# UNIDAD 6

Contenido

[UNIDAD 6 1](#_Toc99645970)

[1. UT6 – PRÁCTICA 01 2](#_Toc99645971)

[1. Diagrama de casos de uso 2](#_Toc99645972)

[1. Caja Negra 2](#_Toc99645973)

[1.1. Resumen de la teoría de caja negra 2](#_Toc99645974)

[1.2. Ejercicio 1 - Aplicación bancaria 2](#_Toc99645975)

[1.3. Ejercicio 2 – Aplicación para sorteo de vivienda 2](#_Toc99645976)

[1.4. Ejercicio 3 – Inversión de un número 2](#_Toc99645977)

[1.5. Ejercicio 4 – Hora 3](#_Toc99645978)

## UT6 – PRÁCTICA 01

### Diagrama de casos de uso

# Caja Negra

## Resumen de la teoría de caja negra

**Determinar las clases equivalencia:**

* Rango: una válida y dos no válidas.
* Valor específico: una válida y una no válida.
* Conjunto de valores:
  + Si cada elemento se trata igual, una válida y otra no válida.
  + Si cada elemento se trata diferente, una válida por cada elemento y otra no válida
* Lógica: una válida y otra no válida.

**Diseñar los casos de prueba**

* **Para las clases válidas:** diseñar el menor número de casos posibles. En cada caso de prueba intentamos abarcar el mayor número de cláses válidas.
* **Para las clases no válidas:** diseñaremos un caso de prueba por cada clase no válida, introduciendo el valor de la clase no válida a probar y en las demás entradas valores correspondientes a cláses válidas.

## Ejercicio 1 - Aplicación bancaria

**Datos de entrada:**

* Código de área: número de 3 dígitos que no empieza por 0 ni por 1
* Nombre de identificación de operación: 6 caracteres
* Órdenes posibles: “cheque”, “depósito”, “pago factura”, “retirada de fondos”

**Determinar las clases equivalencia:**

* Rango:
  + 1 clase válida: 200<código<999
  + 2 clases no válidas: código<200; código>999
* Valor específico:
  + 1 clase válida: 6 caracteres
  + ½ clases no válidas: más de 6 caracteres; menos de 6 caracteres
* Conjunto de valores:
  + 4 clases válidas: 4 órdenes válidas
  + 1 clase no válida: orden no válida
* Lógica (formato datos):
  + 1 clase válida: número
  + 1 clase no válida: no es número

|  |  |  |
| --- | --- | --- |
| **Datos de Entrada** | **Clases Válidas** | **Clases No Válidas** |
| Código de área | (1) 200 <= código <= 999 | (2) código < 200  (3) código > 999  (4) no es número |
| Identificación | (5) 6 caracteres | (6) menos de 6 caracteres  (7) más de 6 caracteres |
| Orden | (8) “cheque”  (9) “depósito”  (10) “pago factura”  (11) “retirada de fondos” | (12) ninguna orden válida |

|  |  |  |  |
| --- | --- | --- | --- |
| **CASOS DE PRUEBA VÁLIDOS** | | | |
| **Código** | **Identificación** | **Orden** | **Clases cubiertas** |
| 301 | aaaaaa | “Cheque” | 1, 5, 8 |
| 200 | numero | “depósito” | 1, 5, 9 |
| 999 | identi | “pago factura” | 1, 5, 10 |
| 201 | codigo | “retirada de fondos” | 1, 5, 11 |

|  |  |  |  |
| --- | --- | --- | --- |
| **CASOS DE PRUEBA NO VÁLIDOS** | | | |
| **Código** | **Identificación** | **Orden** | **Clases cubiertas** |
| 199 | aaaaaa | cheque | 2, 5, 8 |
| 1000 | aaaaaa | cheque | 3, 5, 8 |
| codigo | aaaaaa | cheque | 4, 5, 8 |
| 200 | cinco | cheque | 1, 6, 8 |
| 200 | sietecar | cheque | 1, 7, 8 |
| 200 | aaaaaa | otro | 1, 5, 12 |

## Ejercicio 2 – Aplicación para sorteo de vivienda

El programa debe guardar las puntuaciones en el baremo de una serie de solicitantes de vivienda, y nos debe decir si han resultado beneficiados o no con la adjudicación de una vivienda subvencionada por el Ayuntamiento (La adjudicación se materializa para puntuaciones en el baremo iguales o superiores a 5).

