Pruebas de unidad utilizando JUnit

Juan Manuel Fernández Peña, 2005

JUnit es una herramienta para la realización de pruebas de unidad de software desarrollado con Java, que es una herramienta de software libre, por lo cual se puede extender.

En este documento se describen algunas características de JUnit, cómo se instala y cómo se utiliza. Se muestran dos ejemplos de uso, uno orientado a la prueba de módulos de datos y otro a la de interfaces de usuario.

1. Qué es JUnit y cómo se obtiene

JUnit es una herramienta para Java, llamada JUnit, desarrollada por Erich Gamma y Kent Beck, adoptada y apoyada por grupos partidarios de la programación extrema, la cual, entre otras cosas, sigue una política de primero probar y luego codificar. Ésta y todas las herramientas descendientes de JUnit consisten de una serie de clases que auxilian en la preparación y codificación de casos de prueba y algunos mecanismos auxiliares, que en conjunto permiten ejecutar y verificar el cumplimiento de los casos de prueba. Además provee una interfaz que permite automatizar la ejecución de grupos de casos de prueba.

JUnit ha tenido mucho éxito, por lo cuál a veces se comparan productos comerciales “hace todo lo que Junit y además ...”. Por eso mismo ha sido incorporada en varias IDE. Dos de éstas de gran importancia para Java son Eclipse y NetBeans. La primera es una IDE de software libre (www.eclipse.org) y, entre otras cosas, permite desarrollar software con Java. NetBeans es una herramienta de SUN que se ofrece como una edición comunitaria, por lo que resulta gratis, a pesar de ser propiedad de la compañía.

En caso de no contarse con alguna de tales IDE o preferirse el trabajo con Java desde el SDK (Standard Development Kit), JUnit puede obtenerse en http://www.junit.org/index.htm[1] y se puede copiar a un directorio JUnit y se desempaca. La versión ejecutable viene en una jarra llamada junit.jar, la cual está lista para ser usada. Existe otra jarra con los códigos fuente, para extensiones al mismo.

2. Preparación de las pruebas

Para realizar las pruebas usando Junit, se siguen los pasos que se describen abajo.

1) Crear una clase probadora que extiende TestCase. Debe ser pública.
2) Se incluye la línea
   ```
   import junit.framework.*;
   ```
3) Se agrega la clase a probar (se define una variable privada del tipo de la clase a probar).

[1] Se recomienda visitar el sitio http://www.xprogramming.com/software.htm

4) (opcional) Se crea un método constructor como sigue, donde nombre es el de la clase a probar:

```
public PruebaClase(String nombre)
{ super(nombre); }
```

5) (opcional) Se crea un método setUp que inicializa las variables necesarias.
6) Se crean métodos equivalentes a casos de prueba, que siempre son de tipo void y que incluyen un método que registra el resultado del caso de prueba, como assert, assertEquals o fail. Los nombres de los métodos siempre deben comenzar con la palabra test.
7) (opcional) Se define un método tearDown que limpia y libera las variables creadas para la prueba.
8) Se construye un ensamble (suite) para manejar automáticamente los casos de prueba. La forma más sencilla es como sigue (nombre es el nombre de la clase probadora):

```
Public static Test suite()
{ return new TestSuite(nombre.class); }
```

(TestSuite es descendiente de Test; Test es abstracta).

9) Si se desea probar desde la interfaz de texto, se incluye el siguiente segmento:

```
public static void main(String [ ] args)
{
  junit.textui.TestRunner.run(suite());
}
```

10) Se compila la clase probadora, incluyendo classpath (nombre es el de la clase probadora):

```
javac –classpath .;c:\junit3.8.1\junit.jar nombre
```

[2]

11) Si se usa la prueba desde interfaz de texto, se manda ejecutar la clase probadora incluyendo el classpath (nombre es el de la clase probadora):

```
java –classpath .;c:\junit3.8.1\junit.jar  junit  nombre
```

12) En caso de preferirse el uso de una interfaz gráfica, no se incluye el módulo main (si ya lo tiene, déjelo, no le estorba) y se manda a ejecutar la interfaz y dentro de ella se escribirá el nombre de la clase probadora. La ejecución de la interfaz es como sigue:

```
java –classpath .;c:\junit3.8.1\junit.jar  junit  junit.swingui.TestRunner
```

13) Los resultados aparecen de modo resumido, indicando las fallas en dos grupos: las que identificaron los casos de prueba (failures) y las inesperadas (errors).

