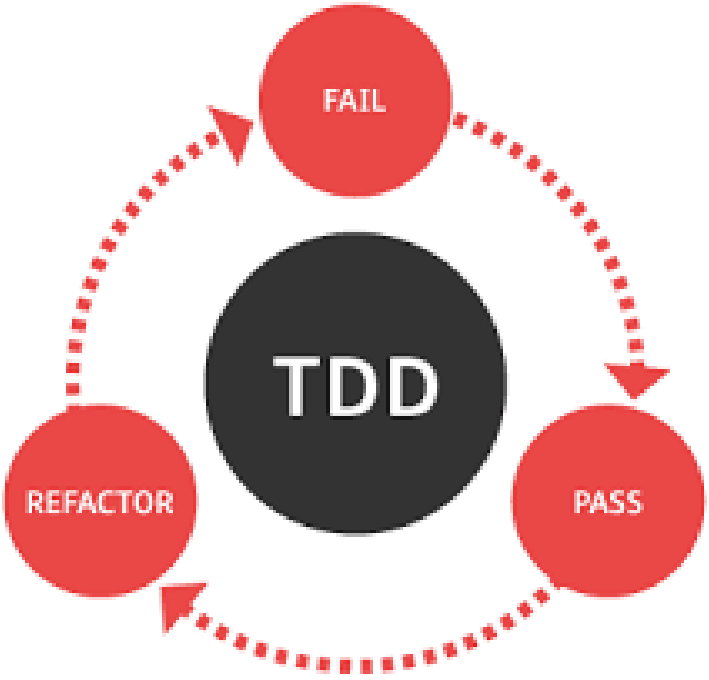
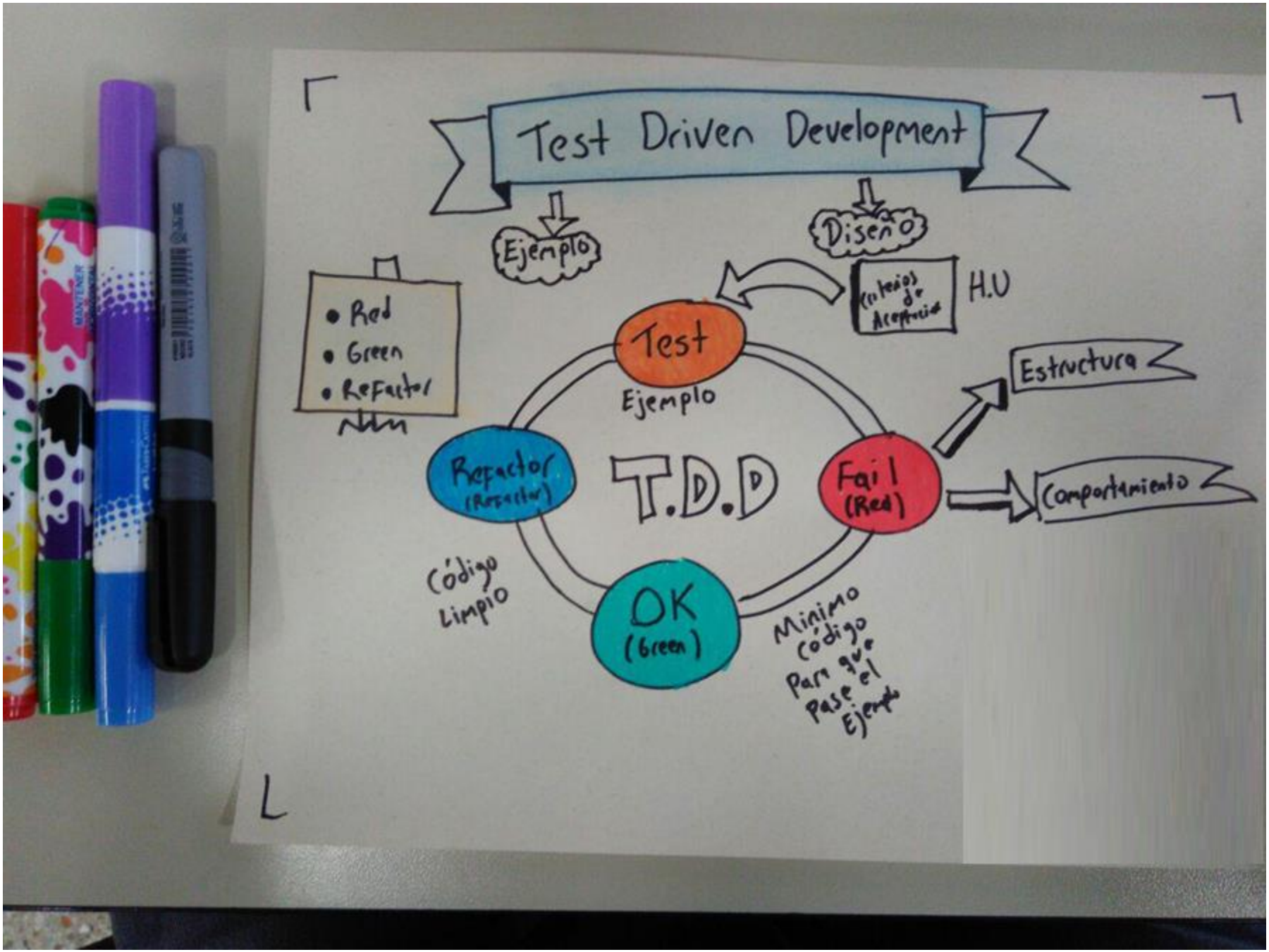
T.D.D




