```java
package Patron.GeneradorIDs;

/*
 * La clase GeneradorIDs funciona como un patrón Singleton porque
 * sólo puede existir una clase que proporcione las IDs únicas a
 * los usuarios. De existir varias, podrían producirse problemas
 * a la hora de asignar las IDs, por ejemplo colisiones.
 */

public class GeneradorIDs {
	private static GeneradorIDs generador;
	private int id_count;

	private GeneradorIDs() {
		this.id_count = 0;
	}

	/*
	 * getInstancia(): GeneradorIDs
	 *		Devuelve la instancia asociada con la clase GeneradorIDs.
	 *		Si todavía no existe, se crea una nueva.
	 */
	public static GeneradorIDs getInstancia() {
		if(generador == null)
			generador = new GeneradorIDs();
		return generador;
	}

	/*
	 * generateID(Rol): String
	 *		Dado un enum Rol, devuelve un ID único que incluye tanto
	 *		el orden en el que se creó con respecto a otros usuarios
	 *		como el rol que posee.
	 *
	 *		Por ejemplo, si ya se han creado 20 usuarios y se quiere
	 *		crear un supervisor de cuentas (valor 1), devolverá:
	 *		1 - 21
	 */
	public String generateID(Rol rol) {
		String id = new String();
		id += rol.getValue() + "-" + id_count++;
		return id;
	}
}

////////////////////////////////////////////////////////////////////////////////////////////
package Patron.GeneradorIDs;

public enum Rol {
	USUARIO(0),
	SUPERVISOR_CUENTAS(1),
	SUPERVISOR_GENERAL(2);

	private final int valor;

	private Rol(int valor) {
		this.valor = valor;
	}

	public int getValue() {
		return this.valor;
	}
}

////////////////////////////////////////////////////////////////////////////////////////////
```

```
package Patron.GeneradorIDsTest;

import Patron.GeneradorIDs.*;

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


public class GeneradorIDsTest {

	private GeneradorIDs generador;

	@Before
	public void inicializar() {
		generador = GeneradorIDs.getInstancia();
	}

	@Test
	public void generadorIDsSeComportaComoSingleton() {
		GeneradorIDs generador2 = GeneradorIDs.getInstancia();
		assertEquals(generador, generador2);
	}

	@Test
	public void generadorIDsGeneraIDsDeFormaSecuencial() {
		String user, user2, acc_supervisor, gen_supervisor, gen_supervisor_generador2;

		/******/

		// El mismo generador genera IDs de forma secuencial
		user = generador.generateID(Rol.USUARIO);
		user2 = generador.generateID(Rol.USUARIO);
		assertFalse(user.equals(user2));
		acc_supervisor = generador.generateID(Rol.SUPERVISOR_CUENTAS);
		assertFalse(user2.equals(acc_supervisor));
		gen_supervisor = generador.generateID(Rol.SUPERVISOR_GENERAL);
		assertFalse(acc_supervisor.equals(gen_supervisor));

		/******/

		// Otro generador (realmente el mismo) debe seguir esa secuencialidad
		// al tratarse de un singleton.
		GeneradorIDs generador2 = GeneradorIDs.getInstancia();
		gen_supervisor_generador2 = generador2.generateID(Rol.SUPERVISOR_GENERAL);
		assertFalse(gen_supervisor.equals(gen_supervisor_generador2));

	}
}
```