Opris Liviu Vlad

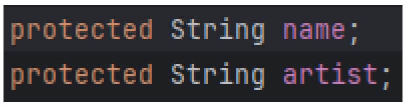
Documentație Proiect
Laborator - MIP

Pentru acest proiect am dezvoltat o mica aplicație care se ocupă cu gestionarea albumelor muzicale. Aceasta aplicatie beneficiază de o interfață ușor de utilizat, realizata in consola. Interfata utilizatorului prezinta un meniu alcatuit dintr-o lista de comenzi pe care user-ul le poate apela pentru a folosi aplicația după bunul sau plac.

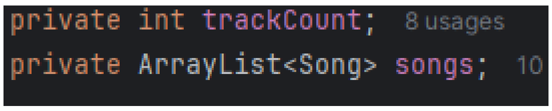
Aceasta aplicatie a fost efectuata în limbajul JAVA folosind cunoștințele deprinse de-a lungul laboratoarelor de MIP. În continuare voi explica utilitatea laboratoarelor pentru realizarea acestui proiect.

<u>Laboratorul 1:</u>

- **tipuri de valori:**
  ex:

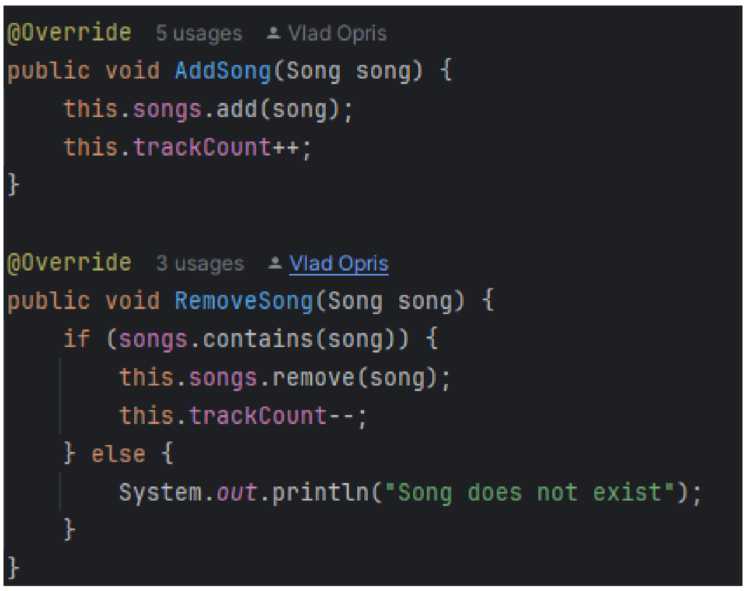


```
protected String name;
protected String artist;
```



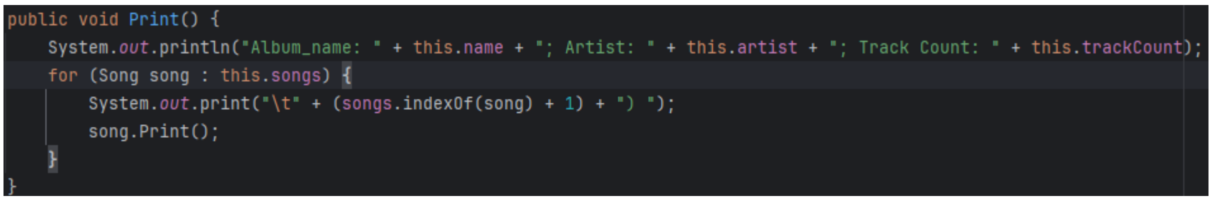
```
private int trackCount;  8 usages
private ArrayList<Song> songs;  10
```

- **functii:**
  ex:

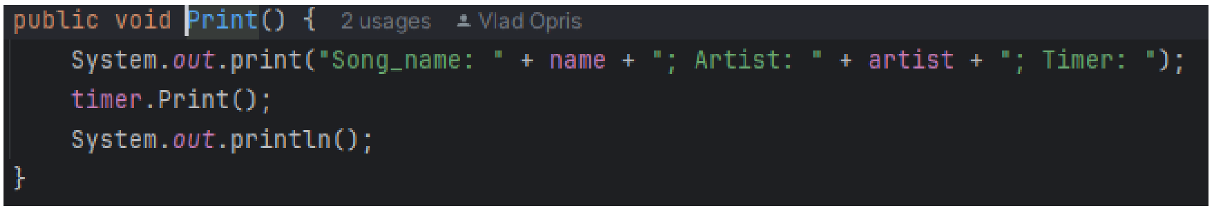


```
@Override  5 usages  Vlad Opris
public void AddSong(Song song) {
    this.songs.add(song);
    this.trackCount++;
}

@Override  3 usages  Vlad Opris
public void RemoveSong(Song song) {
    if (songs.contains(song)) {
        this.songs.remove(song);
        this.trackCount--;
    } else {
        System.out.println("Song does not exist");
    }
}
```