Test Driven Development
Ejemplo
Diseño
H.U
• Red
• Green
• Refactor
Test
Ejemplo
Refactor
T.D.D
Fail
(Red)
OK
(Green)
Estructura
Comportamiento
Código
Limpio
Mínimo
Código
Para que
Pase el
Ejemplo

La maquina de Café


NESCAFÉ
NESCAFÉ
NESCAFÉ

“El objetivo es construir una maquina dispensadora de café utilizando T.D.D.”

1) Seleccionar el tamaño de vaso de café.

Vaso pequeño -> 3 Oz de café.

Vaso Mediano -> 5 Oz de café.

Vaso Grande -> 7 Oz de café.

2) Seleccionar las cucharadas de azúcar.

3) Recoger vaso.

Historia de Usuario

Como: Consumidor de café

Deseo: Tomar un vaso de café

Para: Mitigar el sueño.

Criterios de aceptación

-Podre seleccionar entre 3 tamaños de vaso (Pequeño , Mediano ,Grande).

-Podre seleccionar la cantidad de azúcar.

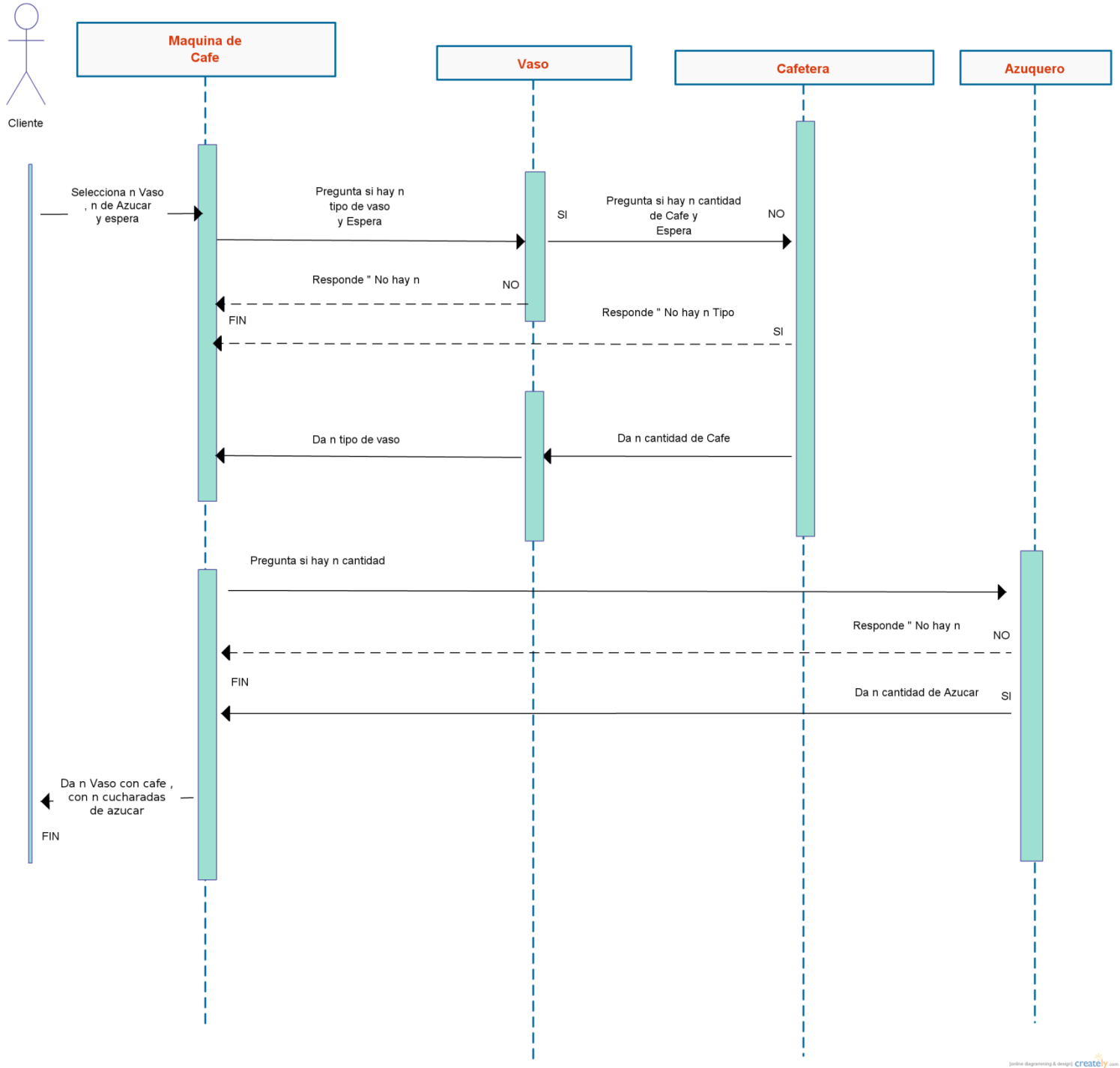
- Mostrar mensaje si no existe vasos o azúcar o café

