Bucle Canciones Java

A continuación, explicamos la implementación y creación de un código simple en JAVA, que permite utilizar las listas circulares en un programa que simula la reproducción en bucle de 5 canciones en una playlist (lista circular). Se utilizan dos clases: **ListaCircular** y **Nodo**

Clase `Nodo`

```
public class Nodo {
    String cancion;
    Nodo siguiente;

    Nodo(String cancion) {
        this.cancion = cancion;
        this.siguiente = null;
    }
}
```

Explicación

1. **Definición de la Clase**:

   ```
   public class Nodo {
   ```

   - `public class Nodo`: Declara una clase pública llamada `Nodo`. La palabra clave `public` indica que esta clase puede ser accedida desde otras clases.

2. **Atributos de la Clase**:

   ```
   String cancion;
   Nodo siguiente;
   ```

- `String cancion`: Declara una variable de instancia `cancion` de tipo `String` que almacenará el nombre de la canción.
- `Nodo siguiente`: Declara una variable de instancia `siguiente` de tipo `Nodo` que apuntará al siguiente nodo en la lista circular.

3. **Constructor de la Clase**:

```
Nodo(String cancion) {
    this.cancion = cancion;
    this.siguiente = null;
}
```

- `Nodo(String cancion)`: Define un constructor que toma un parámetro `cancion` de tipo `String`.
- `this.cancion = cancion`: Asigna el valor del parámetro `cancion` a la variable de instancia `cancion` del nodo actual. La palabra clave `this` se utiliza para referirse a la instancia actual de la clase.
- `this.siguiente = null`: Inicializa la variable `siguiente` a `null`, indicando que, por defecto, este nodo no apunta a ningún otro nodo.

La clase `Nodo` es fundamental para la estructura de la lista circular. Cada instancia de `Nodo` representa un elemento de la lista, almacenando una canción y una referencia al siguiente nodo. Esto permite que los nodos se conecten entre sí, formando una estructura circular.

Clase `ListaCircular`

```
public class ListaCircular {
    private Nodo inicio;
    private Nodo ultimo;

    public ListaCircular(){
        this.inicio = null;
        this.ultimo = null;
    }
```

```
    public void agregarCancion(String cancion){
        Nodo nuevoNodo = new Nodo(cancion);
        if(this.inicio == null){
            this.inicio = nuevoNodo;
            this.ultimo = nuevoNodo;
            ultimo.siguiente = inicio;
        } else {
            ultimo.siguiente = nuevoNodo;
            ultimo = nuevoNodo;
            ultimo.siguiente = inicio;
        }
    }

    public void reproducir(int veces){
        if (inicio != null) {
            Nodo actual = inicio;
            for (int i = 0; i < veces; i++) {
                do {
                    System.out.println("Reproduciendo: " +
actual.cancion);
                    actual = actual.siguiente;
                } while (actual != inicio);
            }
        }
    }
}
```

Explicación

1. **Definición de la Clase**:

   ```
   public class ListaCircular {
   ```

   - `public class ListaCircular`: Declara una clase pública llamada `ListaCircular`.

2. **Atributos de la Clase**:

```
private Nodo inicio;
private Nodo ultimo;
```

- `private Nodo inicio`: Declara una variable de instancia `inicio` de tipo `Nodo` que apuntará al primer nodo de la lista.
- `private Nodo ultimo`: Declara una variable de instancia `ultimo` de tipo `Nodo` que apuntará al último nodo de la lista.

3. **Constructor de la Clase**:

```
public ListaCircular(){
    this.inicio = null;
    this.ultimo = null;
}
```

- `public ListaCircular()`: Define un constructor que inicializa `inicio` y `ultimo` a `null`, indicando que la lista está vacía al principio.

4. **Método `agregarCancion`**:

```
public void agregarCancion(String cancion){
    Nodo nuevoNodo = new Nodo(cancion);
    if(this.inicio == null){
        this.inicio = nuevoNodo;
        this.ultimo = nuevoNodo;
        ultimo.siguiente = inicio;
    } else {
        ultimo.siguiente = nuevoNodo;
        ultimo = nuevoNodo;
        ultimo.siguiente = inicio;
    }
}
```

- `public void agregarCancion(String cancion)`: Define un método público que agrega una nueva canción a la lista.
- `Nodo nuevoNodo = new Nodo(cancion)`: Crea un nuevo nodo con la canción proporcionada.