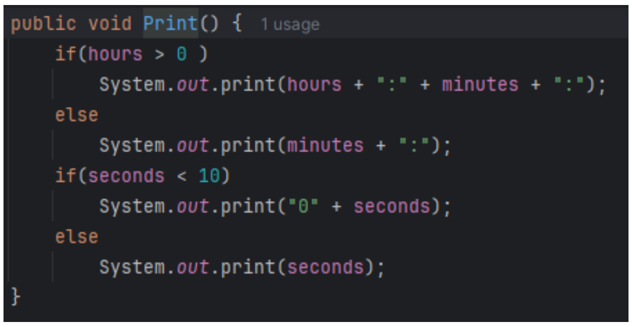
- **output-uri:**
  ex:



```
public void Print() {
    System.out.println("Album_name: " + this.name + "; Artist: " + this.artist + "; Track Count: " + this.trackCount);
    for (Song song : this.songs) {
        System.out.print("\t" + (songs.indexOf(song) + 1) + ") ");
        song.Print();
    }
}
```



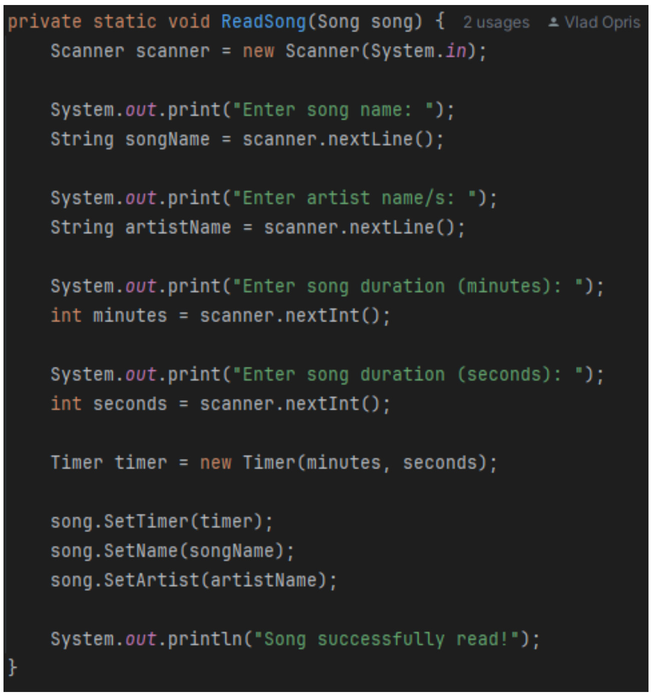
```
public void Print() {  2 usages  Vlad Opris
    System.out.print("Song_name: " + name + "; Artist: " + artist + "; Timer: ");
    timer.Print();
    System.out.println();
}
```



```
public void Print() {  1 usage
    if(hours > 0 )
        System.out.print(hours + ":" + minutes + ":");
    else
        System.out.print(minutes + ":");
    if(seconds < 10)
        System.out.print("0" + seconds);
    else
        System.out.print(seconds);
}
```

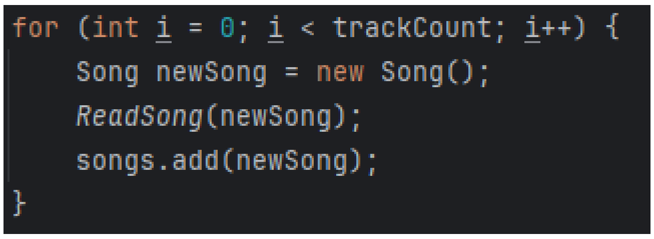
<u>Laboratorul 2:</u>

- **input:**
  ex:

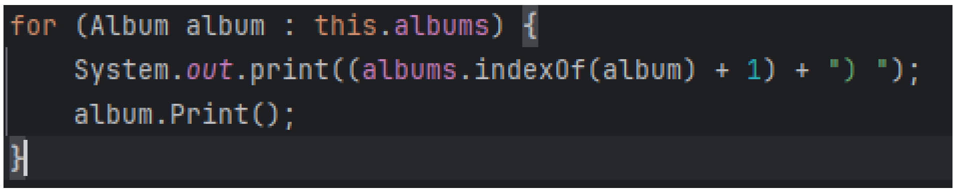


```
private static void ReadSong(Song song) {  2 usages  Vlad Opris
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter song name: ");
    String songName = scanner.nextLine();

    System.out.print("Enter artist name/s: ");
    String artistName = scanner.nextLine();

    System.out.print("Enter song duration (minutes): ");
    int minutes = scanner.nextInt();

    System.out.print("Enter song duration (seconds): ");
    int seconds = scanner.nextInt();

    Timer timer = new Timer(minutes, seconds);

    song.SetTimer(timer);
    song.SetName(songName);
    song.SetArtist(artistName);

    System.out.println("Song successfully read!");
}
```

