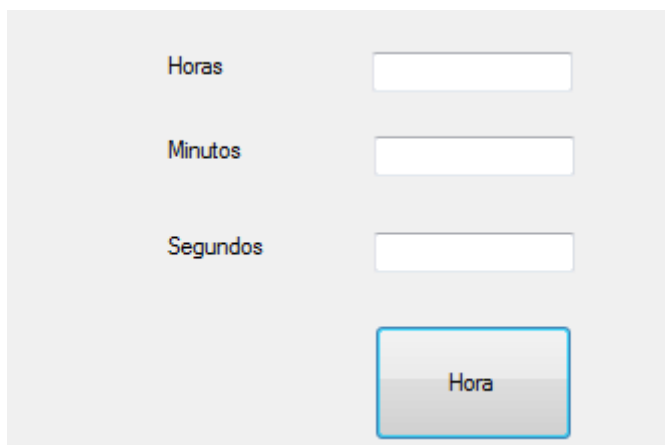


## Caso de prueba caja blanca MostrarHora

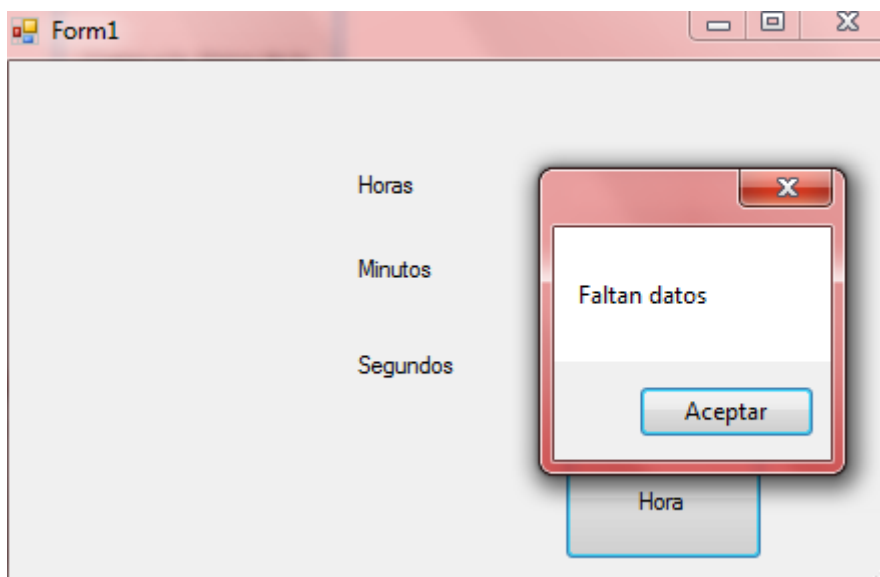
Las clases de equivalencia que utilizaremos serán para horas un rango entre [0,23], para minutos [0,59] y para segundos [0,59] los demás serán números incorrectos que no cumplen la función en el programa. También es una clase de equivalencia los que no sean números.

En nuestro método mostrarhora que es nuestra primera aplicación vamos a comprobar si se cumplen las condiciones de nuestros **if**



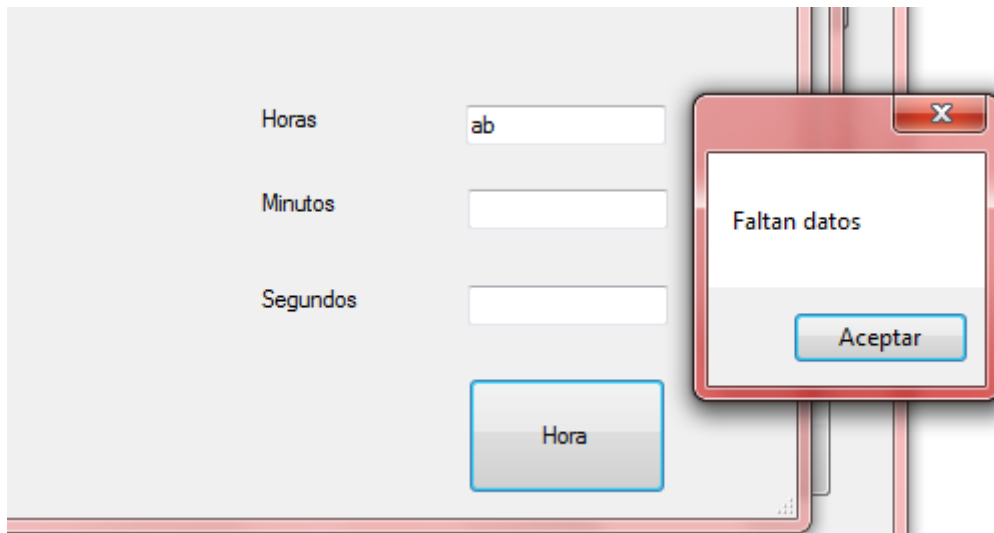
Formulario de entrada para horas, minutos y segundos. El formulario tiene tres campos de texto etiquetados "Horas", "Minutos" y "Segundos". Debajo de estos campos hay un botón etiquetado "Hora".