El programa admite como entrada el DNI y un número entre 0 y 10 sin decimales.

Datos de entrada:

* DNI: 8 números enteros y una letra

Sabemos que el DNI se compone de forma general de 8 cifras numéricas y una letra, aunque algunos DNI antiguos tienen 7 cifras. Es decir, en total la longitud válida será 8 o 9 caracteres. Esto genera estas clases:

**2 clases válidas: una con 8 caracteres y otra con 9**

**2 clases no válidas: una con 7 o menos caracteres y otra con 10 o más.**

Si nos fijamos en la combinación de letras u números las clases serán:

**2 clases válidas: una con 7 cifras y una letra y con 8 y letra, coinciden con las anteriores por lo que no hace falta repetirlas.**

**2 clases no válidas: cualquier combinación diferente de 8 y 9 cifras.**

* Baremo:

Rango

1 válida: con un valor entre 0 y 10

2 con datos no válidos: una con un valor negativo y otra con un valor superior a 10

Si nos fijamos en la lógica, tendríamos dos clases válidas:

0<x<5 -> NO se adjudica la vivienda

5<=x<=10 Se adjudica la vivienda

Si nos fijamos en los decimales de la puntuación en el baremo tenemos:

* Una clase no válida: un número entre 9 y 10 con decimales.

Podríamos pensar en una clase no válida con un número superior a 10 y decimales pero no es necesaria porque queda cubierta en el apartado anterior. Tampoco necesitamos una clase válida sin decimales porque eso coincide con la clase

**Determinar las clases equivalencia:**

* Rango:
  + 1 clase válida: 200<código<999
  + 2 clases no válidas: código<200; código>999
* Valor específico:
  + 1 clase válida: 6 caracteres
  + ½ clases no válidas: más de 6 caracteres; menos de 6 caracteres
* Conjunto de valores:
  + 4 clases válidas: 4 órdenes válidas
  + 1 clase no válida: orden no válida
* Lógica (formato datos):
  + 1 clase válida: número
  + 1 clase no válida: no es número

|  |  |  |
| --- | --- | --- |
| **Datos de Entrada** | **Clases Válidas** | **Clases No Válidas** |
| Código de área | (1) 200 <= código <= 999 | (2) código < 200  (3) código > 999  (4) no es número |
| Identificación | (5) 6 caracteres | (6) menos de 6 caracteres  (7) más de 6 caracteres |
| Orden | (8) “cheque”  (9) “depósito”  (10) “pago factura”  (11) “retirada de fondos” | (12) ninguna orden válida |

## Ejercicio 2 – Aplicación para sorteo de vivienda

El programa debe guardar las puntuaciones en el baremo de una serie de solicitantes de vivienda, y nos debe decir si han resultado beneficiados o no con la adjudicación de una vivienda subvencionada por el Ayuntamiento (La adjudicación se materializa para puntuaciones en el baremo iguales o superiores a 5).

