PRUEBAS DE CÓDIGO

Para confirmar que funciona todo correctamente

probar.py

```python
class TestOperaciones(unittest.TestCase):
    def setUp(self):
        # Aquí, opcionalmente, ejecuta lo que de
        # de comenzar cada test.
        pass

    def test_suma(self):
        esperado = 3
        actual = suma(1, 2)
        # Pásalo en el orden: actual, esperado
        self.assertEqual(actual, esperado)

    def test_resta(self):
        esperado = 5
        actual = resta(10, 5)
        # Pásalo en el orden: actual, esperado
        self.assertEqual(actual, esperado)
```

PRUEBAS DE CÓDIGO

Manuales

- Son llevadas a cabo por personas que navegan e interactúan con el software.
- Pruebas costosas y expuestas a errores humanos.

Automatizadas

- Realizadas por máquinas que ejecutan un "test script".
- Más rápidas y confiables que las que se llevan a cabo manualmente

CUBRIMIENTO

- Determina cómo de buenas son nuestras pruebas unitarias.

- Nos dice la cantidad de código que está sometido a nuestras pruebas.

TABLE OF CONTENTS

Classes in this File	Line Coverage	Branch Coverage	Complexity
Autentia	50% 4/8	25% 1/4	4

```
 1     package com.autentia.tutorial;
 2
 3  1  public class Autentia {
 4
 5           public void tellMeSometing(int i) {
 6  1                if (i < 5) {
 7  1                    System.out.println("Soy menor que cinco");
 8  1                    return;
 9                }
10
11  0                if (i % 2 == 0) {
12  0                    System.out.println("Soy un número par");
13  0                    return;
14                }
15  0           }
16    }
```

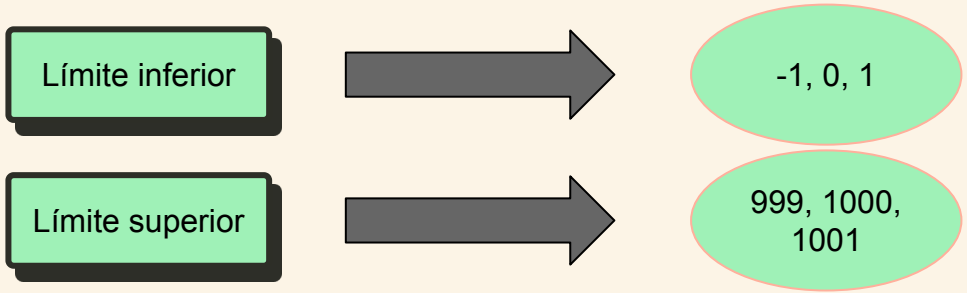
Report generated by Cobertura 1.9 on 10/20/08 10:02 AM.

VALORES LÍMITES

Se trata de tomar los valores que están tanto en el límite inferior como en el superior de un determinado rango. El rango no tiene porque ser numérico, peros si tiene que estar definido, es decir, saber dónde empieza y dónde acaba.

Por ejemplo:

Tenemos un programa que permite valores entre 0 y 1000.



CLASES DE EQUIVALENCIA

El objetivo es ver comportamientos comunes en el rango de valores de nuestro programa, para luego seleccionar un valor representativo de cada conjunto y hacer las pruebas con estos valores

Depurador

- ¿Qué es un depurador?

Un depurador es un programa que permite detectar y diagnosticar fallos en programas informáticos. El objetivo de estas herramientas es garantizar, a largo plazo, que el software funcione en todos los dispositivos y plataformas para los que está pensado. Por este motivo, muchos depuradores no solo analizan el código fuente del programa, sino también su interacción con el sistema operativo que lo ejecuta y con los elementos de hardware. El proceso de depuración o debugging ocurre mientras el programa se ejecuta, de forma que no es necesario cerrarlo para llevar a cabo el análisis.

Ejemplos de Depuradores

01 GNU Debugger

GDB o GNU Debugger es el depurador estándar para el compilador GNU.

02 SoftICE

SoftICE es un depurador en modo kernel propietario y de pago para Microsoft Windows.

03 OllyDbg

OllyDbg es un depurador de código ensamblador de 32 bits para sistemas operativos Microsoft Windows.

04 Cheat Engine

Cheat Engine, abreviado CE, es un escáner de memoria desarrollado como software libre, además de editor hexadecimal y depurador