Primero incluiremos espacios en blanco: Nuestro parámetro de salida ha sido que nos faltan datos

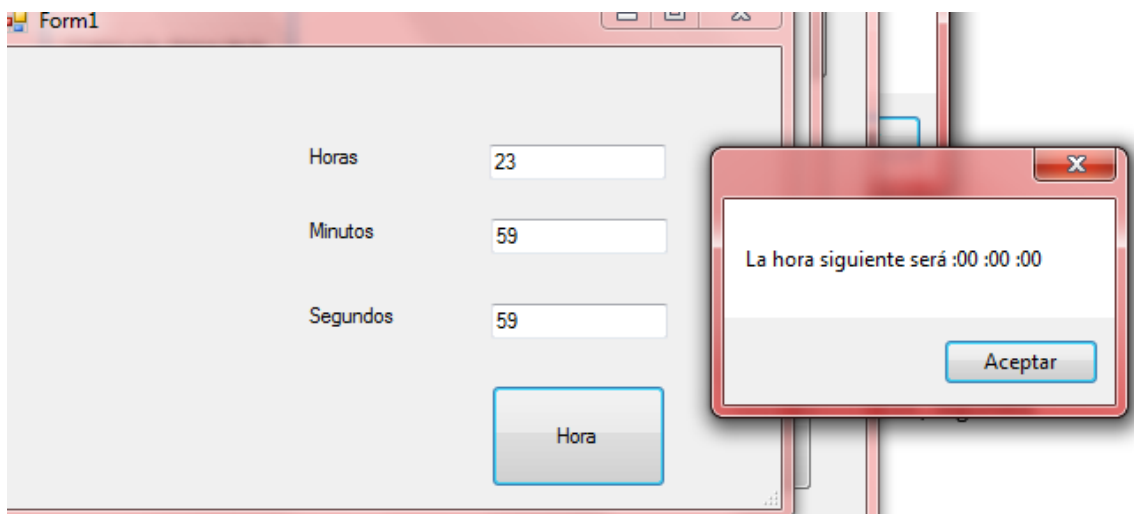


Como no hemos puesto nada se cumple la función de nuestro Catch al no incluir números enteros.

Ahora incluimos strings como **ab** en nuestro programa , se cumple otra vez la función del Catch y nos muestra que faltan datos.



Ahora comprobaremos a poner números correctos , como hora 23 , segundo 59 y minutos 59  
Nos calculará la hora un segundo más como programamos.

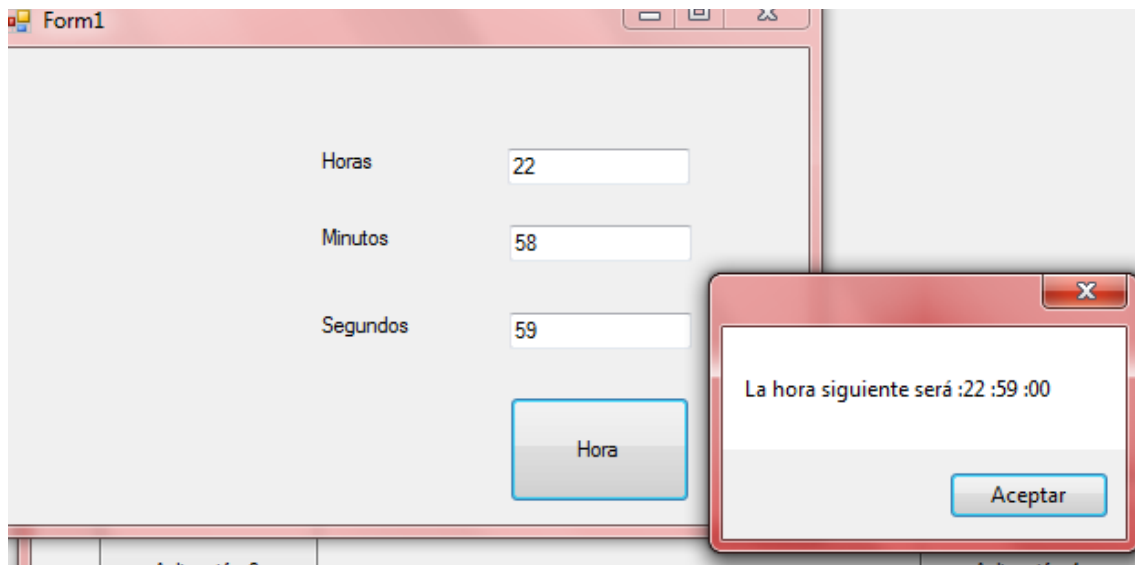


```
if (horas < 24 && minutos < 60 && segundos < 60)
{
    if (segundos < 59)
    {
        segundos = segundos + 1;
        MessageBox.Show("La hora siguiente será :" + horas + " :" + minutos + " :" + segundos);
    }
    if (segundos == 59)
    {
        if (minutos == 59)
        {
            if (horas == 23)
            {
                MessageBox.Show("La hora siguiente será :00 :00 :00 ");
            }
        }
    }
}
```

se cumple el primer if , y los if (segundos == 59) if (minuto == 59) if (horas == 23)

Y la condición del segundo if no se cumple.

