Introducción

Sobre este documento

En esta sección se listan una serie de pruebas de regresión que se le aplican al sistema “Walkers of the city” para verificar su correcto funcionamiento. Para realizar dichas pruebas se aplico Unit testing a través de la librería JUnit de java. Aunque muchos de los test mostrados no están implementados, se listan para que se implementen cuando sea el momento; ya que se considera que son esenciales para mantener un mínimo de control sobre el funcionamiento del sistema.

Modelo general de testing con JUnit

Template genérico

Para hacer Unit Testing con JUnit desarrollamos el siguiente template que nos sirve como modelo general para desarrollar los casos de test, además de especificar ciertas convenciones de escritura y de programación.

```
import junit.framework.TestCase;


public class test_[NOMBRE_CLASE] extends TestCase {

	[OBJETOS NECESARIOS PARA REALIZAR EL TEST]

	public void setUp() {
		[INSTACIAR LOS OBJETOS NECESARIOS]
		[LLEVAR LAS INSTACIAS AL ESTADO ADECUADO]
	}

	public void test[NOMBRE_METODO]() {
		[PROCESO DE TESTING]
		[USAR ALGUN TIPO DE ASSERT]
	}
}
```

Tipos de assert

- assertEquals(expected, actual)
- assertEquals(message, expected, actual)
- assertEquals(expected, actual, delta) - used on doubles or floats, where delta is the difference in precision
- assertEquals(message, expected, actual, delta) - used on doubles or floats, where delta is the difference in precision
- assertFalse(condition)
- assertFalse(message, condition)
- assertNotNull(object)
- assertNotNull(message, object)
- assertNotSame(expected, actual)
- assertNotSame(message, expected, actual)
- assertNull(object)
- assertNull(message, object)
- assertSame(expected, actual)
- assertSame(message, expected, actual)
- assertTrue(condition)
- assertTrue(message, condition)

Unit Testing

Caso N

Objeto del test: [que se está testeando]
Estado: [implementado / no implementado]
Descripción del test: [como se está testeando (en lo referente a la implementación)]
Condiciones de ejecución: [condiciones previas a la ejecución del test]
Resultado: [éxito / falla]

Unit Testing

A continuación se presentan los diferentes casos de testing realizados:

Caso 1

Objeto del test: Conexión tcp de parte del cliente al servidor Localhost y puerto 5000(estado de conexión de un objeto de la clase Conexión)
Estado: implementado
Descripción del test: se testea la dirección del puerto de origen del socket correspondiente, de ser 0 es porque no se produjo la conexión.
Condiciones de ejecución: Se debe tener un puerto abierto por parte del servidor
Resultado: Falla (no esta habilitado el puerto)

```
import junit.framework.TestCase;


public class test_Conexion extends TestCase {
	Conexion c;

	public void setUp()
	{
		c = new Conexion();

	}

	public void testconectar() {
		//fail("Not yet implemented");
		c.conectar("localhost", 5000);
		assertEquals("no hay conexion", 0, c.getSocket().getPort());
	}
	/*De no producirse la conexion el puerto en el socket queda como 0
	 getPort() Returns:
		the remote port number to which this socket is connected,
	    or 0 if the socket is not connected yet.*/


}
```

Caso 2

Objeto del test: Se comprueba que se cree una nueva partida en espera.
Estado: Implementado.
Descripción del test: Luego de ejecutar el método agregar_partida_en_espera de la clase ControladorMasterServer se verifica que el objeto devuelto por el hashMap con la key correspondiente no sea NULL.
Condiciones de ejecución: Ninguna.
Resultado: Falla por falta de implementación de la clase ControladorMasterServer

```
import junit.framework.TestCase;

public class test_ControladorMasterServer extends TestCase {

	ControladorMasterServer a;

	public void setUp() {
		a = new ControladorMasterServer(7170); //puerto = 7170
		Sopa_letras sopa = new Sopa_letras();
		sopa.asignar_palabra(1);
		ControladorSopaDeLetras cs= new ControladorSopaDeLetras(sopa);
		ControladorMovimientoUsuario cm = new
ControladorMovimientoUsuario(new mapa_ciudad());
		Partida nueva = new Partida(cs,cm,new Jugador(1), 1); //id = 1 → 
partida
		a.agregar_partida_en_espera(nueva);
	}

	public void testAgregar_partida_en_espera() {
		assertNotNull("El hashMap no deberia devolver una referencia nula a
la key correspondiente", a.get_partidas_en_espera().get(1));
	}
}
```

Caso 3

Objeto del test: Se comprueba que se verifique correctamente la palabra ingresada por un jugador.
Estado: No implementado.
Descripción del test: Luego de que el jugador ingrese un par de coordenadas correctas se espera que el metodo comprobarPalabra de la clase ControladorSopaLetras retorne true.
Condiciones de ejecución: Las coordenadas ingresadas son correctas.
Resultado: Falla por falta de implementación de las clases involucradas.

