Testing

# Unit Tests

## Unit tests de clase Matriz.

Clase: UnitTestMatriz

SumaMatrices

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método suma de matrices funcione correctamente |
| **Ejecución** | * Se crean 3 matrices de 1x1. * A con valor 1, B con valor 2 y C con valor 3 * Se hace un assertArrayEquals con C y la suma de A y B |
| **Resultado esperado** | 3 (Pass) |
| **Pass/Fail** | **PASS** |

RestaMatrices

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método resta de matrices funcione correctamente |
| **Ejecución** | * Se crean 3 matrices de 1x1. * A con valor 5, B con valor 2 y C con valor 3 * Se hace un assertArrayEquals con C y la resta de A y B |
| **Resultado esperado** | 3 (Pass) |
| **Pass/Fail** | **PASS** |

MultAndMatrices

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método de una Multiplicacion miembro a miembro funcione correctamente |
| **Ejecución** | * Se crean 3 matrices de 1x1. * A con valor 10, B con valor 2 y C con valor 20 * Se hace un assertArrayEquals con C y la multAnd de A y B |
| **Resultado esperado** | 20 (Pass) |
| **Pass/Fail** | **PASS** |

Transpose

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método que transpone una matriz funcione |
| **Ejecución** | * Se crean 2 matrices una 1x3 y otra 3x1. * A con 3 valores y B con los valores de A transpuestos * Se hace un assertArrayEquals con B y la transpuesta de A |
| **Resultado esperado** | Transpuesta de A (Pass) |
| **Pass/Fail** | **PASS** |

Clear

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método de limpiar una matriz funcione correctamente |
| **Ejecución** | * Se crean 2 matrices de 3x1. * A con ciertos valores y B matriz con todos 0. * Se hace un assertArrayEquals con B y A luego de un clear. |
| **Resultado esperado** | A con todos elementos 0 |
| **Pass/Fail** | **PASS** |

SetIdentity

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método de crear una matriz identidad funcione |
| **Ejecución** | * Se crean 2 matrices 3x3 * A la matriz A se le aplica el método setIdentity * A la matriz B se la crea como identidad y se las compara. |
| **Resultado esperado** | Matriz A como identidad |
| **Pass/Fail** | **PASS** |

Producto por un escalar

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método de productor por un escalar funcione |
| **Ejecución** | * Se crean 2 matrices de 3x1. * A con valores 5, -2,10 y B con 15,-6,30 * Se hace un assertArrayEquals con B y A productor por 3. |
| **Resultado esperado** | A con valores 15, -6, 30 |
| **Pass/Fail** | **PASS** |

Es Nula

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método es nula funcione correctamente |
| **Ejecución** | * Se crean 1 matriz de 3x1. * Se le hace productor por un escalar con 0 * Se hace un assertTrue con el método es Nula |
| **Resultado esperado** | True |
| **Pass/Fail** | **PASS** |

## Unit Test clase Cola

meterEnCola

|  |  |
| --- | --- |
| **Descripción del Test** | Comprueba que el método meter en cola funcione |
| **Ejecución** | * Se crea un objeto Cola, y dos hilos que van a ser agregados * Se crean dos runnables que llaman al método meterencola * Se crea una Queue local y se la compara |
| **Resultado esperado** | Queue local igual a la campo Queue de la clase Cola. |
| **Pass/Fail** | **PASS** |

testColaDormidos

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que la cola fifo de hilos dormidos funcione |
| **Ejecución** | * Misma lógica que el test de meterEnCola * La diferencia esta en el Assert * Se compara el primer hilo metido en cola con el hilo obtenido en a través del método obtenerProceso. |
| **Resultado esperado** | Iguales |
| **Pass/Fail** | **PASS** |

testColaDormidosNegativo

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que la cola fifo de hilos dormidos funcione |
| **Ejecución** | * Igual que el test positivo * La diferencia esta en el Assert * Ahora se compara que el hilo obtenido sea distinto al segundo |
| **Resultado esperado** | Distintos hilo obtenido con el hilo metido en segundo lugar. |
| **Pass/Fail** | **PASS** |

## Unit Tests a leerTxtFile

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que el método leerTxtFile funcione correctamente |
| **Ejecución** | * Se guarda en una matriz de enteros la lectura de un archivo txt * Se crea una matriz con el resultado esperado * Se hace un assertArrayEquals entre ambas matrices |
| **Resultado esperado** | Iguales |
| **Pass/Fail** | **PASS** |