3. Ejemplo

A continuación repetimos los mismos pasos pero siguiendo un ejemplo, correspondiente a una clase Cuenta (cuenta bancaria) cuyo código se muestra en la Figura 1. El código se encuentra en el directorio Ejemjunit.

1) Creamos clase PruebaCuenta
2) Incluimos línea de importación de directorio de pruebas

[2] Tenga cuidado con el punto antes del punto y coma; representa el directorio de trabajo

3) Agregamos clase Cuenta (hasta aquí se muestra en Figura 2)
4) Se incluye constructor
5) Se crea método setup muy sencillo (ver Figura 3 que incluye pasos 4 y 5)
6) Se preparan métodos correspondientes a casos de prueba (ver Figura 4 que incluye algunos)

```
public class Cuenta
{
     //Cuenta bancaria simple
     //Juan Manuel Fernández
     //1.0   diciembre 5 de 2005

  private int saldo;
  private String contra;

  public Cuenta(int saldoini, String cc)
  {      //constructor
    saldo = saldoini;
    contra = cc;
  }

  public int deposita(int q)
  {      //receptor de dinero
    saldo += q;
    return saldo;
  }

  public int retira(int q, String cc)
  {      //retira si la contraseña coincide y hay fondos
    if (contra.equals(cc))
      if (saldo >= q) return q; else return -1;
      else return -2;
  }

  public int dimeSaldo()
  {     //verifica saldo
     return saldo;
  }

  public String dimeContra()
  {     //verifica contraseña
     return contra;
  }

}
```

Figura 1 Clase que se desea probar

```
import junit.framework.*;
public class PruebaCuenta extends TestCase
{      //Prueba de Cuenta  /Juan Manuel Fernández  /1.0 diciembre 5 de 2005 //
  private Cuenta micuenta;
}
```

Figura 2 Clase probadora y pasos 1, 2 y 3

```
public PruebaCuenta(String nombre)
{      //constructor
   super(nombre);
}

protected void setUp()
{
   micuenta = new Cuenta(5000,"orion7");
}
```

Figura 3 Pasos 4 y 5

```
public void testDeposita()
  {
     assertEquals(5560, micuenta.deposita(560));
  }

  public void testRetira1()
  {
     assertEquals(-2, micuenta.retira(200,"piscis3"));
  }
```

Figura 4

El paso 7 se omite

7) Se construyen el ensamble (ver figura 5)
8) Se incluye main para prueba de texto (ver figura 6)

```
  public static Test suite()  // ahora formaremos la suite
  {     return TestSuite(PruebaCuenta.class);   }
```

Figura 5 Ensamble de casos de prueba

```
  public static void main(String [] args)
  {      junit.textui.TestRunner.run(suite());   }
```

Figura 6 Ejecución desde interfaz de texto

Finalmente, la clase PruebaCuenta queda como en la Figura 7
Se compila con la línea: javac –classpath .;c:\junit3.8.1\junit.jar PruebaCuenta.java

Ahora, se puede ejecutar desde una ventana de MS-DOS, como se muestra en la Figura 8 (se cambió el fondo por economía de tinta).

```
import junit.framework.*;
public class PruebaCuenta extends TestCase
{      //Prueba de Cuenta  /Juan Manuel Fernández  /1.0 diciembre 5 de 2005 //
  private Cuenta micuenta;

  public PruebaCuenta(String nombre)
  {      //constructor
     super(nombre);
  }

  protected void setUp()
  {
     micuenta = new Cuenta(5000,"orion7");
  }

// casos de prueba

  public void testdimeSaldo()
  {
     assertEquals(5000, micuenta.dimeSaldo());
  }

  public void testdimeContra()
  {
     assertEquals(true, (micuenta.dimeContra()).equals("orion7"));
  }

  public void testDeposita()
  {
     assertEquals(5560, micuenta.deposita(560));
  }

  public void testRetira1()
  {
     assertEquals(-2, micuenta.retira(200,"piscis3"));
  }

  public void testRetira2()
  {
     assertEquals(-1, micuenta.retira(20000,"orion7"));
  }

  public void testRetira3()
  {
     assertEquals(250, micuenta.retira(250,"orion7"));
  }

  public static Test suite()  // ahora formaremos la suite
  {     return TestSuite(PruebaCuenta.class);   }

  public static void main(String [] args)
  {      junit.textui.TestRunner.run(suite());   }
}
```

