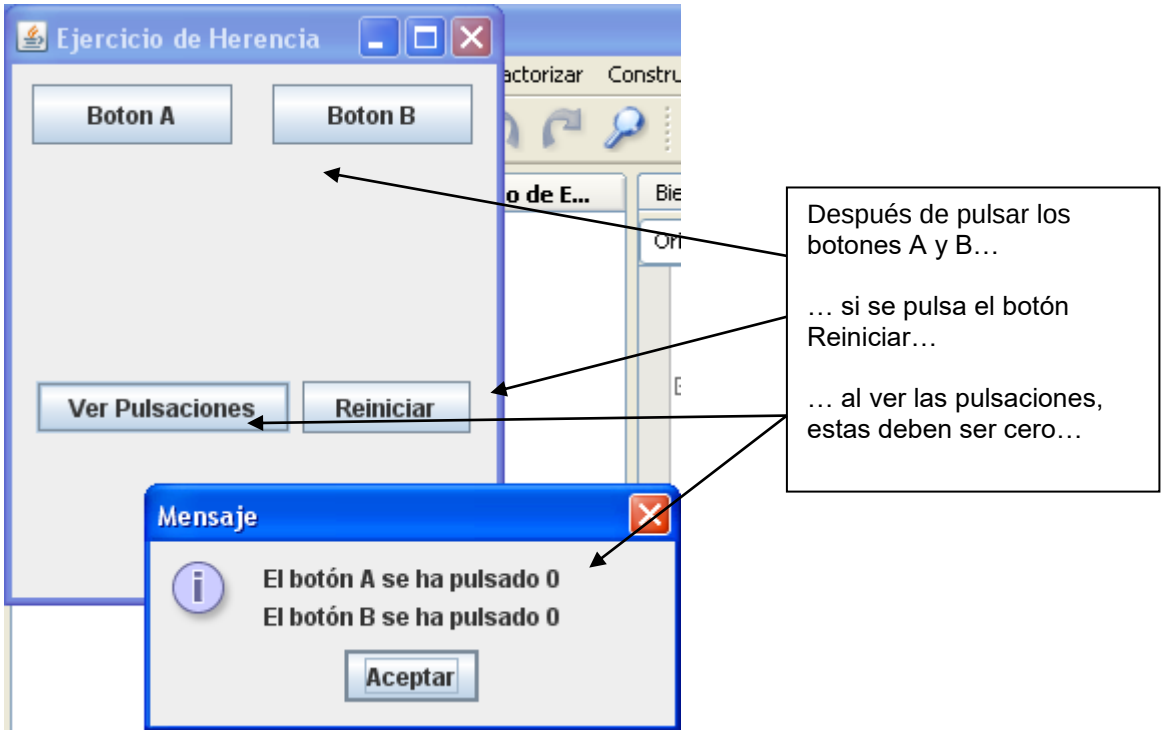


17. Ahora programaremos el botón *Reiniciar* de forma que los contadores internos de ambos botones A y B se pongan a cero.

18. Accede al *actionPerformed* del botón *Reiniciar* y programa lo siguiente:

```
    private void btnReiniciarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        btnBotonA.reiniciar();
        btnBotonB.reiniciar();
    }
```

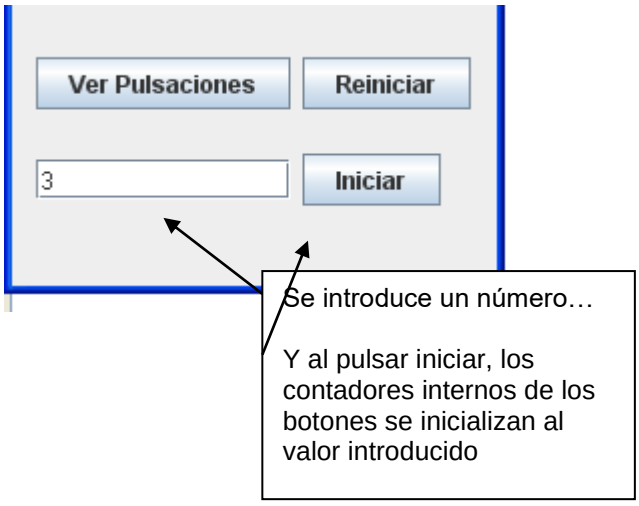
19. Puedes comprobar el funcionamiento de este botón ejecutando el programa y pulsando varias veces los botones A y B. Luego mira las pulsaciones almacenadas en ambos botones. Pulsa el botón *Reiniciar* y comprueba las pulsaciones de nuevo. Deberían ser cero en ambos botones.



EJERCICIO

Se propone al alumno que añada las siguientes modificaciones al programa:

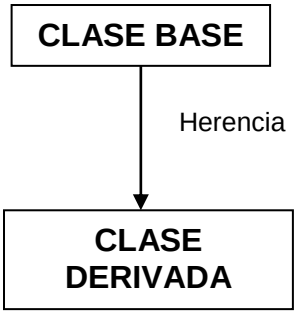
- Añade desde código un nuevo botón Botón C (de tipo BotonContador) que al pulsarse aumente su número de pulsaciones interno en 2 (usa el método *aumentar* propio de la clase BotonContador).
- Cuando se pulse el botón *Ver Pulsaciones* también debería verse el número de pulsaciones del botón C.
- Cuando se pulse Reiniciar, el número de pulsaciones del Botón C debería situarse a cero también.
- Añade desde diseño un cuadro de texto y un botón normal (JButton) llamado *Iniciar*. En el cuadro de texto se introducirá un número, y al pulsar el botón *Iniciar*, los valores internos de los tres botones A, B y C se inicializarán al número introducido:



CONCLUSIÓN

La Herencia consiste en crear una clase que obtenga todas las características de otra. Esta clase a su vez tendrá también características propias.

La clase inicial se denomina clase Base y la clase nueva creada a partir de la clase base se llama clase Derivada:



Se puede hacer que una clase de creación propia *derive* o *herede* de otra ya existente añadiendo:

extends NombreClaseBase

en la línea de creación de la clase.

La clase creada así obtendrá características de la clase de la que hereda.

Aparte, esta clase tendrá métodos propios añadidos por el programador.