Soluciones Kata TDD

Este documento recoge las soluciones a los primeros 8 ejercicios de la Kata Test **Driven Development (TDD): Example Walkthrough** de **Viktor Farcic**: https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/

Solución requisito 1

[JAVA TEST]

```
import org.junit.Assert;
import org.junit.Test;

import kataTDD.StringCalculator1;

public class StringCalculator1Test {


	@Test(expected = RuntimeException.class)
	public final void cuandoHayMasDeDosNumerosEntoncesSeLanzaUnaExcepcion() {
		StringCalculator1.sumar("1,2,3");
	}

	@Test
	public final void cuandoHayDosNumerosNoSeLanzaExcepcion() {
		StringCalculator1.sumar("1,2");
		Assert.assertTrue(true);
	}

	@Test(expected = RuntimeException.class)
	public final void cuandoSeUsaUnCaracterNoNumericoSeLanzaUnaExcepción() {
		StringCalculator1.sumar("1,X");
	}
}
```

Es una buena práctica nombrar a los métodos de prueba de forma que sea fácil de comprender lo que están probando. Aquí se ha usado una notación basada en BDD Cuando [acción] Entonces [Verificación]. En este caso el nombre de uno de los métodos de test es `cuandoHayMasDeDosNumerosEntoncesSeLanzaUnaExcepcion`.

Si hay más de dos números o algún carácter no es numérico se lanzará una excepción. La palabra expected dice al runner de Junit que se espera que el resultado de la prueba sea el lanzamiento de una excepción.

[JAVA IMPLEMENTATION]

```
package kataTDD;

public class StringCalculator1 {

	public static void sumar(final String numeros) {
		String[] ArrayNumeros = numeros.split(",");
		if (ArrayNumeros.length > 2) {
			throw new RuntimeException("Se permiten hasta 2 números separados
					por comas");
		} else {
			for (String numero: ArrayNumeros) {
				Integer.parseInt(numero); // Si no es un número parseInt lanzará una excepción
			}
		}
	}
}
```

Tener en cuenta que la idea es que en TDD se haga lo mínimo necesario para pasar el test y repetir el proceso hasta que la funcionalidad completa esté implementada. En este momento solo estamos interesados en que el método sea capaz de aceptar 0, 1 o 2 números.

Solución requisito 2

[JAVA TEST]

```
	@Test
	public final void cuandoSeUsaLaCadenaVaciaEntoncesElValorDevueltoEs0() {
		Assert.assertEquals(0, StringCalculator2.sumar(""));
	}
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

public class StringCalculator2 {

	public static int sumar(final String numeros) { // Cambiado para evitar void
		String[] arrayNumeros = numeros.split(",");
		if (arrayNumeros.length > 2) {
			throw new RuntimeException("Se permiten hasta 2 números separados
                                        por comas");
		} else {
			for (String numero : arrayNumeros) {
				if (!numero.isEmpty()) {
					Integer.parseInt(numero);
				}
			}
		}
		return 0; // Añadido return
	}
}
```

Solución requisito 3

[JAVA TEST]

```java
@Test
public final void cuandoRecibeUnNumeroEntoncesDevuelveElMismoNumero() {
    Assert.assertEquals(3, StringCalculator3.sumar("3"));
}

@Test
public final void cuandoRecibeDosNumerosEntoncesDevuelveSuSuma() {
    Assert.assertEquals(3+6, StringCalculator3.sumar("3,6"));
}
```

[JAVA IMPLEMENTATION]

```java
package kataTDD;

public class StringCalculator3 {

	public static int sumar(final String numeros) {
		int resultado = 0; // Añadida variable para devolver el resultado
		String[] arrayNumeros = numeros.split(",");
		if (arrayNumeros.length > 2) {
			throw new RuntimeException("Se permiten hasta 2 números separados
                                        por comas");
		}
		for (String numero : arrayNumeros) {
			if (!numero.trim().isEmpty()) { // Añadida pregunta para evitar excepción si cadena vacía
				resultado += Integer.parseInt(numero);
			}
		}
		return resultado; // Se devuelve resultado
	}

}
```

Solución requisito 4

[JAVA TEST]

```
@Test
public final void cuandoRecibeMasDeDosNumerosEntoncesDevuelveSuSuma() {
    Assert.assertEquals(3+6+15+18+46+33, StringCalculator4.sumar("3,6,15,18,46,33"));
}
```

También hay que quitar el caso de prueba que controla que se lance una excepción si se envían más de 2 números en la cadena.