El programa admite como entrada el DNI y un número entre 0 y 10 sin decimales.

|  |  |  |
| --- | --- | --- |
| **Condición** | **Clases válidas** | **Clases No Válida** |
| DNI | 8 cifras más de una letra **(1)** | <9 caracteres **(2)**  >9 caracteres **(3)**  9 caracteres que no sean 8 cifras más letra **(4)** |
| Puntuación Baremo | Un valor x con rango 0<=x<5 **(5)**  Un valor x con rango 5<=x <=10 **(6)** | Un valor <0 **(7)**  Un valor >10**(8)**  Un valor con decimales**(9)** |

|  |  |  |
| --- | --- | --- |
| **Casos de prueba Válidos (DNI + Nota)** | **Clases probadas** | **Resultado esperado** |
| 1234567A 4 | 1 y 5 | Vivienda no adjudicada |
| 12345678A 7 | 1 y 6 | Vivienda adjudicada |

|  |  |  |
| --- | --- | --- |
| **Casos de prueba No Válidos (DNI + Nota)** | **Clases probadas** | **Resultado esperado** |
| 1234567 4 | 2 y 5 | Error de formato DNI |
| 0123456789 4 | 3 y 5 | Error de formato DNI |
| 012345678 4 | 4 y 5 | Error de formato DNI |
| 12345678A -1 | 1 y 7 | Error en puntuación |
| 01234567A 11 | 1 y 8 | Error en puntuación |
| 01234567A 5.5 | 1 y 9 | Error en puntuación |

## Ejercicio 3 – Inversión de un número

Un programa recibe como entrada un número entero y positivo de mínimo 2 cifras y de máximo 9 cifras y devuelve el número resultante de invertir sus cifras.

Si no se introduce un valor acorde a lo descrito rango, etc.), el módulo devolverá el valor “error”. No olvidar el análisis de valores límite (por ejemplo: flotantes y/o caracteres, valores fuera de rango)

**Datos de entrada:**

* **Número correcto:**
* **2 clases válidas: una con 8 caracteres y otra con 9**
* **2 clases no válidas: una con 7 o menos caracteres y otra con 10 o más.**
* Si nos fijamos en la combinación de letras u números las clases serán:
* **2 clases válidas: una con 7 cifras y una letra y con 8 y letra, coinciden con las anteriores por lo que no hace falta repetirlas.**
* **2 clases no válidas: cualquier combinación diferente de 8 y 9 cifras.**

## Ejercicio 4 – Hora

Tenemos un pequeño módulo que lee una hora en formato de hh:mm e indica si la hora es correcta.

Se anexa a continuación la pantalla:

Aplicación

Descripción generada automáticamente con confianza media

Si no se introduce un valor acorde a lo descrito (por ejemplo: flotantes y/o caracteres, valores fuera de rango, etc.), el módulo devolverá el valor “error”. Genere la tabla de clases de equivalencia y los casos de prueba (no olvide el análisis de valores límite).

Solución:

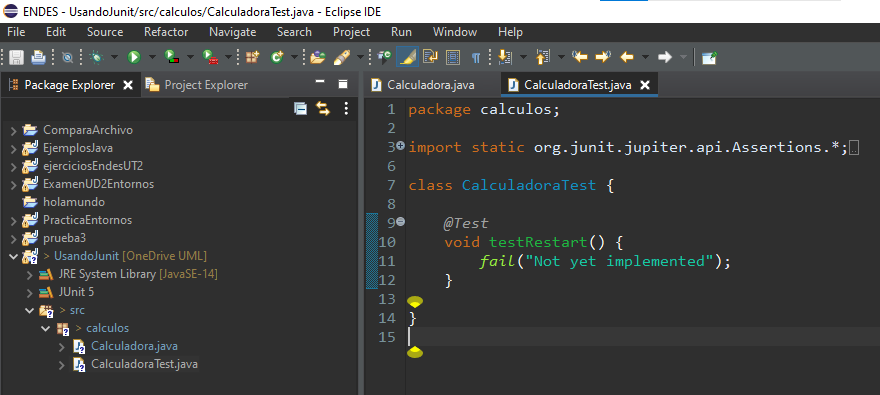
n = número de parámetros

A = conjunto de valores permitidos: los números naturales más el cero. A = N U {0}.