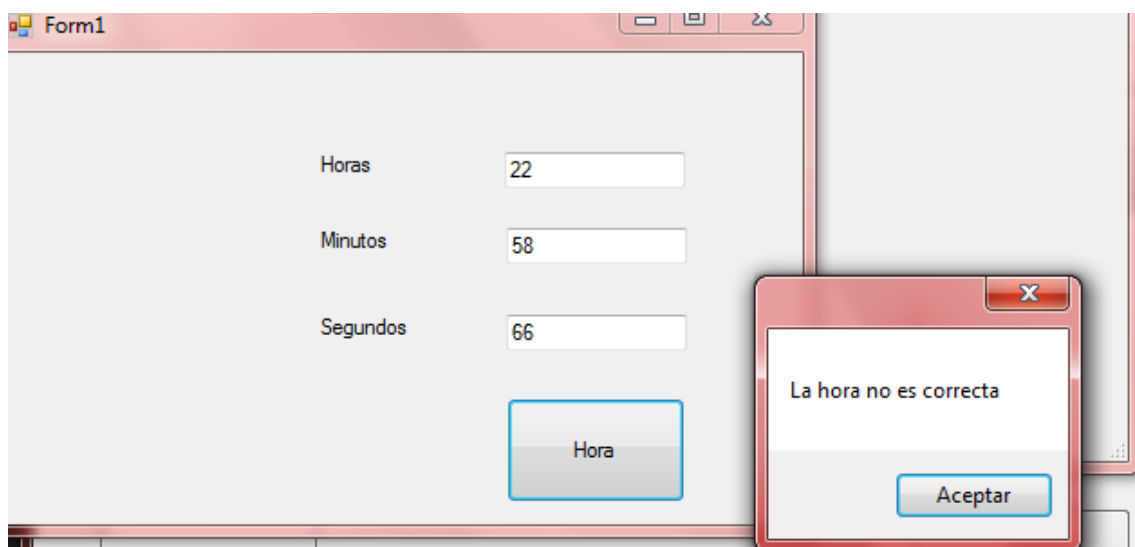
Ahora probamos números correctos entre 0 y 22 horas entre 0 y 58 minutos y 59 segundos



Nos da el resultado correcto.

Pasa por el primer if y el tercer if el de if (segundos == 59) y como no se cumple la condición de (minutos == 59) pasará al else de se if

Vamos a poner cualquier número incorrecto con todos los resultados posibles.



Pasa automáticamente al primer else y te muestra un MessageBox que la hora no es correcta.

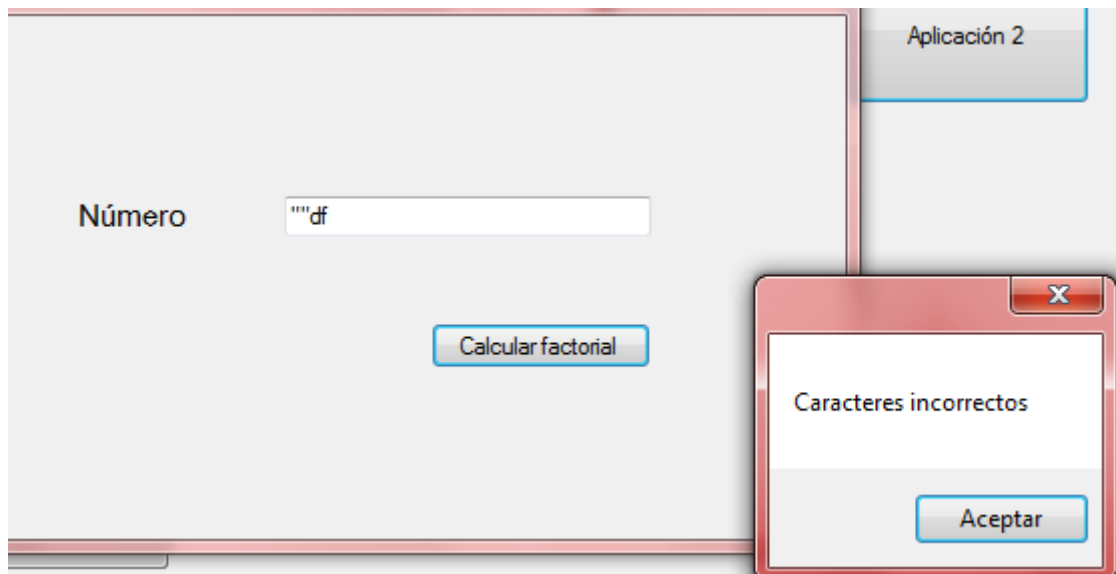
# Caso de prueba caja blanca

## CalcularFactorial

En nuestro método CalcularFactorial que es nuestra segunda aplicación comprobaremos la cobertura de bucles for.

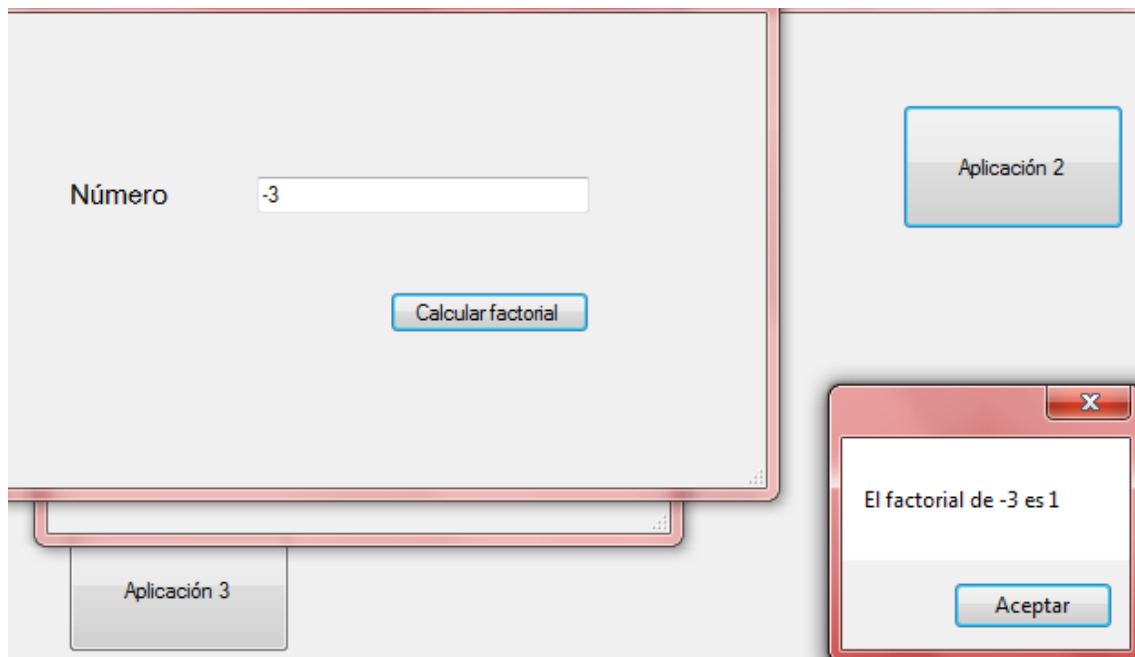
Las clases de equivalencia que utilizaremos serán : Todo lo que no sean números enteros será una clase de equivalencia, números del 4 al 19 serán equivalentes. Los negativos también serán clase de equivalencia.