Conformar equipo

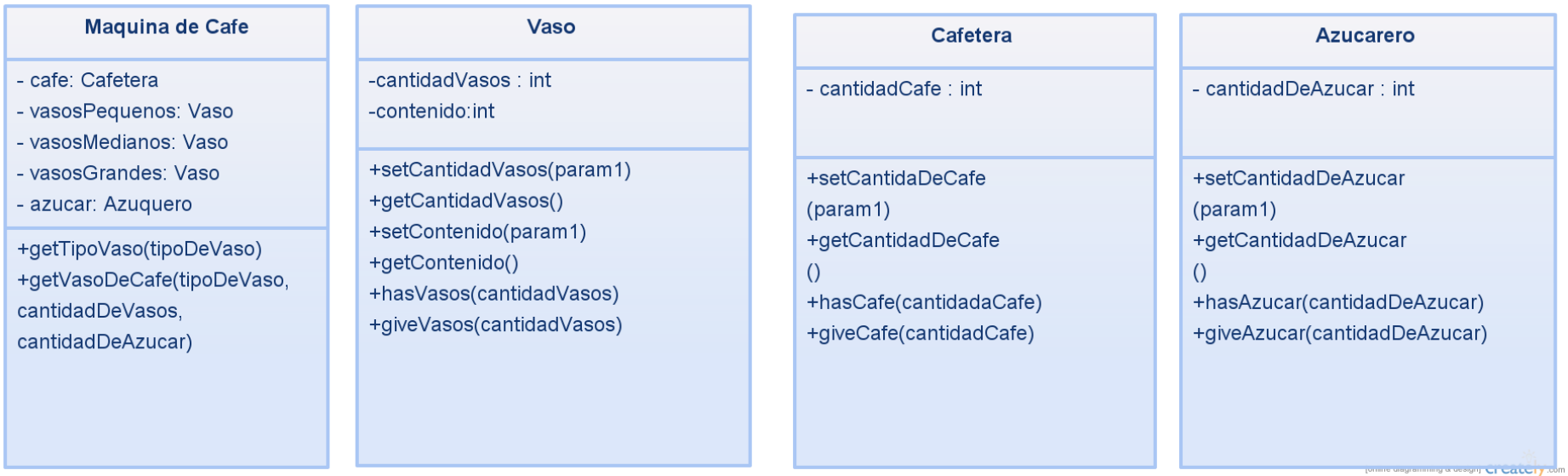
Vasos

Cafetera

Azucarero


Cliente
Maquina de Cafe
Vaso
Cafetera
Azuquero
Selecciona n Vaso , n de Azucar y espera
Pregunta si hay n tipo de vaso y Espera
SI
Pregunta si hay n cantidad de Cafe y Espera
NO
Responde " No hay n
NO
FIN
Responde " No hay n Tipo
SI
Da n tipo de vaso
Da n cantidad de Cafe
Pregunta si hay n cantidad
Responde " No hay n
NO
FIN
Da n cantidad de Azucar
SI
Da n Vaso con cafe , con n cucharadas de azucar
FIN

Comencemos !!


Maquina de Cafe
- cafe: Cafetera
- vasosPequenos: Vaso
- vasosMedianos: Vaso
- vasosGrandes: Vaso
- azucar: Azuquero
+getTipoVaso(tipoDeVaso)
+getVasoDeCafe(tipoDeVaso, cantidadDeVasos, cantidadDeAzucar)
Vaso
-cantidadVasos : int
-contenido:int
+setCantidadVasos(param1)
+getCantidadVasos()
+setContenido(param1)
+getContenido()
+hasVasos(cantidadVasos)
+giveVasos(cantidadVasos)
Cafetera
- cantidadCafe : int
+setCantidaDeCafe (param1)
+getCantidadDeCafe ()
+hasCafe(cantidadaCafe)
+giveCafe(cantidadCafe)
Azucarero
- cantidadDeAzucar : int
+setCantidadDeAzucar (param1)
+getCantidadDeAzucar ()
+hasAzucar(cantidadDeAzucar)
+giveAzucar(cantidadDeAzucar)

```java
import static org.junit.Assert.*;

import org.junit.Test;


public class TestVaso {

	@Test
	public void deberiaDevolverVerdaderoSiExistenVasos() {
		Vaso vasosPequenos=new Vaso(2,10);

		boolean resultado=
				vasosPequenos.hasVasos(1);

		assertEquals(true,resultado);
	}

	@Test
	public void deberiaDevolverFalsoSiNoExistenVasos() {
		Vaso vasosPequenos=new Vaso(1,10);

		boolean resultado=
