



# Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Algoritmos y Estructuras de Datos I (CE 1103)

## Proyecto II: MusicBox Test Plan

### **Profesor:**

Leonardo Andres Araya Martinez

### **Estudiantes:**

Steven Aguilar Alavarez  
Alejandro Arias Alfaro

Verano 2024-2025

# Índice

<b>1. Descripción General</b>	<b>2</b>
<b>2. Pruebas Unitarias del Proyecto</b>	<b>2</b>

# 1. Descripción General

Sistema de reproducción de partituras musicales que implementa una lista doblemente enlazada para almacenar y reproducir notas musicales con duraciones configurables. Desarrollado para el curso de Algoritmos y Estructuras de Datos I.

## 2. Pruebas Unitarias del Proyecto

### Unit Test 1: AddNotes

Verificar que al agregar una nota musical a una lista vacía, el contador de elementos se incremente correctamente, validando así la funcionalidad básica de inserción en la estructura de datos.

#### Implementación

```
[Test]
public void AddNotes()
{
    var playlist = new DoublyLinkedList();
    var note = new Note("Do", "negra");

    playlist.addNote(note);

    Assert.AreEqual(1, playlist.GetCount());
}
```

**Entrada Ingresada:** La entrada que se le ingresa a la prueba es una nota musical.

**Resultado Esperado:** La prueba confirma que el contador de la lista se incrementa a 1, validando que la nota fue agregada exitosamente a la estructura.

Prueba ▲	Duración	Rasgos	Mensaje de error
▲ ✓ Music-Box.Tests (5)	1,3 s		
▲ ✓ Music_Box.Tests (5)	1,3 s		
▲ ✓ StevenUnitTests (5)	1,3 s		
✓ AddNotes	11 ms		
✓ FalseTempo	2 ms		
✓ GetCorrectFrequency	< 1 ms		
✓ GetCorrectNext	2 ms		
✓ PlaySingleNote	1,2 s		

Figura 1: Resultados de addNotes.

## Unit Test 2: FalseTempo

Verificar que al intentar establecer un tempo inválido, el resultado sea falso, validando la protección contra valores no permitidos.

### Implementación

```
[Test]
public void FalseTempo()
{
    var player = new MusicPlayer(new DoublyLinkedList());

    bool result = player.setTempo(10.0f);

    Assert.IsFalse(result);
}
```

**Entrada Ingresada:** La entrada para esta prueba es un valor de tempo inválido, específicamente 10.0f.

**Resultado Esperado:** La prueba confirma que el método devuelve falso, indicando que el tempo no se puede establecer.

Prueba ▲	Duración	Rasgos	Mensaje de error
▲ ✓ Music-Box.Tests (5)	1,3 s		
▲ ✓ Music_Box.Tests (5)	1,3 s		
▲ ✓ StevenUnitTests (5)	1,3 s		
✓ AddNotes	11 ms		
✓ FalseTempo	2 ms		
✓ GetCorrectFrequency	< 1 ms		
✓ GetCorrectNext	2 ms		
✓ PlaySingleNote	1,2 s		

Figura 2: Resultados de FalseTempo.

## Unit Test 3: GetCorrectNext

Verificar que al tener varias notas en la lista, el método devuelve correctamente la siguiente nota.

### Implementación

```
[Test]
public void GetCorrectNext()
{
    var playlist = new DoublyLinkedList();
    var note1 = new Note("Do", "negra");
    var note2 = new Note("Re", "corchea");
    playlist.AddNote(note1);
    playlist.AddNote(note2);

    var firstNote = playlist.GetFirst();
    var nextNote = playlist.GetNext();

    Assert.IsNotNull(nextNote);
}
```

**Entrada Ingresada:** Dos notas musicales con sus duraciones respectivamente.

**Resultado Esperado:** La prueba confirma que el método devuelve una nota válida como siguiente.

Prueba ▲	Duración	Rasgos	Mensaje de error
▲ ✓ Music-Box.Tests (5)	1,3 s		
▲ ✓ Music_Box.Tests (5)	1,3 s		
▲ ✓ StevenUnitTests (5)	1,3 s		
✓ AddNotes	11 ms		
✓ FalseTempo	2 ms		
✓ GetCorrectFrequency	< 1 ms		
✓ GetCorrectNext	2 ms		
✓ PlaySingleNote	1,2 s		

Figura 3: Resultados de GetCorrectNext.

## Unit Test 4: GetCorrectFrequency

Verificar que la frecuencia de la nota específica sea la correcta (261.63 Hz).