Caso 4

Objeto del test: Se comprueba que un auto luego de moverse en línea recta no se haya salido de los valores permitidos en el mapa
Estado: No Implementado.
Descripción del test: Luego de ejecutar el método move_forward de la clase auto, se verifica que los valores de sus coordenadas hayan aumentado en 1 en la dirección correspondiente.
Condiciones de ejecución: El auto estaba sobre una calle (en donde es correcto su paso).
Resultado: Falla por falta de implementación.

Caso 5

Objeto del test: Conexión tcp de parte del cliente al servidor Localhost y puerto 5000(estado de conexión de un objeto de la clase Conexión)
Estado: implementado
Descripción del test: se testea la dirección del puerto de destino del socket correspondiente, de ser 0 es porque no se produjo la conexión.
Condiciones de ejecución: Se debe tener un puerto abierto por parte del servidor
Resultado: Falla (no esta habilitado el puerto)

```
import junit.framework.TestCase;
public class test_Conexion extends TestCase {
	Conexion c;
	public void setUp()
	{
		c = new Conexion();
	}
	public void testconectar() {
		//fail("Not yet implemented");
		c.conectar("localhost", 5000);
		assertEquals("no hay conexion", 0, c.getSocket().getPort());
	}
/*De no producirse la conexion el puerto en el socket queda como 0
	 getPort() Returns:
		the remote port number to which this socket is connected,
		or 0 if the socket is not connected yet.*/

}
```

Caso 6

Objeto del test: Se comprueba que el jugador y el ControlCambios se encuentran agregados al hashmap correspondiente de MasterServer y se cree correctamente el AnalizadorMaster
Estado: Implementado.
Descripción del test: Se crea una instancia de AnalizadorMaster a traves del metodo ControladorMasterServer::crear_analizador() y se controla que los campos del mismo sean distintos de null e iguales a los correspondientes en el hashmap de ControladorMasterServer
Condiciones de ejecución: Ninguna
Resultado: Falla por falta de implementación de la clase ControladorMasterServer

```
import junit.framework.TestCase;
public class test_ControladorMasterServer extends TestCase {
	ControladorMasterServer a;
	Jugador jugador;
	ControlCambios control;
	ControladorMovimientoJugador cm;
	ControladorSopadeLetras cs;
	string comando;

	public void setUp() {
		a = new ControladorMasterServer(7170); //puerto = 7170
		Sopa_letras sopa = new Sopa_letras();
		sopa.asignar_palabra(1);
		ControladorSopaDeLetras cs= new ControladorSopaDeLetras(sopa);
		ControladorMovimientoUsuario cm = new
ControladorMovimientoUsuario(new mapa_ciudad());
		jugador=new Jugador(cs,cm,1,"192.168.0.2");// partida id =1
		comando = "arriba";
		control = new ControlCambios();
	}
	public void test() {
		AnalizadorMaster am;
		a.controlador_notificadores.put(1,control);
		a.mapeo_ip_jugador.put(jugador.get_ip(),jugador);
		am = a.crear_analizador(jugador, comando, control);

		// comprobación de nulidad de los objetos
		assertNotNull("el campo AnalizadorMaster::jugador no debería ser
	nulo", am.get_jugador());
		assertNotNull("el campo AnalizadorMaster::cambios no debería ser
	nulo", am.get_ControlCambios());
		// comprobación de igualdad entre instancias de objetos
		assertEquals("El jugador que posee el AnalizadorMaster debería ser
el mismo que posee el ControladorMasterServer según el ip recibido",
a.mapeo_ip_jugador.get(jugador.get_ip()), am.get_jugador());
		assertEquals("La instancia de ControlCambios que posee el
AnalizadorMaster debería ser el mismo que posee el ControladorMasterServer según
el ip recibido", a.controladores_notificadores.get(1), am.get_cambios());    }
}
```