Modo estandar e unit test

Prueba para la clase Title

Vamos a crear un test para la clase Title, donde verificamos que los métodos getTitle(), getTitleId() y getPrice() funcionan correctamente.

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class TitleTest {

    private Title title;

    @BeforeEach
    public void setUp() {
        // Inicializamos el objeto Title antes de cada test
        title = new Title("T001", "Java Programming", 29.99);
    }

    @Test
    public void testGetTitleId() {
        // Verificamos que el ID del título sea correcto
        assertEquals("T001", title.getTitleId(), "El ID del título no es el esperado.");
    }

    @Test
    public void testGetTitle() {
        // Verificamos que el nombre del título sea correcto
        assertEquals("Java Programming", title.getTitle(), "El nombre del título no es
el esperado.");
    }

    @Test
    public void testGetPrice() {
```

```
        // Verificamos que el precio del título sea correcto
        assertEquals(29.99, title.getPrice(), 0.01, "El precio del título no es el
esperado.");
    }

    @Test
    public void testTitleNotNull() {
        // Verificamos que el objeto Title no sea nulo
        assertNotNull(title, "El objeto Title es nulo.");
    }
}
```

Explicación del Código:

1. **@BeforeEach**: Este método se ejecuta antes de cada test, y aquí inicializamos un objeto Title para usarlo en todos los tests.
2. **@Test**: Los métodos que llevan esta anotación son los que JUnit ejecutará como pruebas unitarias.
3. **assertEquals**: Compara el valor esperado con el valor real devuelto por el método. Si los valores no coinciden, la prueba falla.
    - En el caso de los números decimales (price), usamos un tercer parámetro (0.01) para indicar la tolerancia, ya que los números de punto flotante pueden tener pequeñas imprecisiones.
4. **assertNotNull**: Asegura que el objeto title no sea nulo.

Pruebas Unitarias para la Clase Sales

Ahora, vamos a probar la clase Sales para asegurarnos de que las ventas se registran correctamente.

```
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
```

```
public class SalesTest {

    private Sales sale;

    @BeforeEach
    public void setUp() {
        // Inicializamos un objeto de ventas
        sale = new Sales("S001", "T001", "Store01", 50.00);
    }

    @Test
    public void testGetSaleId() {
        // Verificamos que el ID de la venta sea correcto
        assertEquals("S001", sale.getSaleId(), "El ID de la venta no es el esperado.");
    }

    @Test
    public void testGetAmount() {
        // Verificamos que el monto de la venta sea correcto
        assertEquals(50.00, sale.getAmount(), 0.01, "El monto de la venta no es el
esperado.");
    }

    @Test
    public void testGetStoreId() {
        // Verificamos que el ID de la tienda sea correcto
        assertEquals("Store01", sale.getStoreId(), "El ID de la tienda no es el
esperado.");
    }
}
```

Pruebas de la Lógica de CRUD en TitleModel

También podemos probar la clase TitleModel, que gestiona el CRUD (crear, leer, actualizar, eliminar) de objetos Title.

```java
import static org.junit.jupiter.api.Assertions.assertTrue;
import static org.junit.jupiter.api.Assertions.assertNull;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class TitleModelTest {

    private TitleModel titleModel;
    private Title title1;

    @BeforeEach
    public void setUp() {
        // Inicializamos el modelo de título y un título de prueba
        titleModel = new TitleModel();
        title1 = new Title("T001", "Java Programming", 29.99);
        titleModel.addTitle(title1);
    }

    @Test
    public void testAddTitle() {
        // Verificamos que el libro se haya agregado correctamente
        Title fetchedTitle = titleModel.getTitleById("T001");
        assertEquals("Java Programming", fetchedTitle.getTitle(), "El título no se ha
añadido correctamente.");
    }

    @Test
    public void testRemoveTitle() {
        // Eliminamos el título y verificamos que ya no exista
        titleModel.removeTitle("T001");
        Title fetchedTitle = titleModel.getTitleById("T001");
        assertNull(fetchedTitle, "El título no se ha eliminado correctamente.");
    }
}
```

Explicación:

- **testAddTitle**: Verificamos que al agregar un libro a la lista, podemos recuperarlo por su ID.
- **testRemoveTitle**: Verificamos que al eliminar un libro, ya no sea posible recuperarlo.

Pruebas para la Clase Job y Employee

Puedes realizar pruebas similares para las clases Job y Employee siguiendo la misma estructura.

Ejecutando las Pruebas

Para ejecutar las pruebas, puedes usar tu IDE (como IntelliJ IDEA, Eclipse) o desde la línea de comandos con Maven o Gradle.

- **Con Maven:**

```
mvn test
```

- **Con Gradle**:

```
gradle test
```