## Unit Tests Politicas

testPoliticas

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que las políticas fifo funcionen correctamente. |
| **Ejecución** | * Creo una matriz que simula la matriz de tr sensibilizadas * A un objeto políticas se le llama el método insertFifo con 3 nros. * Se compara el primero valor ingresado y el valor devuelto por   el método cual de políticas. |
| **Resultado esperado** | Que el método cual devuelva el primer elemento ingresado (fifo) |
| **Pass/Fail** | **PASS** |

testPoliticasNegativo

|  |  |
| --- | --- |
| **Descripción del Test** | Chequea que las políticas fifo funcionen correctamente. |
| **Ejecución** | * Creo una matriz que simula la matriz de tr sensibilizadas * A un objeto políticas se le llama el método insertFifo con 3 nros. * Se compara el segundo valor ingresado y el valor devuelto por   el método cual de políticas. |
| **Resultado esperado** | Que el método cual devuelva un numero distinto al segundo ingresado. |
| **Pass/Fail** | **PASS** |

## Test de P-Invariantes

|  |  |
| --- | --- |
| **Descripción del Test** | Test de los p-invariantes. |
| **Ejecución** | * Se inicializa la red de Petri con su marcado e incidencia * Se le pasa un marcado distinto del inicial. * Se hace assertTrue o assertFalse al método   comprobarPinvariante depediendo si el marcado es uno valido o no |
| **Resultado esperado** | True |
| **Pass/Fail** | **PASS** |

Distintos Marcados pasados:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Marcado no valido -> Devuelve false y el tast pasa correctamente

Marcado 2:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | 0 | 26 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 26 | 0 | 1 | 0 | 1 | 0 | 1 |

Marcado valido -> Devuelve true y pasa el test correctamente.

Marcado 3:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 26 | 0 | 0 | 26 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 26 | 0 | 1 | 1 | 1 | 1 | 1 |

Marcado No Valido -> Devuelve false y el test pasa correctamente

Marcado 4:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 25 | 0 | 0 | 24 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | 1 | 0 | 0 |

Marcado Valido -> Devuelve true y el test pasa correctamente.

## Unit test T-Invariante

|  |  |
| --- | --- |
| **Descripción del Test** | Test de los t-invariantes. |
| **Ejecución** | * Se inicializa la red de Petri, el gestor de monitor y las colas. * Ahora los actores se construyen limitando la cantidad de veces   que ejecutan su secuencia.   * Se compara un contador de t-invariantes obtentido después   de que los actores terminaran con sus secuencias y el vector esperado  que se relaciona con la cantidad de piezas producidas. |
| **Resultado esperado** | Que ambos vectores sean iguales. |
| **Pass/Fail** | **PASS** |

Hilo 1 ejecuta -> 13 11 0 12 y le pasamos 15 repeticiones

Hilo 2 ejecuta -> 18 y le pasamos 35 repeticiones

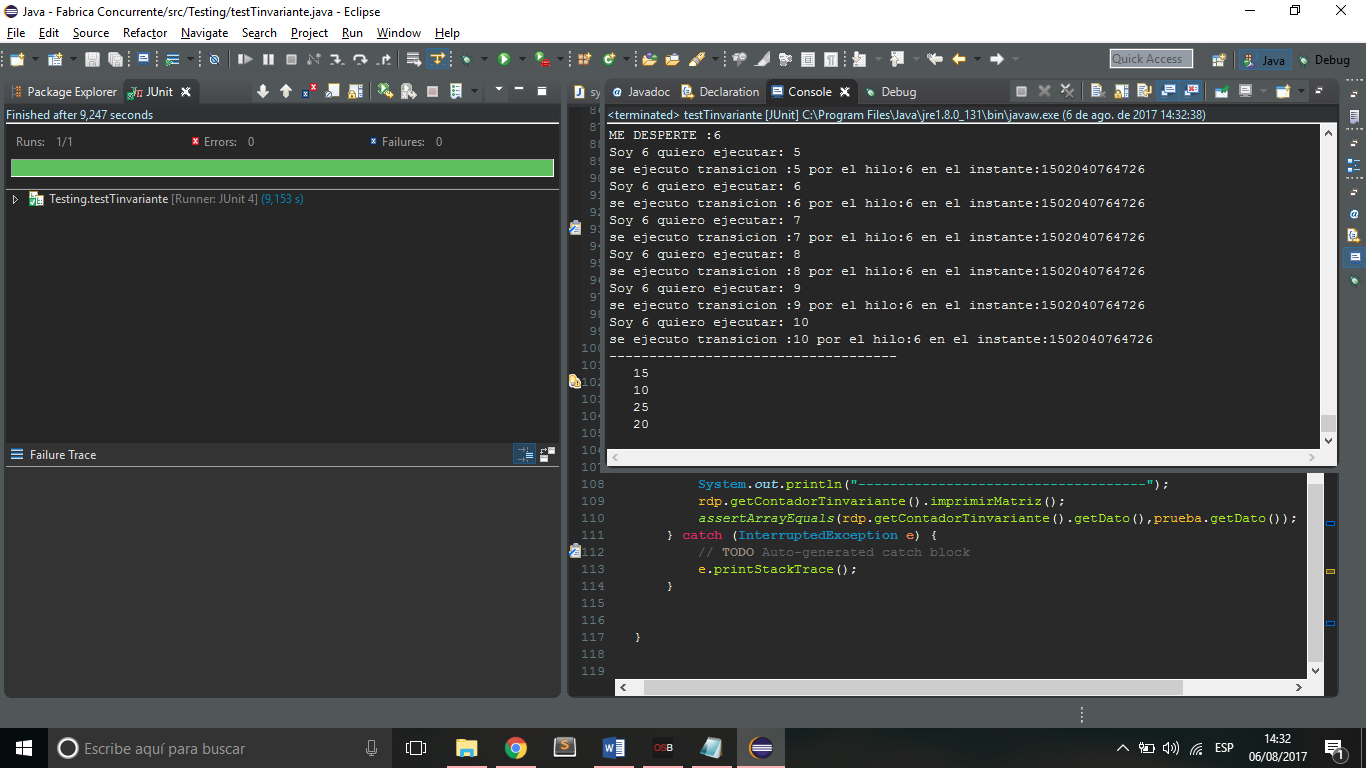
Hilo 3 ejecuta -> 19 y le pasamos 35 repeticiones