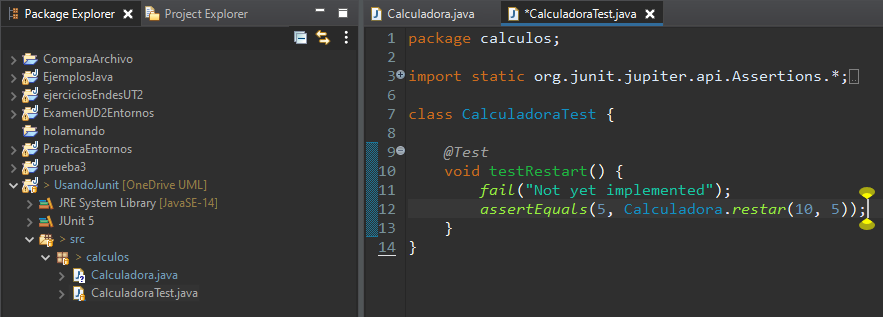
## PRÁCTICA 01 DE TEMA JUNIT

### EJERCICIO 1

Hemos hecho un paquete cálculos y dos clases, una llamada Calculadora y otra CalculadoraTest. Además, dentro de CalculadoraTest hemos añadido un test con un mensaje de texto de error.

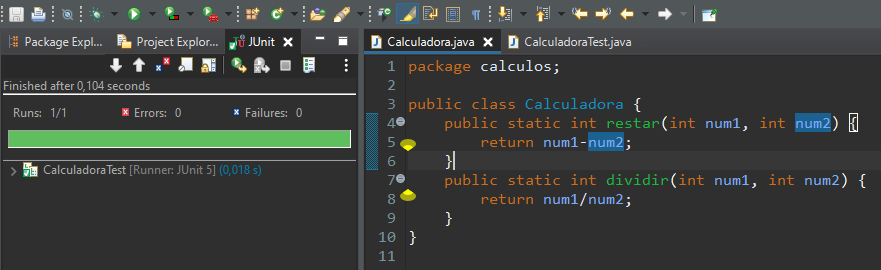


También, ponemos un ejemplo con assertEquals donde ponemos un ejemplo de un caso de la calculadora para ver si funciona.

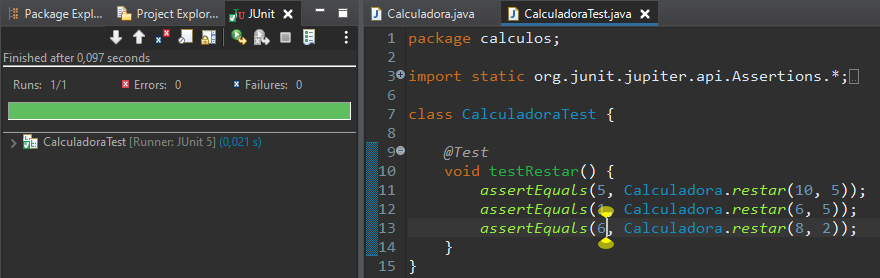


Dio error y lo hemos corregido invirtiendo el orden del return.

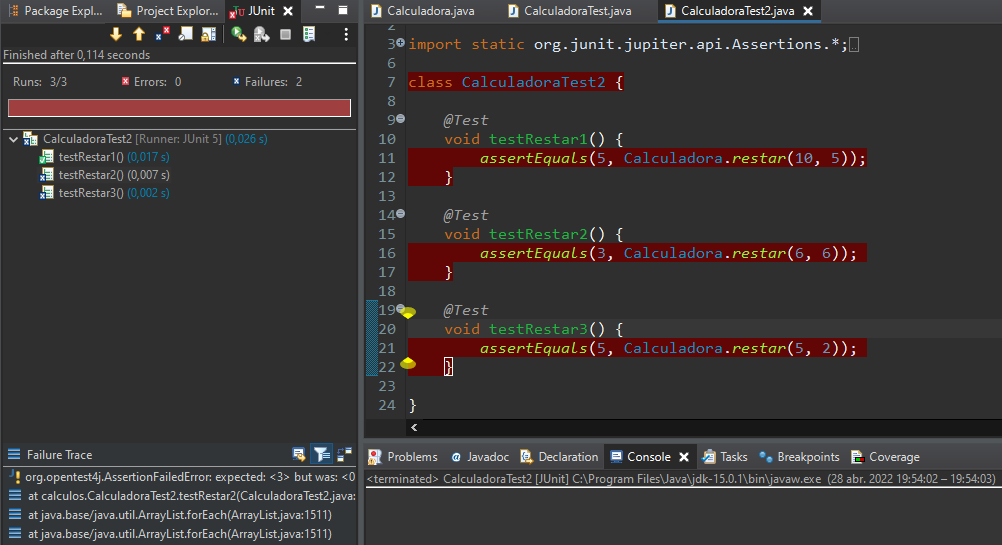
Con el JUnit se comprueba y no da errores.