Figura 7 Clase PruebaCuenta terminada

```
C:\Ejemjunit>java –classpath .;\junit3.8.1\junit.jar PruebaCuenta
.....
Time: 0

OK (6 tests)


C:\Ejemjunit>
```

Figura 8 Ejecución desde interfaz de texto

En la Figura 9 se muestra la ejecución en interfaz gráfica, incluyendo la llamada desde MS-DOS. El campo de texto donde dice “PruebaCuenta” se llena al estar en ejecución.

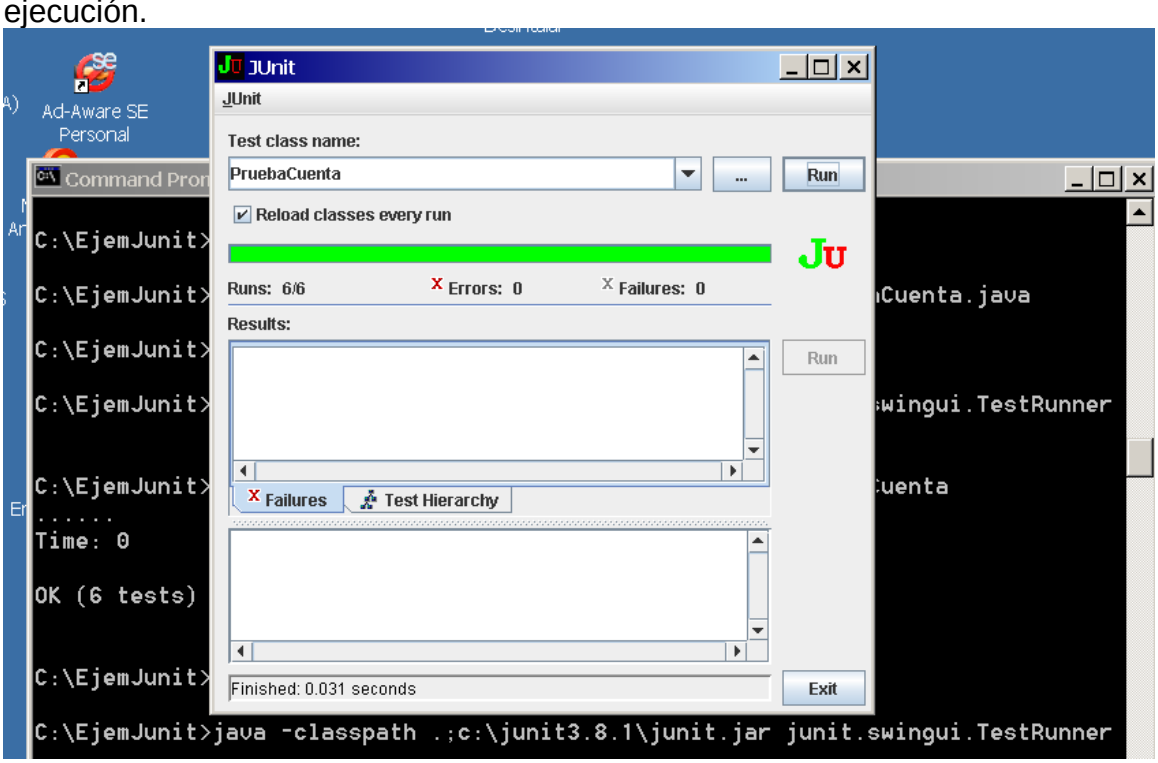

Figura 9 Ejecución desde interfaz gráfica

4. Uso desde NetBeans

La IDE NetBeans es un ambiente de desarrollo que ofrece SUN como beneficio a la comunidad, por lo cual es gratis. Entre los elementos que tiene incorporados está JUnit. Para emplearlo se sigue el procedimiento siguiente:

1) Se compila la clase que se desea probar y se deja abierta (su código aparece en la ventana superior derecha de la aplicación).
2) En el menú Tools, opción JUnitTests, opción CreateTests abre una ventana donde se ofrecen opciones para la generación de la clase de prueba. Por default se llamará Test<nombre de la clase a probar>. Basta dar OK.