[JAVA IMPLEMENTATION]

```
package kataTDD;

public class StringCalculator4 {

	public static int sumar(final String numeros) {
		int resultado = 0;
		String[] arrayNumeros = numeros.split(",");
		// Eliminado el lanzamiento de excepción si vienen más de 2 números en la cadena
		for (String numero : arrayNumeros) {
			if (!numero.trim().isEmpty()) {
				resultado += Integer.parseInt(numero);
			}
		}
		return resultado;
	}
}
```

Solución requisito 5

[JAVA TEST]

```
@Test

public final void cuandoSeUsaNuevaLineaEntreNumerosEntoncesSeDevuelveSuSuma() {

   Assert.assertEquals(3+6+15, StringCalculator5.sumar("3,6n15"));

}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

public class StringCalculator5 {

	public static int sumar(final String numeros) {
		int resultado = 0;
		String[] arrayNumeros = numeros.split(",|n"); // Se añade |n a split
		for (String numero : arrayNumeros) {
			if (!numero.trim().isEmpty()) {
				resultado += Integer.parseInt(numero);
			}
		}
		return resultado;
	}
}
```

Solución requisito 6

[JAVA TEST]

```
@Test
public final void cuandoSeEspecificaUnDelimitadorEntoncesSeUsaParaSepararNumeros() {
   Assert.assertEquals(3+6+15, StringCalculator6.sumar("//;n3;6;15"));
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

public class StringCalculator6 {
	/*
	 * Partimos el código en 2 métodos. El método inicial analiza la cadena buscando
	 * el delimitador y después llama al método sumar con los parámetros cadena y
	 * delimitador para que sume los números.
	 */

	public static int sumar(final String numeros) {
	    String delimitador = ",|n";
	    String numerosSinDelimitador = numeros;
	    if (numeros.startsWith("//")) {
	        int indice = numeros.indexOf("//") + 2;
	        delimitador = numeros.substring(indice, indice + 1);
	        numerosSinDelimitador = numeros.substring(numeros.indexOf("n") + 1);
	    }
	    return sumar(numerosSinDelimitador, delimitador);
	}

	private static int sumar(final String numeros, final String delimitador) {
	    int resultado = 0;
	    String[] arrayNumeros = numeros.split(delimitador);
	    for (String numero : arrayNumeros) {
	        if (!numero.trim().isEmpty()) {
	            resultado += Integer.parseInt(numero.trim());
	        }
	    }
	    return resultado;
	}
}
```

Solución requisito 7

[JAVA TEST]

```
@Test(expected = RuntimeException.class)
public final void cuandoSeUsanNumerosNegativosEntoncesSeLanzaExcepción() {
    StringCalculator7.sumar("3,-6,15,18,46,33");
}
@Test
public final void cuandoSeUsanNumerosNegativosEntoncesSeLanzaExcepciónConMensaje() {
    RuntimeException exception = null;
    try {
        StringCalculator7.sumar("3,-6,15,-18,46,33");
    } catch (RuntimeException e) {
        exception = e;
    }
    Assert.assertNotNull(exception);
    Assert.assertEquals("Valores negativos no permitidos: [-6, -18]", exception.getMessage());
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

import java.util.ArrayList;
import java.util.List;

public class StringCalculator7 {

	public static int sumar(final String numeros) {
		String delimitador = ",|n";
		String numerosSinDelimitador = numeros;
		if (numeros.startsWith("//")) {
			int indice = numeros.indexOf("//") + 2;
			delimitador = numeros.substring(indice, indice + 1);
			numerosSinDelimitador = numeros.substring(numeros.indexOf("n") + 1);
		}
		return sumar(numerosSinDelimitador, delimitador);
	}

	@SuppressWarnings("unchecked")
```

```
    private static int sumar(final String numeros, final String delimitador) {
        int resultado = 0;
        String[] arrayNumeros = numeros.split(delimitador);
        @SuppressWarnings("rawtypes")
           List numerosNegativos = new ArrayList(); // Se añade esta lista para almacenar negativos
        for (String numero : arrayNumeros) {
            if (!numero.trim().isEmpty()) {
                 // Se extrae cada número de la cadena
                 int numeroCadena = Integer.parseInt(numero.trim());
                 // Si es negativo se añade a la lista de negativos
                if (numeroCadena < 0){
                 numerosNegativos.add(numeroCadena);
                }
                 resultado += Integer.parseInt(numero.trim());
            }
        }
        // Si hay numeros negativos se lanza una excepción
        if (numerosNegativos.size() > 0) {
            throw new RuntimeException("Valores negativos no permitidos: " + numerosNegativos.toString());
        }
        return resultado;
    }
}
```