Hemos añadido dos asertos más con más otros 2 casos.

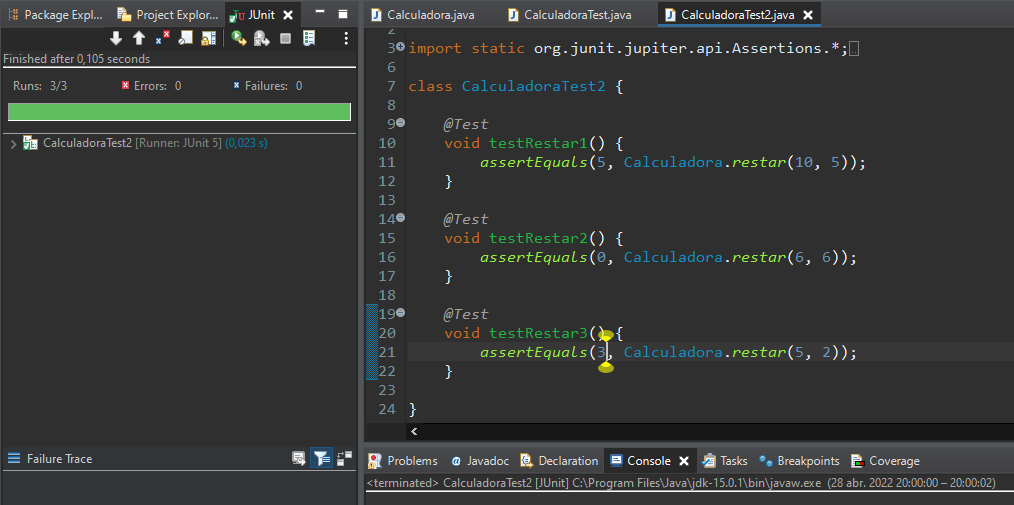


Con el JUnit hemos comprobado que los casos que hemos añadido dan un resultado incorrecto. Procedemos a arreglarlo en la próxima captura.



### EJERCICIO 2

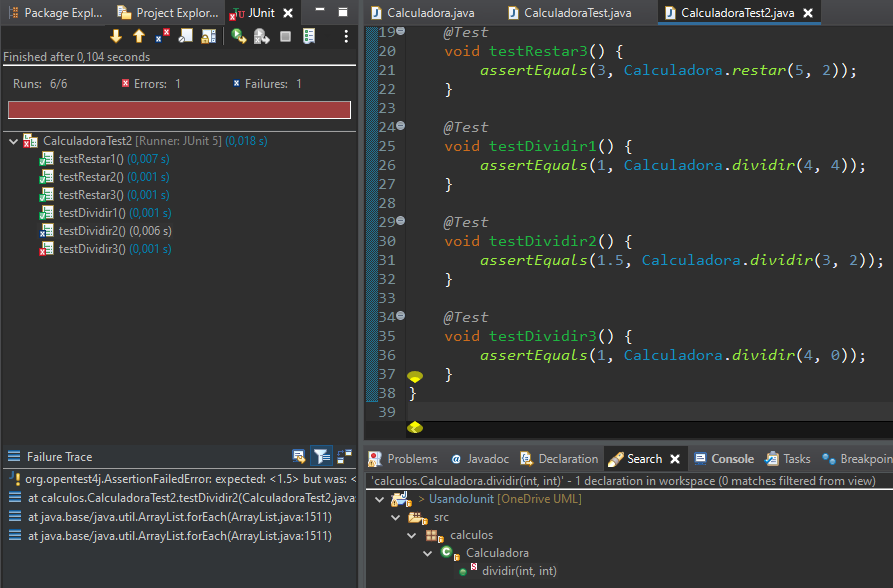
En este ejercicio hemos puesto una captura con la solución. Como se puede comprobar, después de correr el JUnit no da errores.