3) En un directorio asociado al proyecto donde se tiene la clase que se va a probar, se guarda la clase probadora. Por default crea métodos de prueba por cada método de la clase y todos traen llamadas a fail (método que siempre falla y envía un mensaje).
4) Se reemplazan los métodos que se desea usar y se agregan otros si es necesario.
5) Se compila la clase probadora
6) En el menú Run, opción Other, se selecciona Test <nombre declase a probar> y en la ventana de abajo aparecerá la salida de las pruebas. A diferencia del método a pié, aparecerá una línea por caso de prueba ejecutado.

5. Ejemplo con NetBeans

1) Usando el mismo ejemplo del método manual, se prepara la clase Cuenta y se compila.
2) Se selecciona Tools → JUnitTests → CreateTests → OK, con lo que se crea la clase probadora TestCuenta. (Ver Figura 10)

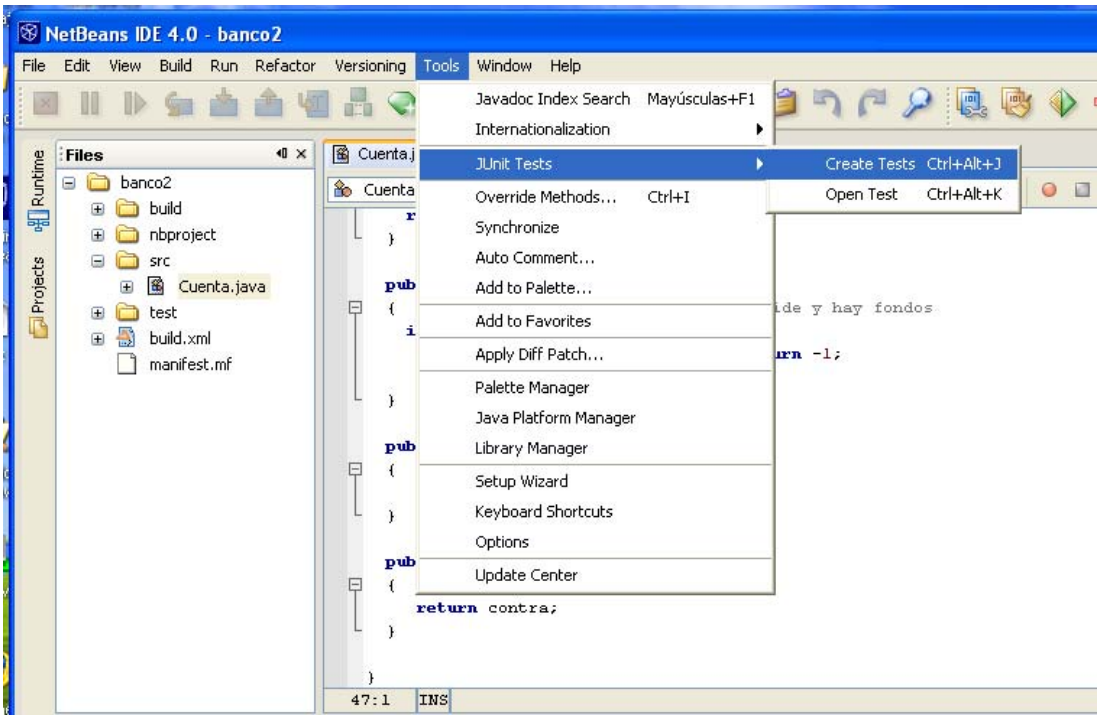

Figura 10 Creación de pruebas para clase Cuenta

3) Los métodos quedan como el ejemplo que se muestra en la Figura 11; reemplazamos el método fail por el contenido que deseamos, quedando como en la Figura 12.

```
public void testDeposita() {
    System.out.println("testDeposita");

    // TODO add your test code below by replacing the default call to fail.
    fail("The test case is empty.");
}
```

Figura 11 Ejemplo de método generado automáticamente

```
public void testDeposita() {
    System.out.println("testDeposita");

    // TODO add your test code below by replacing the default call to fail.
    assertEquals(5560, micuenta.deposita(560));
}
```

Figura 12 Ejemplo de método corregido

4) Se ejecuta: Run → Other → Test Cuenta, apareciendo la salida de la Figura 13.

```
testDeposita
testRetira
testRetira
testRetira
testDimeSaldo
testDimeContra
Testsuite: CuentaTest
Tests run: 6, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
```