### Implementación

```
[Test]
public void GetCorrectFrequency()
{
    var note = new Note("Do", "negra");

    double frequency = note.Frequency;

    Assert.AreEqual(261.63, frequency);
}
```

**Entrada Ingresada:** La entrada para esta prueba es una nota musical.

**Resultado Esperado:** La prueba confirma que la frecuencia de la nota probada es 261.63 Hz.

Prueba ▲	Duración	Rasgos	Mensaje de error
▲ ✓ Music-Box.Tests (5)	1,3 s		
▲ ✓ Music_Box.Tests (5)	1,3 s		
▲ ✓ StevenUnitTests (5)	1,3 s		
✓ AddNotes	11 ms		
✓ FalseTempo	2 ms		
✓ GetCorrectFrequency	< 1 ms		
✓ GetCorrectNext	2 ms		
✓ PlaySingleNote	1,2 s		

Figura 4: Resultados de GetCorrectFrequency.

## Unit Test 5: PlaySingleNote

Verificar que al intentar reproducir una sola nota, no se produzca ninguna excepción.

### Implementación

```
[Test]
public void PlaySingleNote()
{
    var playlist = new DoublyLinkedList();
    var player = new MusicPlayer(playlist);
    var note = new Note("Do", "negra");

    Assert.DoesNotThrow(() => player.PlaySingleNote(note));
}
```

**Entrada Ingresada:** La entrada para esta prueba es una nota musical.

**Resultado Esperado:** La prueba confirma que la reproducción de una sola nota no lanza excepciones.

Prueba ▲	Duración	Rasgos	Mensaje de error
▲ ✓ Music-Box.Tests (5)	1,3 s		
▲ ✓ Music_Box.Tests (5)	1,3 s		
▲ ✓ StevenUnitTests (5)	1,3 s		
✓ AddNotes	11 ms		
✓ FalseTempo	2 ms		
✓ GetCorrectFrequency	< 1 ms		
✓ GetCorrectNext	2 ms		
✓ PlaySingleNote	1,2 s		

Figura 5: Resultados de PlaySingleNote.

## Unit Test 6: ParseNotesTest

Verificar que la entrada se divida correctamente y se inserte en la lista enlazada.

### Implementación

```
[Test]
public void ParseNotesTest()
{
    string input = "(Do, negra), (Re, blanca), (Mi, corchea)";

    // Act
    List<Note> notes = Parser.ParseNotes(input);

    Assert.AreEqual(3, notes.Count);
    Assert.AreEqual("Do", notes[0].NoteName);
    Assert.AreEqual("negra", notes[0].Duration);
    Assert.AreEqual("Re", notes[1].NoteName);
    Assert.AreEqual("blanca", notes[1].Duration);
    Assert.AreEqual("Mi", notes[2].NoteName);
    Assert.AreEqual("corchea", notes[2].Duration);
}
```

**Entrada Ingresada:** La entrada para esta prueba es una partitura de 3 notas musicales.

**Resultado Esperado:** La prueba confirma que las notas fueron divididas en sus respectivos nodos de la lista enlazada.

Prueba	Duración	Rasgos	Mensaje de error
Music-Box.Tests (10)	9,2 s		
Music_Box.Tests (10)	9,2 s		
StevenUnitTests (10)	9,2 s		
AddNotes	12 ms		
FalseTempo	2 ms		
GetCorrectFrequency	< 1 ms		
GetCorrectNext	2 ms		
ParseNotes_Test	< 1 ms		
PlayBackward_Test	4,1 s		
PlayForward_ValidPlaylist_Repla...	4 s		
PlaySingleNote	1,2 s		
PrintList_Test	1 ms		
SetTempo_Test	< 1 ms		

Figura 6: Resultados de ParseNotesTest.



## Unit Test 7: PlayForwardTest

Verificar se reproduzca el playlist hacia adelante sin ninguna excepcion.

### Implementación

```
[Test]
public void PlayForwardTest()
{
    var playlist = new DoublyLinkedList();
    playlist.addNote(new Note("Do", "negra"));
    playlist.addNote(new Note("Re", "blanca"));
    playlist.addNote(new Note("Mi", "corchea"));
    var player = new MusicPlayer(playlist);

    Assert.DoesNotThrow(() => player.PlayForward());
}
```

**Entrada Ingresada:** La entrada para esta prueba es la lista enlazada con las notas que se van a reproducir.

**Resultado Esperado:** La prueba confirma que las notas fueron reproducidas sin excepciones.

Prueba	Duración	Rasgos	Mensaje de error
Music-Box.Tests (10)	9,2 s		
Music_Box.Tests (10)	9,2 s		
StevenUnitTests (10)	9,2 s		
AddNotes	12 ms		
FalseTempo	2 ms		
GetCorrectFrequency	< 1 ms		
GetCorrectNext	2 ms		
ParseNotes_Test	< 1 ms		
PlayBackward_Test	4,1 s		
PlayForward_ValidPlaylist_Repla...	4 s		
PlaySingleNote	1,2 s		
PrintList_Test	1 ms		
SetTempo_Test	< 1 ms		

Figura 7: Resultados de PlayForwardTest.