### EJERCICIO 3

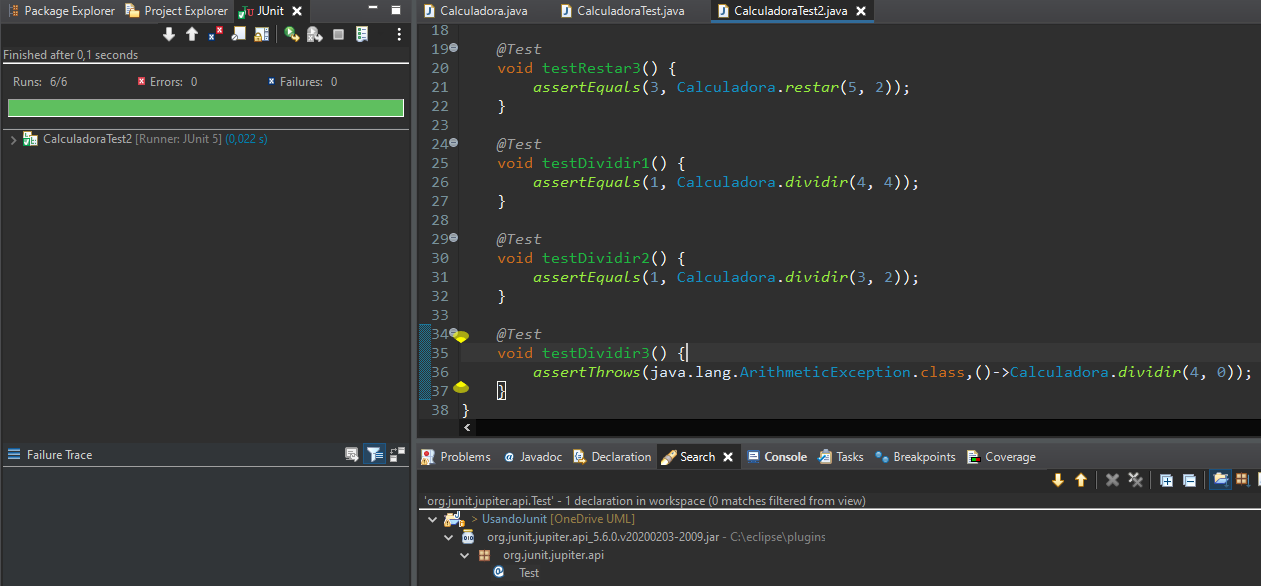
En este ejercicio añadiremos más asertos. En esta ocasión, de dividir.

Todos los casos salen positivos excepto el tercero que no se puede realizar. En el siguiente ejercicio veremos como corregirlo.