Hilo 4 ejecuta -> 1 2 3 4 y le pasamos 25 repeticiones

Hilo 5 ejecuta -> 14 15 y le pasamos 10 repeticiones

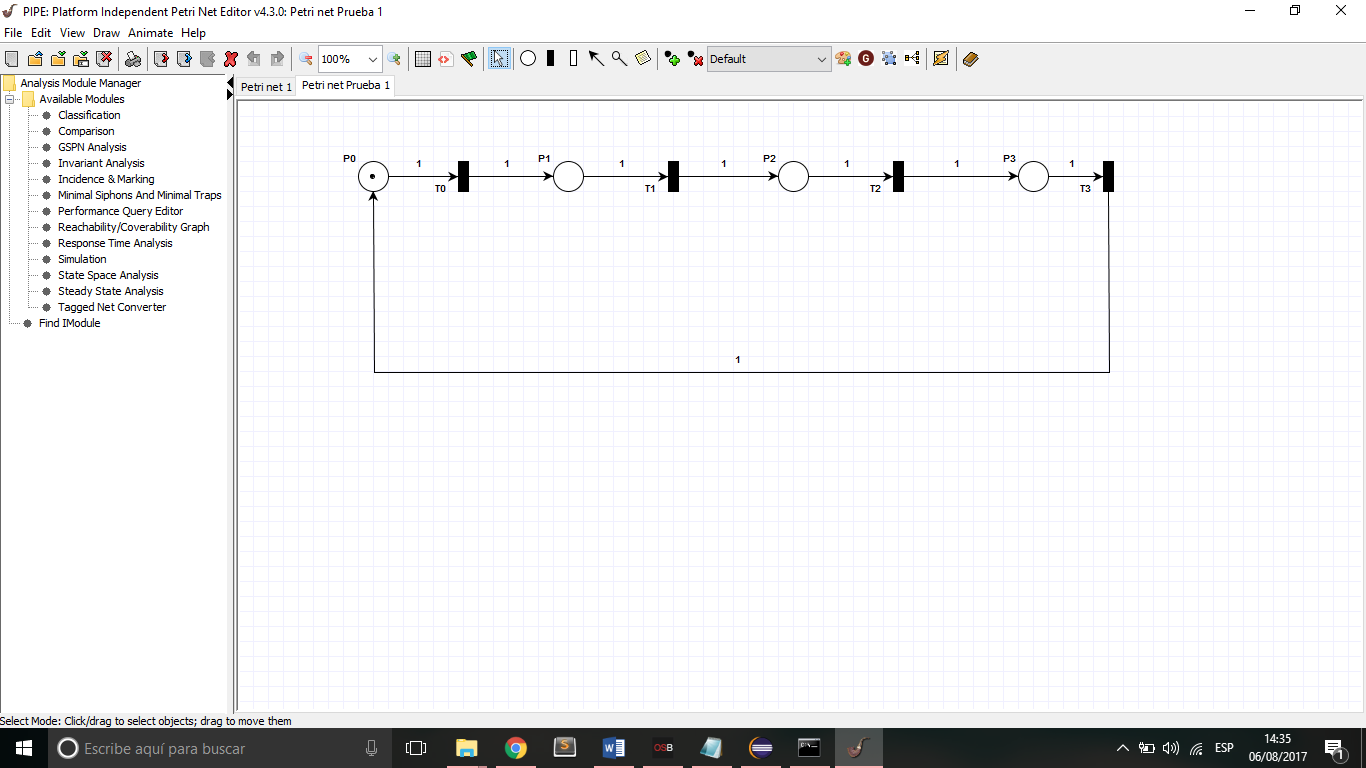
Hilo 6 ejecuta -> 16 17 y le pasamos 10 repeticiones