- `if(this.inicio == null)`: Verifica si la lista está vacía.
  - Si está vacía, el nuevo nodo se convierte en `inicio` y `ultimo`, y se cierra el ciclo apuntando `ultimo.siguiente` a `inicio`.
- `else`: Si la lista no está vacía, el nuevo nodo se agrega al final y se actualiza `ultimo` para cerrar el ciclo nuevamente.

5. **Método `reproducir`**:

```
public void reproducir(int veces){
    if (inicio != null) {
        Nodo actual = inicio;
        for (int i = 0; i < veces; i++) {
            do {
                System.out.println("Reproduciendo: " + a
ctual.cancion);
                actual = actual.siguiente;
            } while (actual != inicio);
        }
    }
}
```

- `public void reproducir(int veces)`: Define un método público que reproduce las canciones en la lista un número específico de veces.
- `if (inicio != null)`: Verifica si la lista no está vacía.
- `Nodo actual = inicio`: Inicializa un nodo `actual` que comienza en el `inicio` de la lista.
- `for (int i = 0; i < veces; i++)`: Un bucle que se ejecuta el número de veces especificado.
  - `do { ... } while (actual != inicio)`: Un bucle `do-while` que recorre la lista circular, imprimiendo cada canción y avanzando al siguiente nodo hasta que vuelve al `inicio`.

La clase `ListaCircular` maneja la lógica para agregar nodos y reproducir las canciones en un ciclo continuo. Los métodos `agregarCancion` y `reproducir` permiten gestionar y recorrer la lista circular, respectivamente.

Clase `Main`

```
public class Main {
    public static void main(String[] args) {
        ListaCircular listaCanciones = new ListaCircular();

        // Agregamos las canciones
        listaCanciones.agregarCancion("Nice Guy - BOYNEXTDO
OR");
        listaCanciones.agregarCancion("0X1=LOVESONG (I Know
I Love You) feat. Seori - TOMORROW X TOGETHER");
        listaCanciones.agregarCancion("Style (Taylor´s Vers
ion) - Taylor Swift");
        listaCanciones.agregarCancion("Rock with you - SEVE
NTEEN");
        listaCanciones.agregarCancion("Espresso - Sabrina C
arpenter");

        // Simulamos la reproducción en bucle
        System.out.println("Iniciando reproducción:");
        listaCanciones.reproducir(5);
    }
}
```

Explicación

1. **Definición de la Clase**:

   ```
   public class Main {
   ```

   - `public class Main`: Declara una clase pública llamada `Main`.

2. **Método `main`**:

   ```
   public static void main(String[] args) {
   ```

   - `public static void main(String[] args)`: Define el método `main`, que es el punto de entrada de cualquier aplicación Java. La palabra clave `static`

indica que este método puede ser llamado sin crear una instancia de la clase `Main`.

3. **Creación de la Lista Circular**:

```
ListaCircular listaCanciones = new ListaCircular();
```

- `ListaCircular listaCanciones = new ListaCircular()`: Crea una nueva instancia de `ListaCircular` llamada `listaCanciones`.

4. **Agregar Canciones a la Lista**:

```
listaCanciones.agregarCancion("Nice Guy - BOYNEXTDOOR");
listaCanciones.agregarCancion("0X1=LOVESONG (I Know I Lo
ve You) feat. Seori - TOMORROW X TOGETHER");
listaCanciones.agregarCancion("Style (Taylor´s Version)
- Taylor Swift");
listaCanciones.agregarCancion("Rock with you - SEVENTEE
N");
listaCanciones.agregarCancion("Espresso - Sabrina Carpen
ter");
```

- `listaCanciones.agregarCancion(...)`: Llama al método `agregarCancion` de `ListaCircular` para agregar varias canciones a la lista. Cada llamada crea un nuevo nodo en la lista circular con la canción especificada.

5. **Simular la Reproducción en Bucle**:

```
System.out.println("Iniciando reproducción:");
listaCanciones.reproducir(5);
```

- `System.out.println("Iniciando reproducción:")`: Imprime un mensaje en la consola indicando que la reproducción está comenzando.
- `listaCanciones.reproducir(5)`: Llama al método `reproducir` de `ListaCircular` para reproducir las canciones en la lista 5 veces. Este método recorre la lista circular y imprime cada canción en un bucle.

El método `main` crea una instancia de `ListaCircular`, agrega varias canciones a la lista y luego simula la reproducción de estas canciones en un bucle continuo. Este enfoque permite ver cómo se comporta la lista circular en una situación práctica, como la reproducción de música en un servicio de streaming.