				vasosPequenos.hasVasos(2);

		assertEquals(false,resultado);
	}

	@Test
	public void deberiaRestarCantidadDeVaso() {
		Vaso vasosPequenos=new Vaso(5,10);

		vasosPequenos.giveVasos(1);

		assertEquals(4,vasosPequenos.getCantidadVasos());
	}
```

```java
import org.junit.Test;


public class TestCafetera {

	@Test
	public void deberiaDevolverVerdaderoSiExisteCafe() {
		Cafetera cafetera=new Cafetera(10);

		boolean resultado=
				cafetera.hasCafe(5);

		assertEquals(true,resultado);

	}

	@Test
	public void deberiaDevolverFalsoSiNoExisteCafe() {
		Cafetera cafetera=new Cafetera(10);

		boolean resultado=
				cafetera.hasCafe(11);

		assertEquals(false,resultado);

	}

	@Test
	public void deberiaRestarcafeAlaCafetera() {
		Cafetera cafetera=new Cafetera(10);

		cafetera.giveCafe(7);

		assertEquals(3,cafetera.getCantidadCafe());

	}

}
```

```java
public class TestAzuquero {
	Azucarero azuquero;

	@Before
	public void setUp(){
		azuquero = new Azucarero(10);
	}

	@Test
	public void deberiadevolverVerdaderoSiHaySuficienteAzucarEnElAzuquero() {

		boolean resultado= azuquero.hasAzucar(5);

		assertEquals(true,resultado);

		resultado= azuquero.hasAzucar(10);

		assertEquals(true,resultado);
	}

	@Test
	public void deberiadevolverFalsoPorqueNoHaySuficienteAzucarEnElAzuquero() {

		boolean resultado= azuquero.hasAzucar(15);

		assertEquals(false,resultado);
	}

	@Test
	public void deberiaRestarAzucarAlAzuquero() {

		azuquero.giveAzucar(5);

		assertEquals(5,azuquero.getCantidadAzucar());

		azuquero.giveAzucar(2);

		assertEquals(3,azuquero.getCantidadAzucar());
	}
```