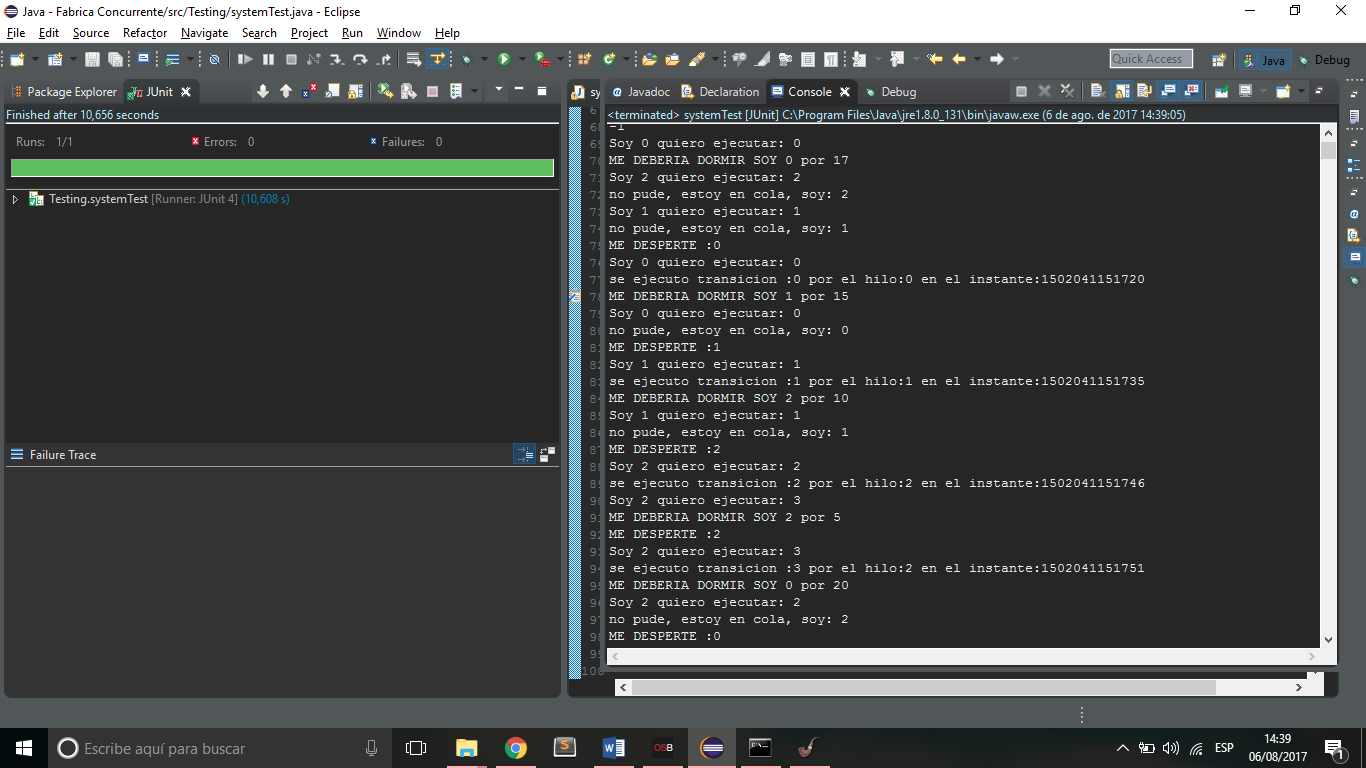
Hilo 7 ejecuta -> 5 6 7 8 9 10 y le pasamos 20 repeticiones

Y deberíamos obtener [15 10 25 20]

# System Test

Se realiza ahora un system test utilizando la siguiente red:

Con 4 hilos, uno para cada transición.



Y se ve claramente una correcta ejecución de la red.