```
package Proyecto;

import static org.junit.Assert.*;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

import java.util.Arrays;
import java.util.List;

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

public class TestProyecto {

	private final int idDemanda=7007;
	private final int idOferta=101;
	private Oferta oferta;
	private Demanda demanda;
	Usuario usuario;
	Propuesta propuesta;

	@Before
	public void inicializar (){
		usuario = mock(Usuario.class);
		when(usuario.getNombre()).thenReturn("Paquito Perez");
		demanda = Demanda.crearDemanda(idDemanda, usuario.getNombre());
		oferta = Oferta.crearOferta(idOferta, "Proyecto Software");
	}

	//Comprueba que si la demanda cumple todos los requisitos de la oferta, la puntuacion debe
ser 10
	@Test
	public void PropuestaCoincidencias() throws Exception {
		List<Boolean> lista = Arrays.asList(true, true, true);
		oferta.setLenguajes(lista);
		oferta.setIdiomas(lista);
		demanda.setLenguajes(lista);
		demanda.setIdiomas(lista);
		oferta.setExperiencia(4);
		demanda.setExperiencia(4);
		propuesta = new Propuesta(demanda, oferta);

		assertEquals(10.0, propuesta.getPuntuacion(), 1);
	}

	//Comprueba que si la demanda no cumple ningun requisito de la oferta, la puntuacion debe
ser 0
	@Test
	public void PropuestaSinCoincidencias() throws Exception {
		List<Boolean> lista1 = Arrays.asList(true, true, false);
		List<Boolean> lista2 = Arrays.asList(false, false, false);
		oferta.setLenguajes(lista1);
		oferta.setIdiomas(lista1);
		demanda.setLenguajes(lista2);
		demanda.setIdiomas(lista2);
		oferta.setExperiencia(10);
		demanda.setExperiencia(0);
		propuesta = new Propuesta(demanda, oferta);
		assertEquals(0.0, propuesta.getPuntuacion(), 1);
	}

	//Comprueba que para un caso cualquiera, la propuesta devuelve una puntuacion comprendida
entre 0 y 10
	@Test
	public void PropuestaCorrecta() throws Exception {
		List<Boolean> lista1 = Arrays.asList(true, true, false);
		List<Boolean> lista2 = Arrays.asList(false, true, false);
		oferta.setLenguajes(lista1);
```

```
                oferta.setIdiomas(lista1);
                demanda.setLenguajes(lista2);
                demanda.setIdiomas(lista2);
                oferta.setExperiencia(10);
                demanda.setExperiencia(8);
                propuesta = new Propuesta(demanda, oferta);
                assertTrue(propuesta.getPuntuacion()>=0 && propuesta.getPuntuacion()<=10);
        }

        //Comprueba que para el caso en que la diferencia de a�os de experiencia entre la
requerida por la oferta y la del demandante
        //solo sea de 1 a�o, el algoritmo realizado en propuesta le asignara 2/3 de la
ponderacion asignada para los a�os de experiencia.
        @Test
        public void DiferenciaA�osExperiencia1() throws Exception{
                oferta.setExperiencia(10);
                demanda.setExperiencia(9);
                propuesta = new Propuesta(demanda, oferta);
                assertEquals(propuesta.getPuntuacion(), 6.66, 2);
        }

        //Comprueba que para el caso en que la diferencia de a�os de experiencia entre la
requerida por la oferta y la del demandante
        //sea de 3 o menos, el algoritmo realizado en propuesta le asignara 1/3 de la ponderacion
asignada para los a�os de experiencia.
        @Test
        public void DiferenciaA�osExperienciaMenor3() throws Exception{
                oferta.setExperiencia(10);
                demanda.setExperiencia(7);
                propuesta = new Propuesta(demanda, oferta);
                assertEquals(propuesta.getPuntuacion(), 3.33, 2);
        }


        //Cuando finalice cada test, las listas deben de vaciarse
        @After
        public void finalizar (){
                oferta.getLenguajes().clear();
                oferta.getIdiomas().clear();
                demanda.getIdiomas().clear();
                demanda.getLenguajes().clear();
        }


}
```