Primero meteremos de parámetros de entrada caracteres



Se cumple la condición del catch y nos dice que los caracteres son incorrectos.

Ahora probaremos números negativos:

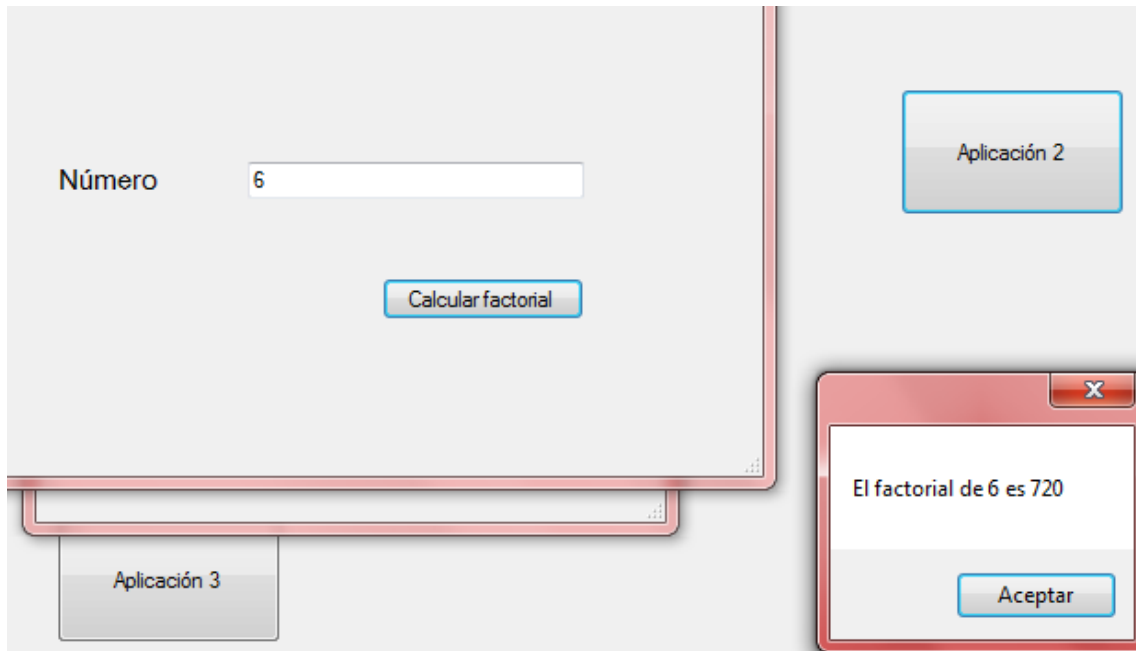


Nos dará el resultado es 1, ya que no se pueden calcular factoriales negativos,

```
for (i = numero; i > 0; i--)  
{  
    factorial = factorial * i;  
}  
  
return factorial;
```

Nuestro bucle for no se llega a activar porque no se cumple la condición de  $i > 0$  y nos mostrará que el factorial es 1, que es lo que tenemos inicializado en nuestra variable.