## Unit Test 8: PlayBackwardTest

Verificar se reproduzca el playlist hacia atras sin ninguna excepcion.

### Implementación

```
[Test]
public void PlayBackwardTest()
{
    var playlist = new DoublyLinkedList();
    playlist.addNote(new Note("Do", "negra"));
    playlist.addNote(new Note("Re", "blanca"));
    playlist.addNote(new Note("Mi", "corchea"));
    var player = new MusicPlayer(playlist);

    Assert.DoesNotThrow(() => player.PlayBackward());
}
```

**Entrada Ingresada:** La entrada para esta prueba es la lista enlazada con las notas que se van a reproducir en el orden original.

**Resultado Esperado:** La prueba confirma que las notas fueron reproducidas sin excepciones.

Prueba	Duración	Rasgos	Mensaje de error
Music-Box.Tests (10)	9,2 s		
Music_Box.Tests (10)	9,2 s		
StevenUnitTests (10)	9,2 s		
AddNotes	12 ms		
FalseTempo	2 ms		
GetCorrectFrequency	< 1 ms		
GetCorrectNext	2 ms		
ParseNotes_Test	< 1 ms		
PlayBackward_Test	4,1 s		
PlayForward_ValidPlaylist_Repla...	4 s		
PlaySingleNote	1,2 s		
PrintList_Test	1 ms		
SetTempo_Test	< 1 ms		

Figura 8: Resultados de PlayBackwardTest.

## Unit Test 9: SetTempoTest

Verificar se pueda modificar el valor predeterminado del tiempo.

### Implementación

```
[Test]
public void SetTempoTest()
{
    var player = new MusicPlayer(new DoublyLinkedList());

    Assert.IsTrue(player.setTempo(1.0f)); // Dentro del rango
    v lido
    Assert.IsFalse(player.setTempo(0.05f)); // Fuera del rango (muy
    bajo)
    Assert.IsFalse(player.setTempo(10.0f)); // Fuera del rango (muy
    alto)
}
```

**Entrada Ingresada:** La entrada para esta prueba es tres diferentes tiempos, uno valido y dos fuera de rango.

**Resultado Esperado:** La prueba confirma que el metodo valida las entradas y cambia el valor original.

Prueba	Duración	Rasgos	Mensaje de error
Music-Box.Tests (10)	9,2 s		
Music_Box.Tests (10)	9,2 s		
StevenUnitTests (10)	9,2 s		
AddNotes	12 ms		
FalseTempo	2 ms		
GetCorrectFrequency	< 1 ms		
GetCorrectNext	2 ms		
ParseNotes_Test	< 1 ms		
PlayBackward_Test	4,1 s		
PlayForward_ValidPlaylist_Repla...	4 s		
PlaySingleNote	1,2 s		
PrintList_Test	1 ms		
SetTempo_Test	< 1 ms		

Figura 9: Resultados de SetTempoTest.

## Unit Test 10: PrintListTest

Verificar que se imprima correctamente la lista.

### Implementación

```
[Test]
public void PrintListTest()
{
    var playlist = new DoublyLinkedList();
    playlist.addNote(new Note("Do", "negra"));
    playlist.addNote(new Note("Re", "blanca"));
    playlist.addNote(new Note("Mi", "corchea"));

    using (var sw = new StringWriter())
    {
        Console.SetOut(sw);

        playlist.PrintList();

        var output = sw.ToString().Trim();
        StringAssert.Contains("Nota: Do, Duraci n: negra", output)
;
        StringAssert.Contains("Nota: Re, Duraci n: blanca", output
);
        StringAssert.Contains("Nota: Mi, Duraci n: corchea",
output);
    }
}
```

**Entrada Ingresada:** La entrada para esta prueba es una lista con tres elementos.

**Resultado Esperado:** La prueba confirma que el metodo imprime la lista con el formato esperado.

Prueba ^	Duración	Rasgos	Mensaje de error
Music-Box.Tests (10)	9,2 s		
Music_Box.Tests (10)	9,2 s		
StevenUnitTests (10)	9,2 s		
AddNotes	12 ms		
FalseTempo	2 ms		
GetCorrectFrequency	< 1 ms		
GetCorrectNext	2 ms		
ParseNotes_Test	< 1 ms		
PlayBackward_Test	4,1 s		
PlayForward_ValidPlaylist_Repla...	4 s		
PlaySingleNote	1,2 s		
PrintList_Test	1 ms		
SetTempo_Test	< 1 ms		

Figura 10: Resultados de PrintListTest.