Figura 13 Salida de las pruebas

6 Uso de Junit en Eclipse

La plataforma Eclipse también incluye Junit como un plug-in, de modo que se pueden integrar las pruebas con el desarrollo. Sin embargo ofrece menos funcionalidad que NetBeans y deben realizarse a pié más cosas. Para usarlo se siguen los pasos siguientes:

1) Se crea la clase a probar (no se compila porque usualmente la compilación es automática en Eclipse)
2) En **File** se selecciona **New** y ahí Junit Test Case (ver Figura 14). Es posible que indique que Junit no está en el path y pregunta si lo agrega; diga que sí.
3) En la ventana que aparece se indica si se desea setUp, constructor, tearDown o main. Al final se oprime **Finish** (ver Figura 15)
4) Aparecerá una clase con el nombre formado por el de la clase bajo prueba y la palabra Test. Note que no incluye casos de prueba ni el contenido de los métodos incluidos, excepto el constructor.
5) Revise la primera línea (import) y verifique que diga import junit,framework.*;

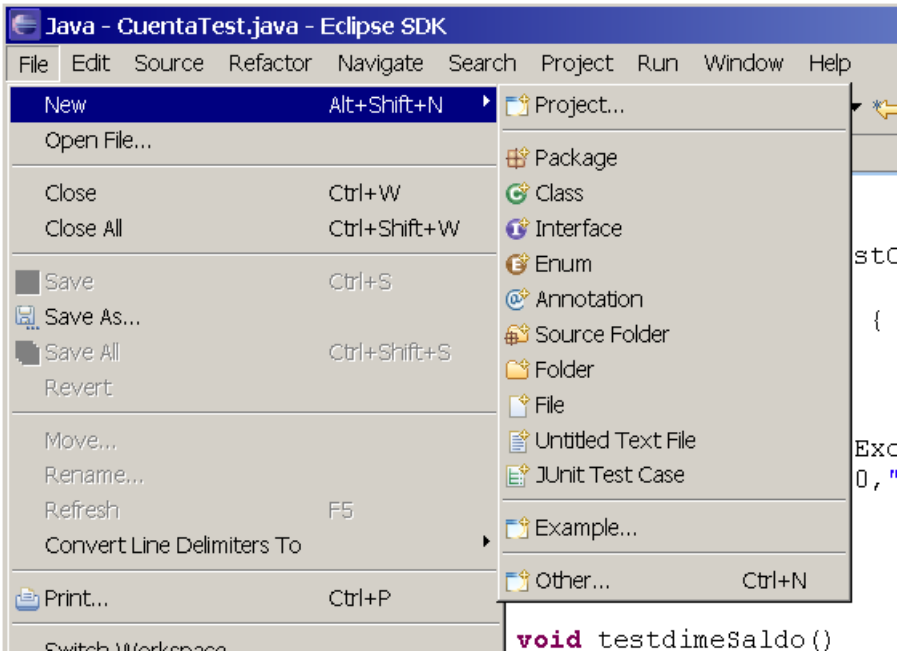

Figura 14 Creación de clase de prueba