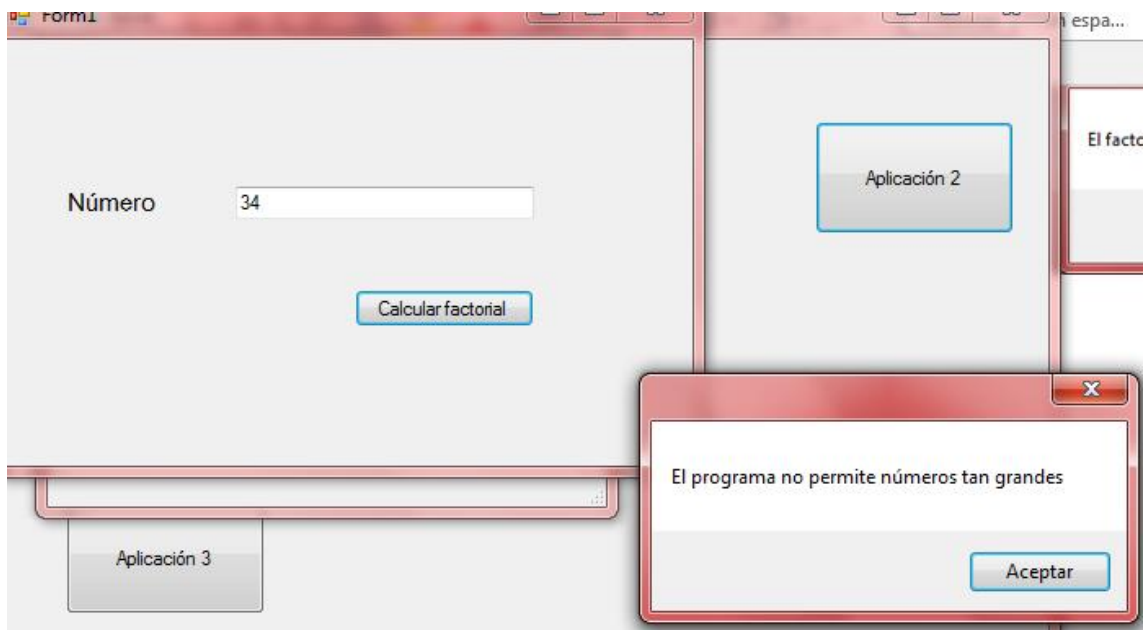
Probemos con números entre el 4 y el 33



Nuestro programa calcula perfectamente.

El bucle for se recorre 6 veces y se acumula perfectamente.

Ahora probemos con un número grande como el 34



Como las variables no pueden acumular números tan grandes nos dará un error de que el programa no permite números así.

```

if (factorial == 0)
{
    MessageBox.Show("El programa no permite números tan grandes");
}
else
{
    MessageBox.Show("El factorial de " + numero + " es " + factorial);
}

```

Cuando te devuelven números que no se pueden mostrar, te saldrá un 0 y si se cumple esa condición nos saldrá un MessageBox.

Y por último probaremos el número 0.



Pasa exactamente lo mismo que con los números negativos

```

for (i = numero; i > 0; i--)
{
    factorial = factorial * i;
}

return factorial;

```

no llega a pasar por el bucle y nos devuelve 1.

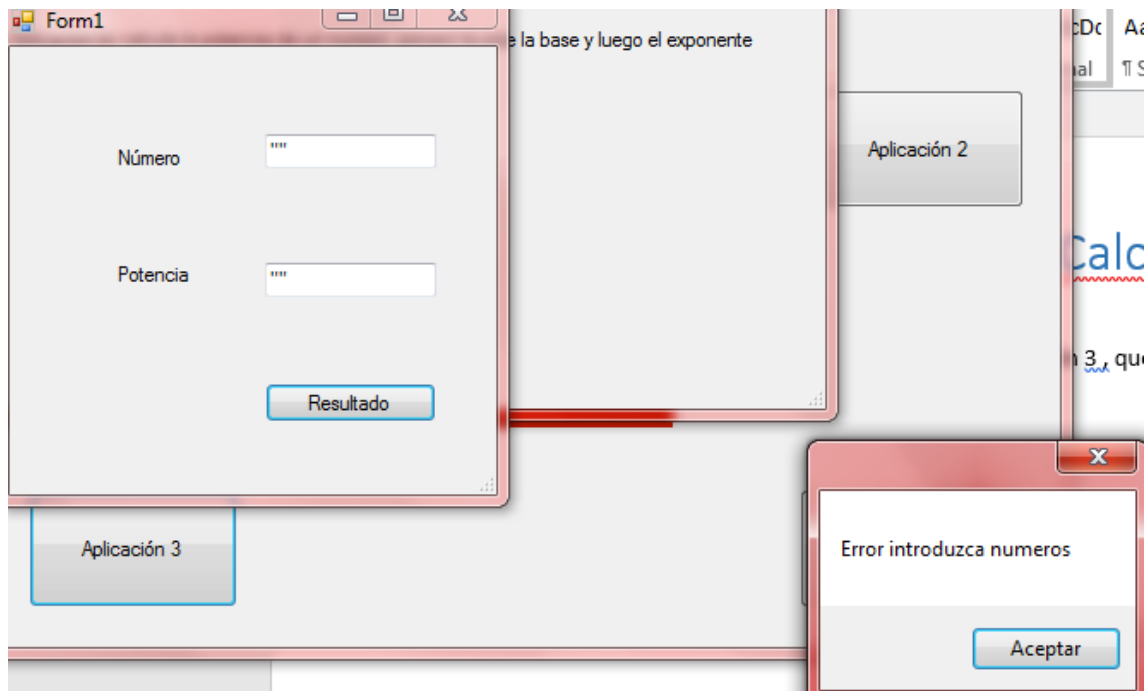
## Caso de prueba caja negra CalcularPotencia

Vamos a hacer pruebas de caja negra con nuestra aplicación 3 , que nos permite calcular una potencia.

Los casos de uso serán todo lo que no sean números enteros. Los números negativos escritos en el exponente serán otro caso de uso.