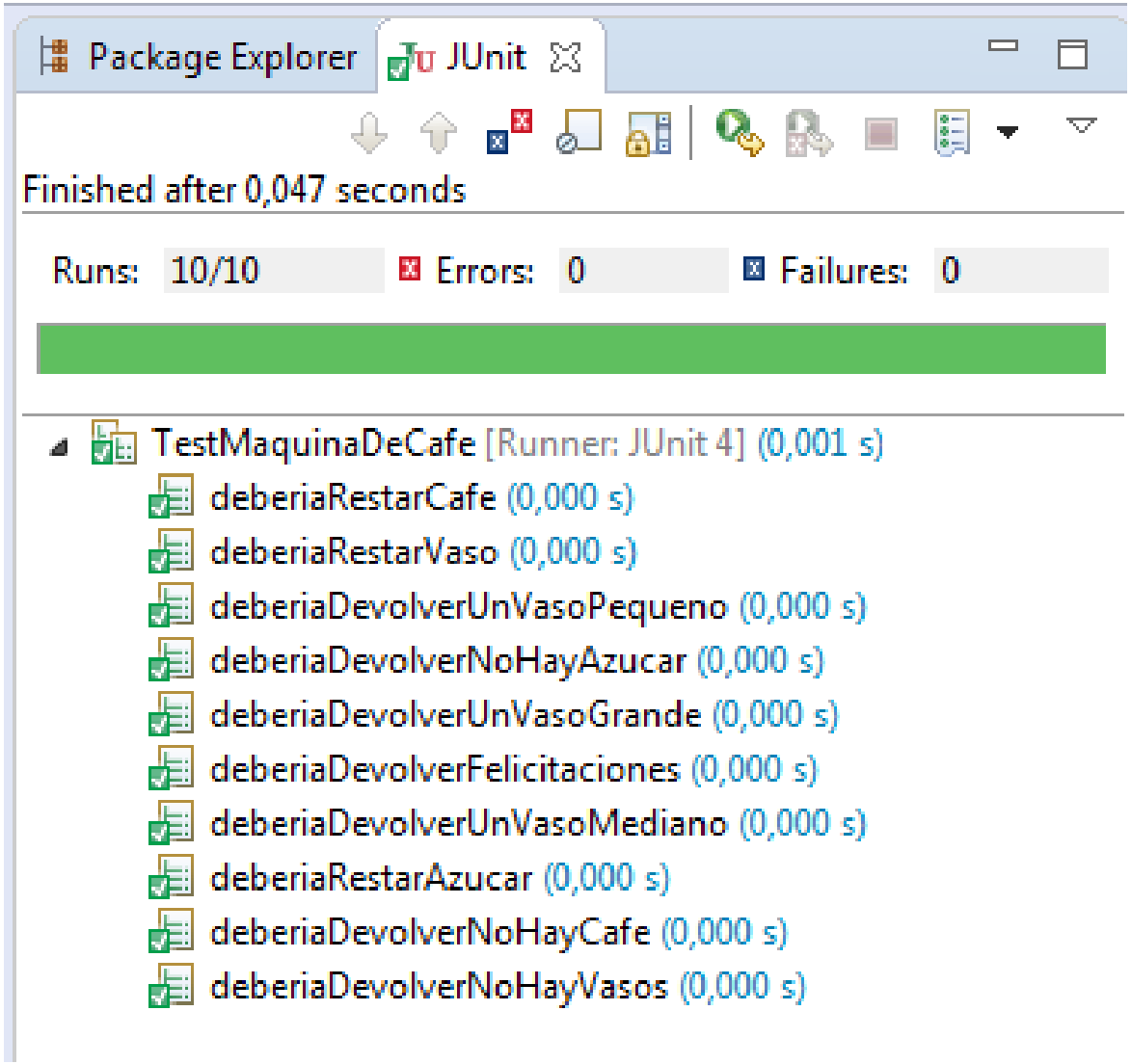
```java
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;


public class TestMaquinaDeCafe {
	Cafetera cafetera;
	Vaso vasosPequeno;
	Vaso vasosMediano;
	Vaso vasosGrande;
	Azucarero azucarero;
	MaquinaDeCafe maquinaDeCafe;

	@Before
	public void setUp(){
		cafetera=new Cafetera(50);
		vasosPequeno=new Vaso(5,10);
		vasosMediano=new Vaso(5,20);
		vasosGrande=new Vaso(5,30);
		azucarero=new Azucarero(20);

		maquinaDeCafe=new MaquinaDeCafe();
		maquinaDeCafe.setCafetera(cafetera);
		maquinaDeCafe.setVasosPequeno(vasosPequeno);
		maquinaDeCafe.setVasosMediano(vasosMediano);
		maquinaDeCafe.setVasosGrande(vasosGrande);
		maquinaDeCafe.setAzucarero(azucarero);
	}

	@Test
	public void deberiaDevolverUnVasoPequeno() {

		Vaso vaso=maquinaDeCafe.getTipoDeVaso("pequeno");

		assertEquals(maquinaDeCafe.vasosPequeno,vaso);

	}
```

```java
@Test
public void deberiaDevolverUnVasoMediano() {
        MaquinaDeCafe maquinaDeCafe=new MaquinaDeCafe();

        Vaso vaso=maquinaDeCafe.getTipoDeVaso("mediano");

        assertEquals(maquinaDeCafe.vasosMediano,vaso);

}

@Test
public void deberiaDevolverUnVasoGrande() {

        Vaso vaso=maquinaDeCafe.getTipoDeVaso("grande");

        assertEquals(maquinaDeCafe.vasosGrande,vaso);

}

@Test
public void deberiaDevolverNoHayVasos() {

        Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

        String resultado=
                        maquinaDeCafe.getVasoDeCafe(vaso,10,2);

        assertEquals("No hay Vasos",resultado);

}
```

```java
@Test
public void deberiaDevolverNoHayCafe() {

        cafetera=new Cafetera(5);
        maquinaDeCafe.setCafetera(cafetera);

        Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

        String resultado=
                        maquinaDeCafe.getVasoDeCafe(vaso,1,2);

        assertEquals("No hay Cafe",resultado);

}

@Test
public void deberiaDevolverNoHayAzucar() {

        azucarero=new Azucarero(2);