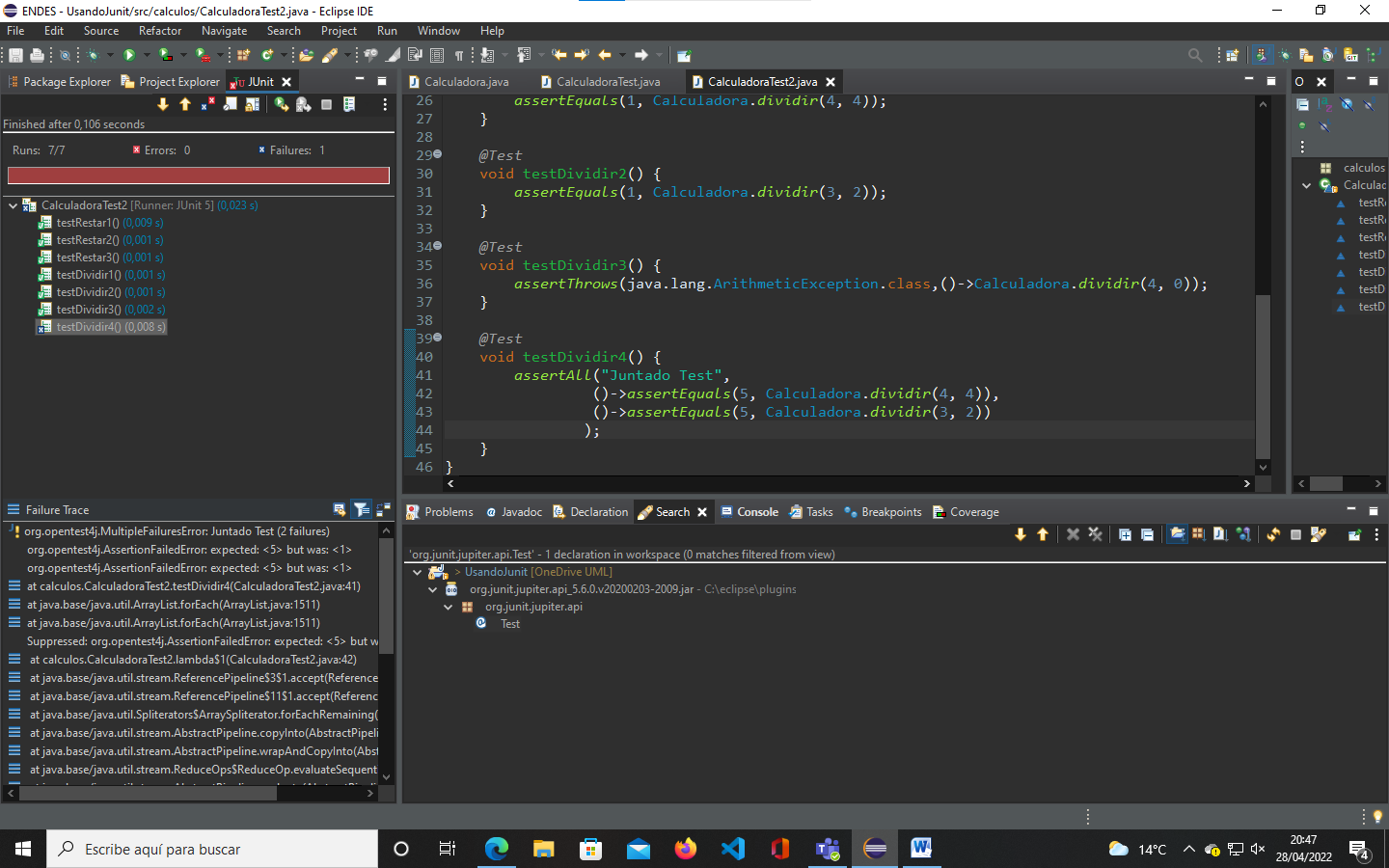


### EJERCICIO 4

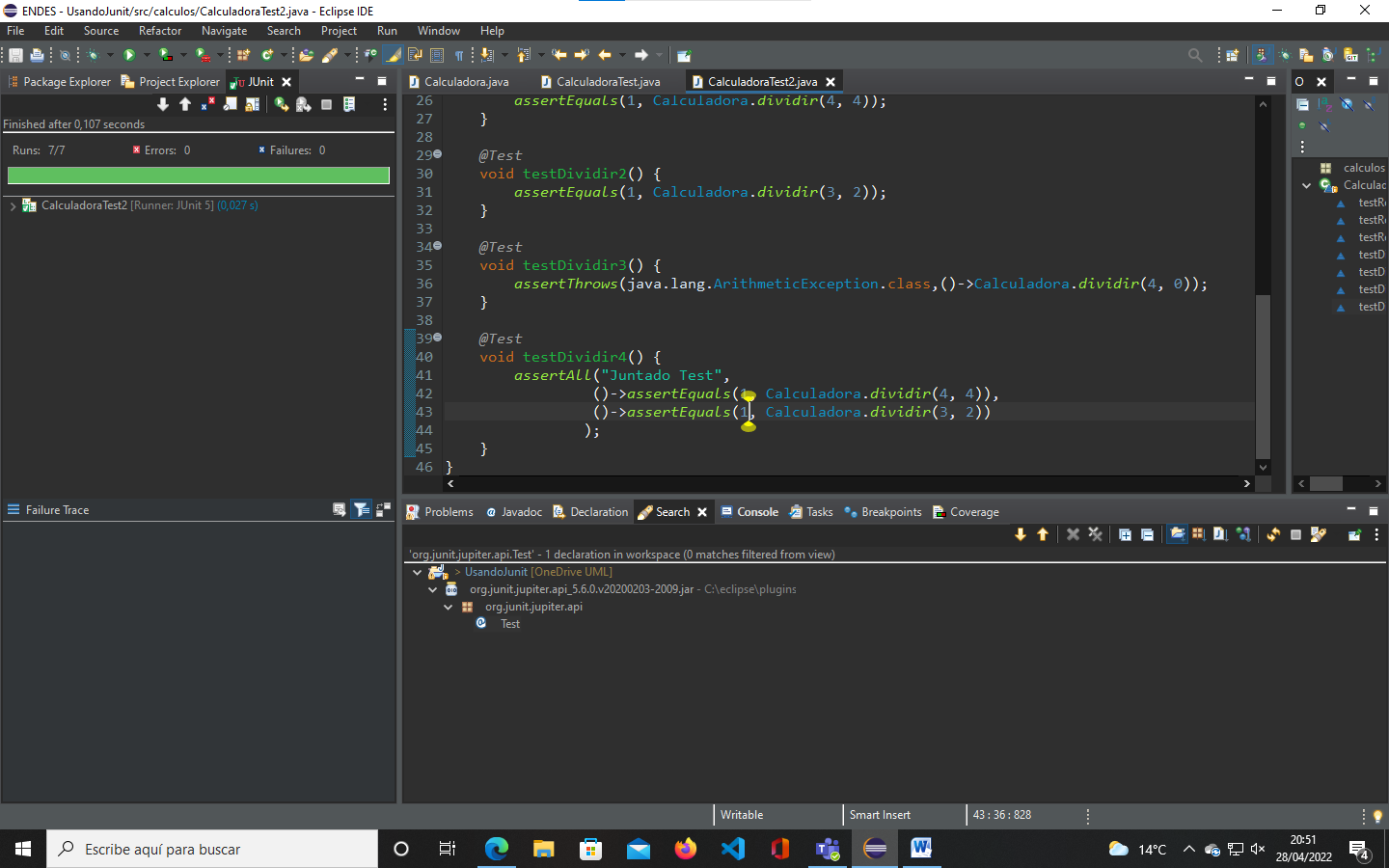
En este ejercicio hemos añadido una excepción en el void testDividir3(). Estamos poniendo un error esperado mediante la expresión assertThrows[…]

¡

### EJERCICIO 5

Añadimos otro testDividir con un nuevo aserto llamado AsertAll donde podremos realizar varios test en un mismo assert.

En esta nueva captura hemos resuelto los test que salían como error.



### EJERCICIO 6

En este ejercicio hemos añadido otro test con un nuevo assert, AssertTrue. Con este aserto forzamos que el resultado sea un booleano