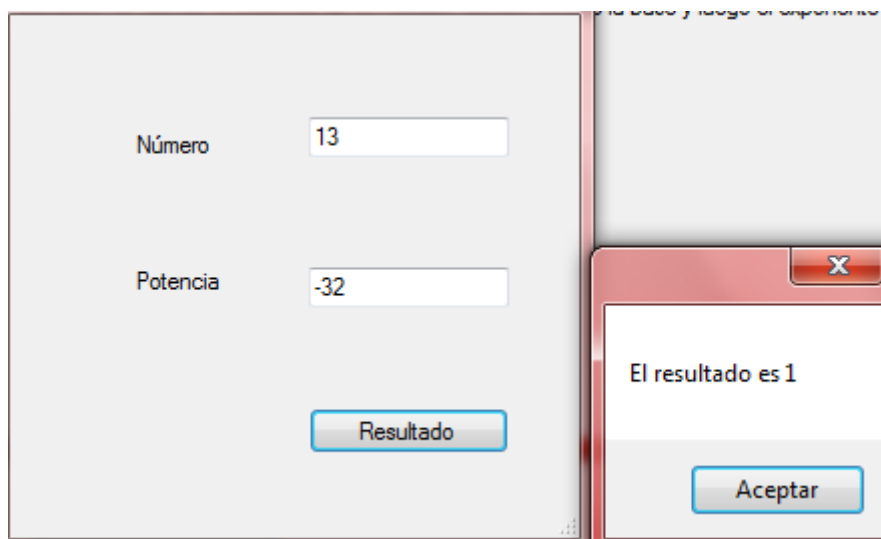
Y los números 0,1,2,3 y -1,-2,-3 serán especiales.

Primero introduciremos mensajes en blanco



Nos da un error de que debemos introducir números.

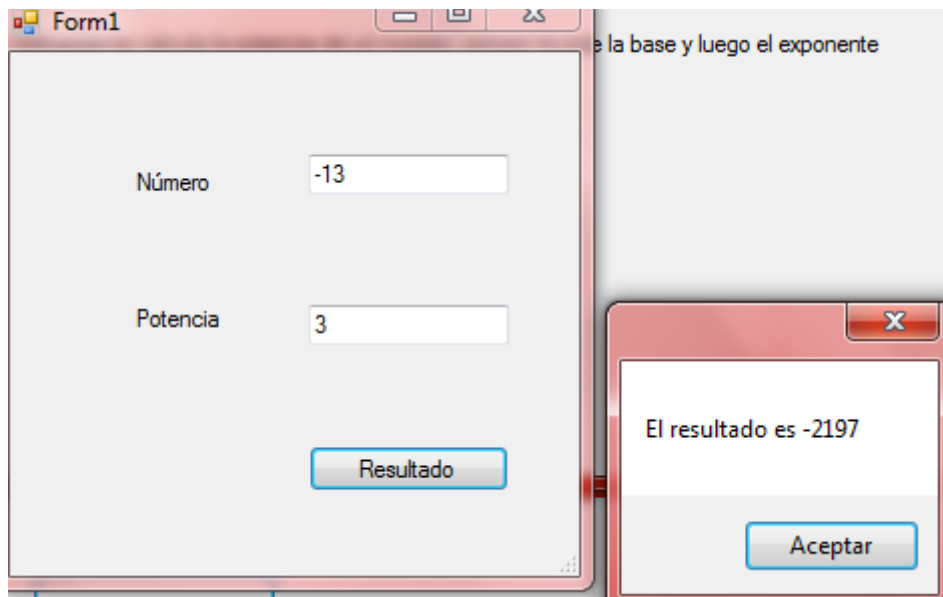
Ahora escribiremos números negativos en la potencia





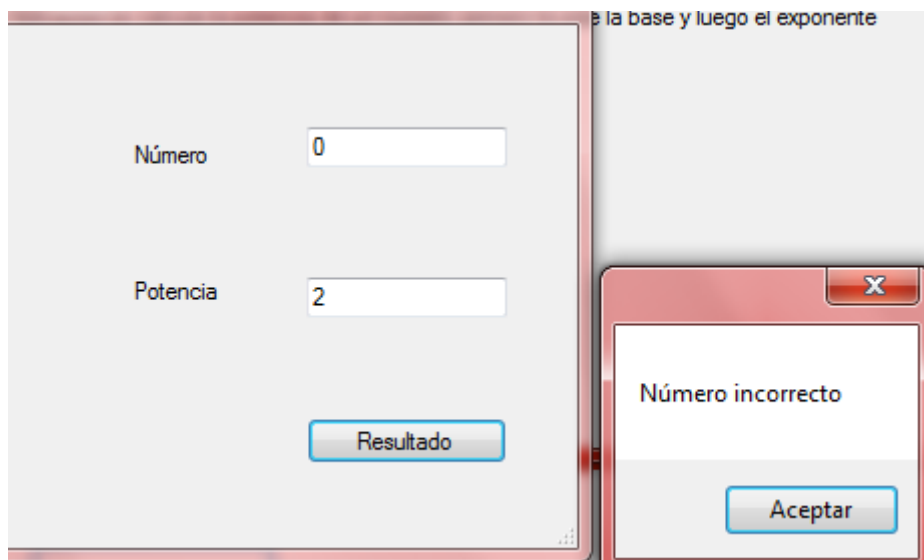
Nos devuelve siempre 1, no se pueden poner potencias negativas .