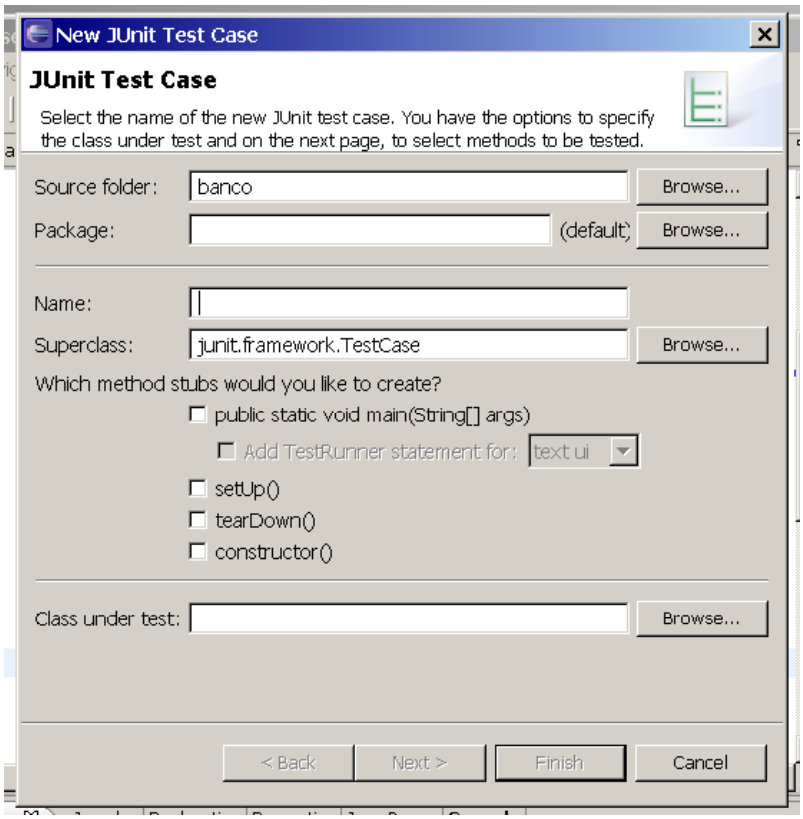

Figura 15 Ventana auxiliar de creación de clase probadora

6) Agregue los métodos correspondientes a caos de prueba, tal como se hacía en el método manual (sección 2).
7) Agregue el método para crear el ensamble (suite) y, si lo requiere, el main, como se indicó en la sección 2.
8) Verifique que no se indiquen problemas, recordando que Eclipse va compilando automáticamente; los problemas se indican en la parte inferior de la ventana, en la pestaña Problems.
9) Si no hay problemas, en el menú **Run** elija **Run As** y ahí Junit Test (ver Figura 16)

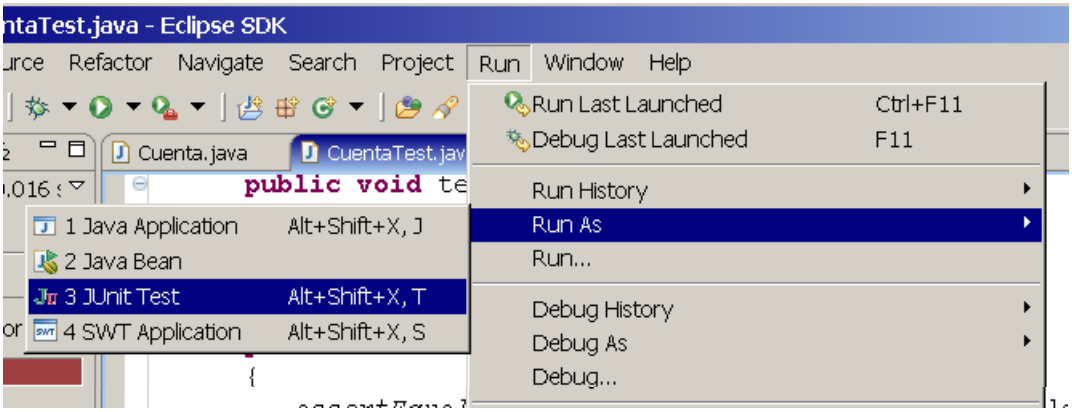

Figura 16 Iniciando ejecución de casos de prueba

10) El resultado de la ejecución se mostrará en el lado izquierdo de la ventana de Eclipse (ver Figura 17) mostrando una barra verde si todos los casos pasaron y roja si no. Los problemas se listan abajo, como en el ejemplo de la Figura 17.

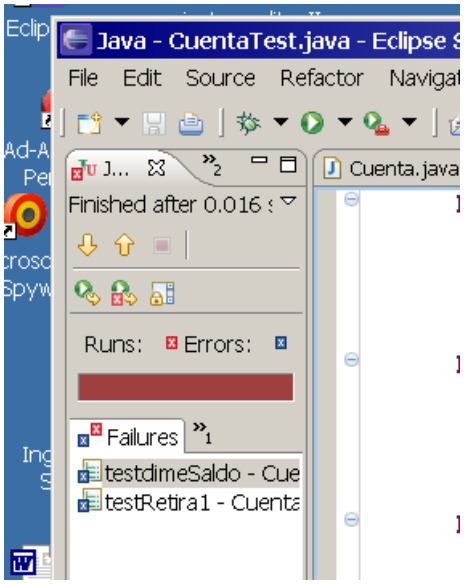

Figura 17 Resultado de la ejecución de casos de prueba mostrando dos fallas