Solución requisito 8

[JAVA TEST]

```
@Test
public final void cuandoUnoOMasNumerosSonMayoresQue1000EntoncesSeIgnoranEnLaSuma() {
    Assert.assertEquals(3+1000+6, StringCalculator8.sumar("3,1000,1001,6,1234"));
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

import java.util.ArrayList;
import java.util.List;

public class StringCalculator8 {
	public static int sumar(final String numeros) {
		String delimitador = ",|n";
		String numerosSinDelimitador = numeros;
		if (numeros.startsWith("//")) {
			int indice = numeros.indexOf("//") + 2;
			delimitador = numeros.substring(indice, indice + 1);
			numerosSinDelimitador = numeros.substring(numeros.indexOf("n") + 1);
		}
		return sumar(numerosSinDelimitador, delimitador);
	}

	@SuppressWarnings("unchecked")
	private static int sumar(final String numeros, final String delimitador) {
		int resultado = 0;
		String[] arrayNumeros = numeros.split(delimitador);
		@SuppressWarnings("rawtypes")
		List numerosNegativos = new ArrayList();
		for (String numero : arrayNumeros) {
			if (!numero.trim().isEmpty()) {
				int numeroCadena = Integer.parseInt(numero.trim());
				if (numeroCadena < 0){
					numerosNegativos.add(numeroCadena);
				} else if (numeroCadena <=1000) //Solo se suma si es menor de 1001
				 resultado += numeroCadena;
```

```
                }
            }
            if (numerosNegativos.size() > 0) {
                throw new RuntimeException("Valores negativos no permitidos: " + numerosNegativos.toString());
            }
            return resultado;
        }
}
```

Solución requisito 9

[JAVA TEST]

```
@Test
public final void delimitadoresPuedenSerDeMasDeUnCaracter() {
    Assert.assertEquals(1+2+3, StringCalculator9.sumar( "//[--]n1--2--3"));
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

import java.util.ArrayList;
import java.util.List;
public class StringCalculator9 {

	public static int sumar(final String numeros) {
	    String delimitador = ",|n";
	    String numerosSinDelimitador = numeros;
	    if (numeros.startsWith("//")) {

	        int indice = numeros.indexOf("//") + 2;
	        if (numeros.substring(indice,indice+1).equals("[")){
	               delimitador="";
	               for (int i=indice+1;!numeros.substring(i,i+1).equals("]");i++){
	                    delimitador = delimitador+numeros.substring(i, i + 1);
	               }
	        }
	        else{
	             delimitador = numeros.substring(indice, indice + 1);
	        }
	        numerosSinDelimitador = numeros.substring(numeros.indexOf("n") + 1);
	    }
	    return sumar(numerosSinDelimitador, delimitador);
	}

	@SuppressWarnings("unchecked")
	private static int sumar(final String numeros, final String delimitador) {
	    int resultado = 0;
	    String[] arrayNumeros = numeros.split(delimitador);
```

```
        @SuppressWarnings("rawtypes")
           List numerosNegativos = new ArrayList();
        for (String numero : arrayNumeros) {
            if (!numero.trim().isEmpty()) {
                 int numeroCadena = Integer.parseInt(numero.trim());
                if (numeroCadena < 0){
                 numerosNegativos.add(numeroCadena);
                } else if (numeroCadena <=1000) //Solo se suma si es menor de 1001
                 resultado += numeroCadena;
            }
        }
        if (numerosNegativos.size() > 0) {
            throw new RuntimeException("Valores negativos no permitidos: " + numerosNegativos.toString());
        }
        return resultado;
    }
}
```

Solución requisito 10

[JAVA TEST]

```
@Test
public final void cuandoHayMultiplesDelimitadores() {
    Assert.assertEquals(1+2+3, StringCalculator10.sumar("//[-][%]n1-2%3"));
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

import java.util.ArrayList;
import java.util.List;

public class StringCalculator10 {
      public static int sumar(final String numeros) {
          String delimitador = ",|n";
          String delimitador1;
          String delimitador2;
          String numerosSinDelimitador = numeros;
          if (numeros.startsWith("//")) { // contiene un delimitador distinto de , o n
              int indice = numeros.indexOf("//") + 2;
              // investigar si es mas de un delimitador
              int numCorchetes=0;
              for (int j=0;j<numeros.length();j++){
                   if (numeros.substring(j,j+1).equals("[")){
                      numCorchetes++;
                   }
              }
              int cuantosDelimitadores;
              if (numCorchetes>1){
                   cuantosDelimitadores=2;
              }
              else{
                   cuantosDelimitadores=1;
              }
              // fin investigación

              switch (cuantosDelimitadores){
```

```
            case 1: // Hay un solo delimitador
                if (numeros.substring(indice,indice+1).equals("[")){ // es un delimitador de más de 1 caracter
                  delimitador="";
                  for (int i=indice+1;!numeros.substring(i,i+1).equals("]");i++){
                       delimitador = delimitador+numeros.substring(i, i + 1);
                      }
                }
                else{
                 delimitador = numeros.substring(indice, indice + 1); // es un  delimitador de 1 caracter
                }
                break;

            case 2: //Hay 2 delimitadores de un caracter
               delimitador1 = numeros.substring(indice+1, indice+2);
                delimitador2=numeros.substring(indice+4, indice+5);
                delimitador= delimitador1+"|"+delimitador2;
                 break;
            }
            numerosSinDelimitador = numeros.substring(numeros.indexOf("n") + 1);
        }

        return sumar(numerosSinDelimitador, delimitador);
    }

    @SuppressWarnings("unchecked")
    private static int sumar(final String numeros, final String delimitador) {
        int resultado = 0;
        String[] arrayNumeros = numeros.split(delimitador);
        @SuppressWarnings("rawtypes")
           List numerosNegativos = new ArrayList();
        for (String numero : arrayNumeros) {
            if (!numero.trim().isEmpty()) {
                 int numeroCadena = Integer.parseInt(numero.trim());
                if (numeroCadena < 0){
                 numerosNegativos.add(numeroCadena);
                } else if (numeroCadena <=1000) //Solo se suma si es menor de 1001
                 resultado += numeroCadena;
            }
        }
        if (numerosNegativos.size() > 0) {
            throw new RuntimeException("Valores negativos no permitidos: " + numerosNegativos.toString());
        }
        return resultado;
}
```