Ahora probaremos un número negativo en el número y uno positivo en el exponente



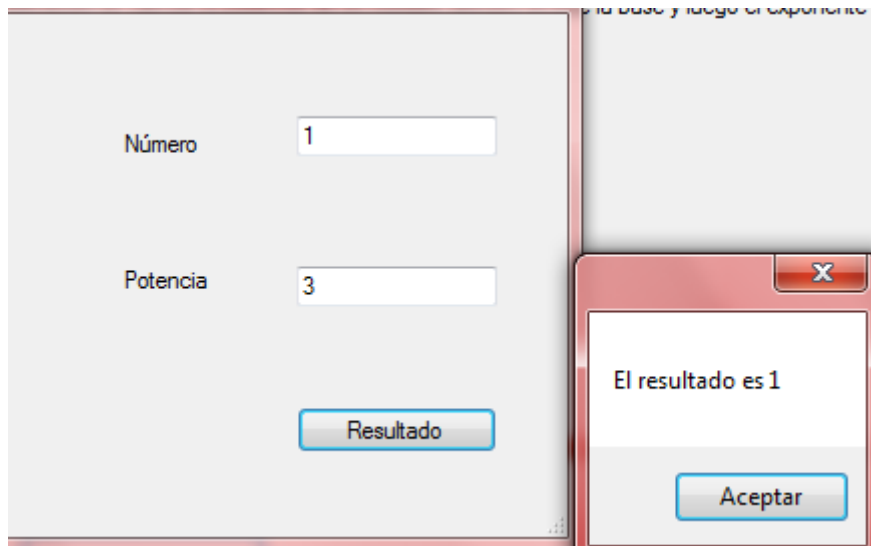
Nos da un resultado correcto.

Ahora probaremos el número 0



Nos da un resultado incorrecto.

Probamos ahora con el 1



Nos da siempre 1.

## Caso de prueba caja negra

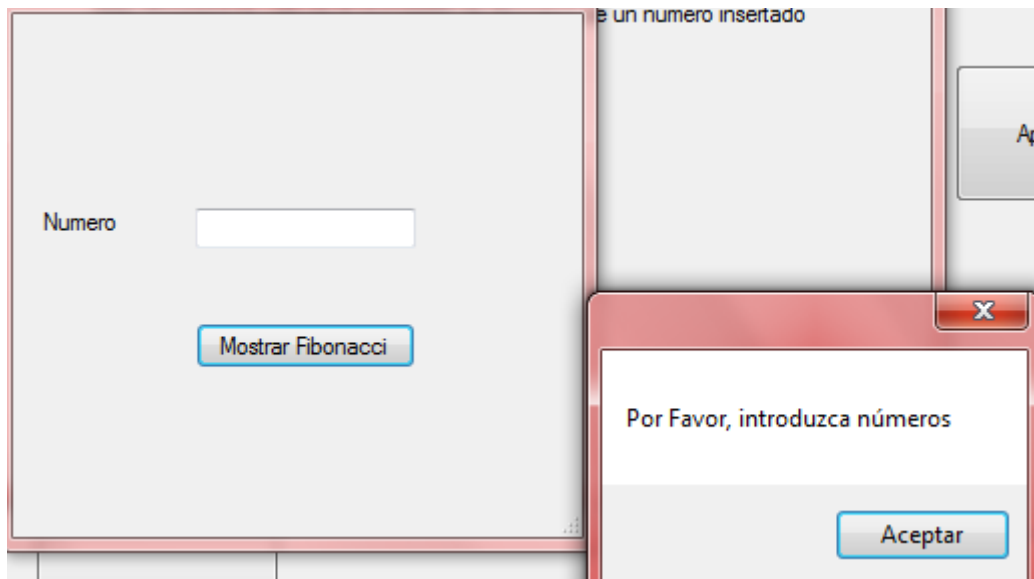
### SucesiónFibonacci

Vamos a probar el método SucesiónFibonacci que es nuestra aplicación 4.

Nuestro casos de equivalencia serán : todo lo que no sean números enteros, números negativos, números positivos entre 4 y 15,

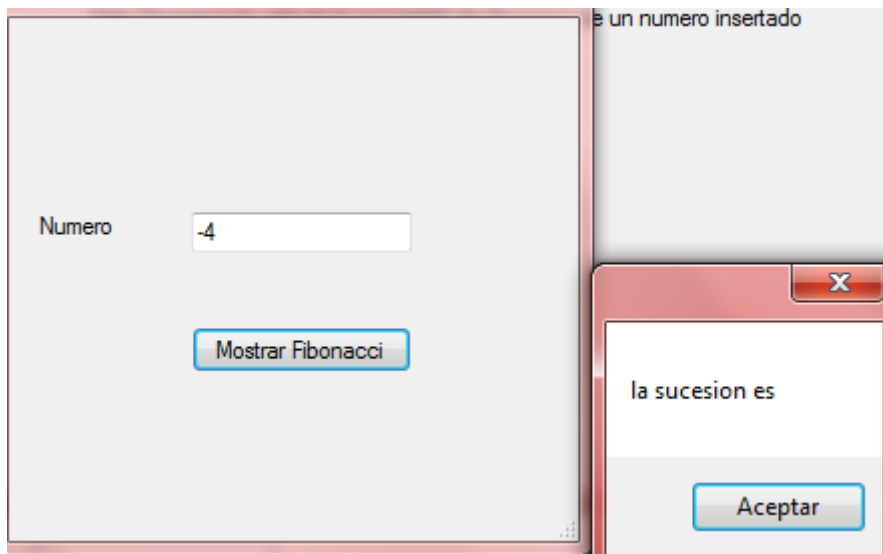
Y los números 0 ,1,2,3 serán casos excepcionales.

Primero probaremos caracteres en blanco.



