Solucionario Guia de ejercicios

JUnit

- **Descripción Paso 1**: Test para el método eliminarPersona()

```
public class ServicioPersonaTest {
  private static Logger logger =
  Logger.getLogger("cl.desafiolatam.servicios.ServicioPersonaTest");
private final ServicioPersona servicioPersona = new ServicioPersona();

  @Test
  public void testEliminarPersona() {
  logger.info("info eliminar persona");
  Persona pepe = new Persona("1234-1", "pepe");
  String respuestaServicio = servicioPersona.eliminarPersona(pepe);
assertEquals(respuestaServicio, "Eliminada");
 }
}
```

- **Descripción Paso 2:** Test para el método listarPersona()

```
public class ServicioPersonaTest {
  private static Logger logger =
  Logger.getLogger("cl.desafiolatam.servicios.ServicioPersonaTest");
private final ServicioPersona servicioPersona = new ServicioPersona();

  @Test

  public void testListarPersona() {
  logger.info("info listar persona");
  Map<String, String> listaPersonas = servicioPersona.listarPersonas();
assertNotNull(listaPersonas);
 }
}
```

Tests dobles

Paso 1: Para la prueba del método `testEliminarPersona`, se pasan parámetros similares llamando al método `eliminarPersona` del repositorio.

```
package repositorio;
import modelos.Persona;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

  public class RepositorioPersonaTest {
   private RepositorioPersona repositorioPersona =
   mock(RepositorioPersona.class);
  //resto de la clase

  @Test
   public void testEliminarPersona() {
   Persona sam = new Persona("1-4", "Sam");
   when(repositorioPersona.eliminarPersona(sam)).thenReturn("OK");
String eliminarRes = repositorioPersona.eliminarPersona(sam);
   assertEquals("OK", eliminarRes);
   verify(repositorioPersona).eliminarPersona(sam);
  }
}
```

Paso 2: Finalmente, para método de prueba `testListarPersona` se establece un mapa llamado `mockRespuesta`, el cual es un `HashMap<String, String>` que será seteado como el valor que será retornado por parte del repositorio. Con esto tenemos la respuesta esperada y se puede hacer el flujo de llamar al método del repositorio. Se comprueba si el método ocurrió una vez usando `verify`.

```
package repositorio;
import modelos.Persona;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import java.util.HashMap;
import java.util.Map;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

  public class RepositorioPersonaTest {
   private RepositorioPersona repositorioPersona =
   mock(RepositorioPersona.class);
   //resto de la clase

  @Test
  public void testListarPersona() {
  Map<String, String> mockRespuesta = new HashMap<>();
  when(repositorioPersona.listarPersonas()).thenReturn(mockRespuesta);
Map<String,String> listarRes = repositorioPersona.listarPersonas();
assertEquals(mockRespuesta, listarRes);
  verify(repositorioPersona).listarPersonas();
 }
}
```

Test Driven Development

Paso 1: Crear la clase EquipoFutbol.

```
package modelo;

public class EquipoFutbol {
     private int juegosGanados;
     private int juegosPerdidos;
     private int juegosEmpatados;

    public EquipoFutbol(int juegosGanados, int juegosPerdidos, int
juegosEmpatados) {
          this.juegosGanados = juegosGanados;
          this.juegosPerdidos = juegosPerdidos;
          this.juegosEmpatados = juegosEmpatados;
     }
      public int getJuegosGanados() {
      return juegosGanados;
     }
    public int getJuegosPerdidos() {
       return juegosPerdidos;
    }
    public int getJuegosEmpatados() {
       return juegosEmpatados;
    }

}
```

Paso 2: Crear los test en la clase EquipoFutbolTest.

```
public class EquipoFutbolTest {
     private static final int CUATRO_JUEGOS_GANADOS = 4;
     private static final int CINCO_JUEGOS_EMPATADOS = 5;
     private static final int TRES_JUEGOS_PERDIDOS = 3;

     @Test
      public void constructorDebeSetearJuegosGanados() {
        EquipoFutbol team = new EquipoFutbol(CUATRO_JUEGOS_GANADOS,
TRES_JUEGOS_PERDIDOS, CINCO_JUEGOS_EMPATADOS);
         assertEquals(CUATRO_JUEGOS_GANADOS,team.getJuegosGanados());
         assertEquals(TRES_JUEGOS_PERDIDOS,team.getJuegosPerdidos());
         assertEquals(CINCO_JUEGOS_EMPATADOS,team.getJuegosEmpatados());
     }
}
```