Solución requisito 11

[JAVA TEST]

```
@Test
public final void cuandoHayMultiplesDelimitadoresDeUnCaracter() {
    Assert.assertEquals(1+2+3+4, StringCalculator11.sumar("//[-][%][#]n1-2%3#4"));
}

@Test
public final void cuandoHayDosDelimitadoresDeVariosCaracteres() {
    Assert.assertEquals(1+2+3, StringCalculator11.sumar("//[--][%%]n1--2%%3"));
}

@Test
public final void cuandoHayMultiplesDelimitadoresDeVariosCaracteres() {
    Assert.assertEquals(1+2+3+4, StringCalculator11.sumar("//[--][%%][##]n1--2%%3##4"));
}
```

[JAVA IMPLEMENTATION]

```
package kataTDD;

import java.util.ArrayList;
import java.util.List;

public class StringCalculator11 {

    public static int sumar(final String numeros) {

        String delimitador = ",|n";
        String numerosSinDelimitador = numeros;

        if (numeros.startsWith("//")) { // contiene un delimitador distinto de , o n

            int indice = numeros.indexOf("//") + 2;
            // investigar si es mas de un delimitador
            int numCorchetes=0;
            for (int j=0;j<numeros.length();j++){
                 if (numeros.substring(j,j+1).equals("[")){
                    numCorchetes++;
```

```
            }
        }
        int cuantosDelimitadores;
        if (numCorchetes>1){
            cuantosDelimitadores=2;
        }
        else{
            cuantosDelimitadores=1;
        }
        // fin investigación delimitador

        switch (cuantosDelimitadores){
        case 1: // Hay un solo delimitador
            if (numeros.substring(indice,indice+1).equals("[")){ // es un delimitador de más de 1 caracter
              delimitador="";
              for (int i=indice+1;!numeros.substring(i,i+1).equals("]");i++){
                  delimitador = delimitador+numeros.substring(i, i + 1);
                 }
            }
            else{
             delimitador = numeros.substring(indice, indice + 1); // es un  delimitador de 1 caracter
            }
            break;

        case 2:
             // Hay varios delimitadores
             String restoCadena=numeros;
             int desdeCorchete;
             int hastaCorchete;
             delimitador="";
             for (int k=0; k<numCorchetes; k++){
                    desdeCorchete = restoCadena.indexOf('[');
                    hastaCorchete = restoCadena.indexOf(']');
                    delimitador = delimitador + restoCadena.substring(desdeCorchete+1, hastaCorchete) + "|";
                    restoCadena = restoCadena.substring(hastaCorchete+1);
             }
          System.out.println(delimitador);
            break;

        }

        numerosSinDelimitador = numeros.substring(numeros.indexOf("n") + 1);
    }
```

```
            return sumar(numerosSinDelimitador, delimitador);
        }

        @SuppressWarnings("unchecked")
        private static int sumar(final String numeros, final String delimitador) {
            int resultado = 0;
            String[] arrayNumeros = numeros.split(delimitador);
            @SuppressWarnings("rawtypes")
               List numerosNegativos = new ArrayList();
            for (String numero : arrayNumeros) {
                if (!numero.trim().isEmpty()) {
                     int numeroCadena = Integer.parseInt(numero.trim());
                    if (numeroCadena < 0){
                     numerosNegativos.add(numeroCadena);
                    } else if (numeroCadena <=1000) //Solo se suma si es menor de 1001
                     resultado += numeroCadena;
                }
            }
            if (numerosNegativos.size() > 0) {
                throw new RuntimeException("Valores negativos no permitidos: " + numerosNegativos.toString());
            }
            return resultado;
        }
}
```