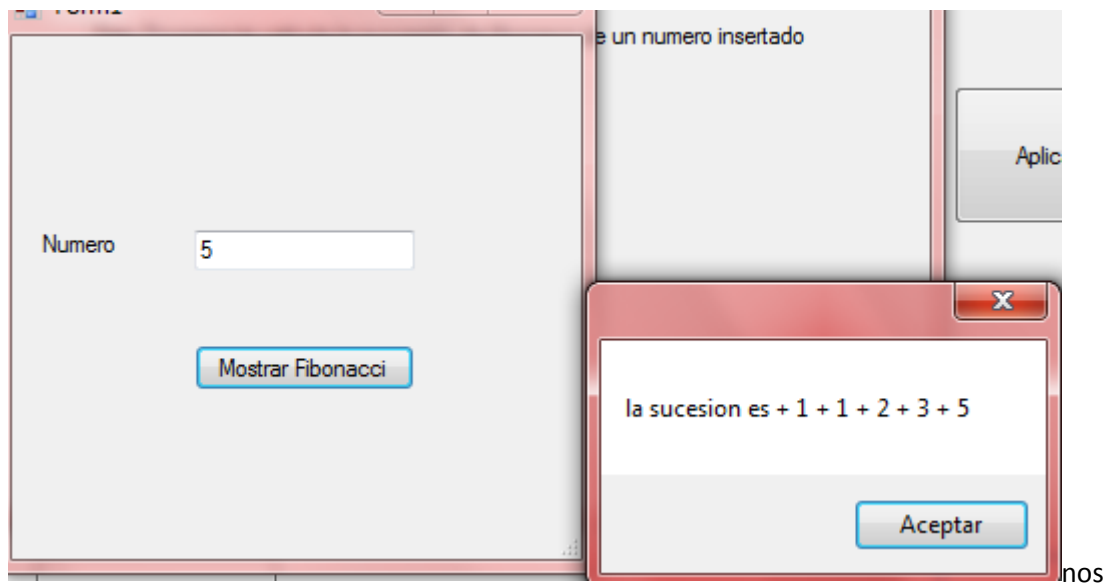
Nos pide por favor que introduzcamos números.

Probemos con los negativos



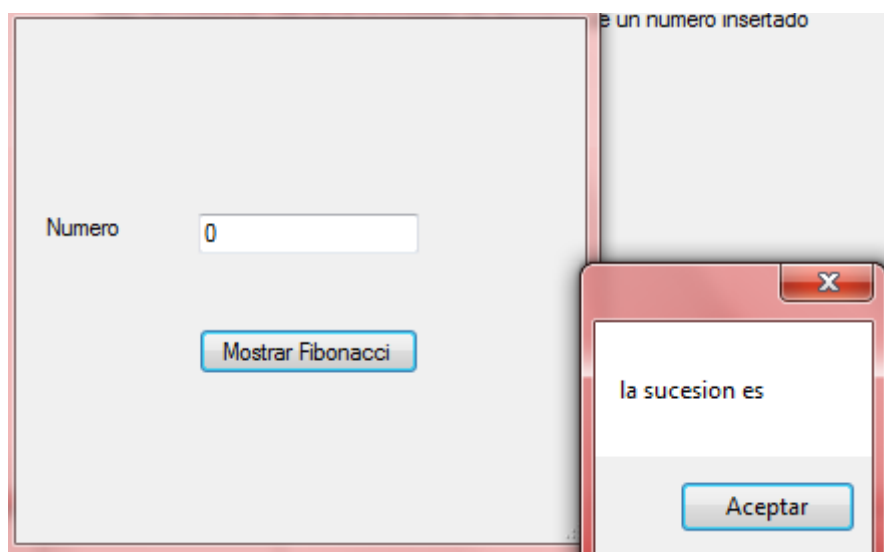
No nos muestra ninguna sucesión.

Probamos ahora números positivos entre 4 y 15



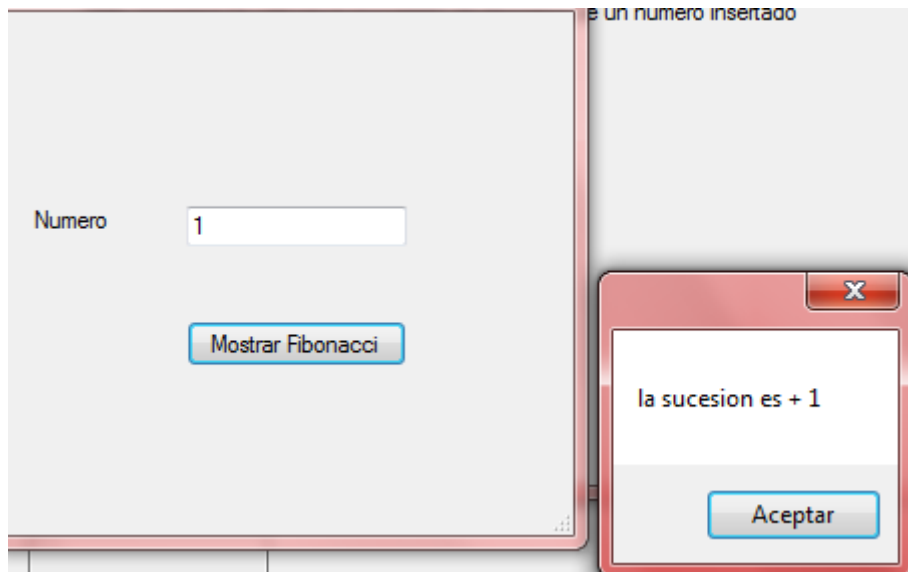
muestra sin problemas la sucesión.

Vamos a probar el número 0



No nos muestra ningún número, todo en blanco.

Probemos ahora con el número 1



Nos muestra la primera sucesión de Fibonacci que es el número 1.