        maquinaDeCafe.setAzucarero(azucarero);

        Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

        String resultado=
                        maquinaDeCafe.getVasoDeCafe(vaso,1,3);

        assertEquals("No hay Azucar",resultado);

}
```

```java
@Test
public void deberiaRestarCafe() {


        Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

        maquinaDeCafe.getVasoDeCafe(vaso,1,3);

        int resultado=
                        maquinaDeCafe.getCafetera().getCantidadCafe();

        assertEquals(40,resultado);

}

@Test
public void deberiaRestarVaso() {


        Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

        maquinaDeCafe.getVasoDeCafe(vaso,1,3);

        int resultado=
                        maquinaDeCafe.getVasosPequeno().getCantidadVasos();

        assertEquals(4,resultado);

}
```

```
    @Test
    public void deberiaRestarAzucar() {


            Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

            maquinaDeCafe.getVasoDeCafe(vaso,1,3);

            int resultado=
                            maquinaDeCafe.getAzucarero().getCantidadAzucar();

            assertEquals(17,resultado);

    }


    @Test
    public void deberiaDevolverFelicitaciones() {


            Vaso vaso= maquinaDeCafe.getTipoDeVaso("pequeno");

            String resultado=
                            maquinaDeCafe.getVasoDeCafe(vaso,1,3);

            assertEquals("Felicitaciones",resultado);

    }
}
```


Package Explorer
JUnit
Finished after 0,047 seconds
Runs: 10/10
Errors: 0
Failures: 0
TestMaquinaDeCafe [Runner: JUnit 4] (0,001 s)
deberiaRestarCafe (0,000 s)
deberiaRestarVaso (0,000 s)
deberiaDevolverUnVasoPequeno (0,000 s)
deberiaDevolverNoHayAzucar (0,000 s)
deberiaDevolverUnVasoGrande (0,000 s)
deberiaDevolverFelicitaciones (0,000 s)
deberiaDevolverUnVasoMediano (0,000 s)
deberiaRestarAzucar (0,000 s)
deberiaDevolverNoHayCafe (0,000 s)
deberiaDevolverNoHayVasos (0,000 s)