- **for:**
  ex:

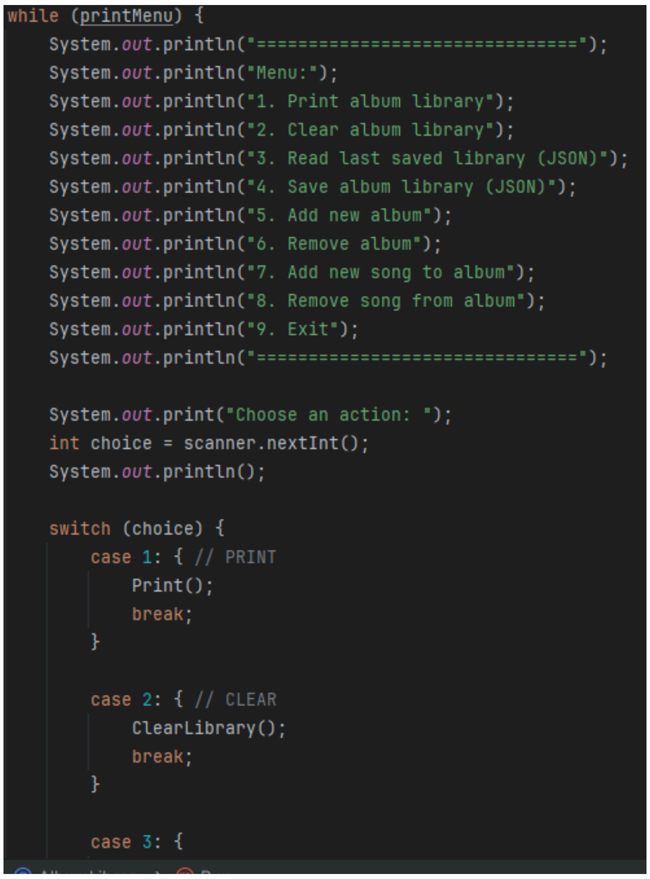


```java
for (int i = 0; i < trackCount; i++) {
    Song newSong = new Song();
    ReadSong(newSong);
    songs.add(newSong);
}
```



```java
for (Album album : this.albums) {
    System.out.print((albums.indexOf(album) + 1) + ") ");
    album.Print();
}
```

- **while:**
  ex:

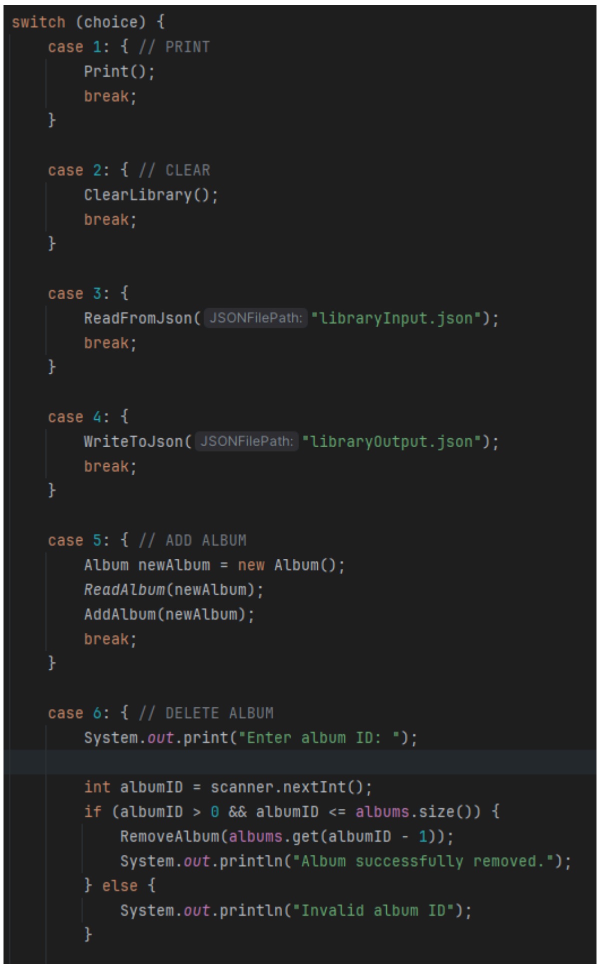


```java
while (printMenu) {
    System.out.println("==============================");
    System.out.println("Menu:");
    System.out.println("1. Print album library");
    System.out.println("2. Clear album library");
    System.out.println("3. Read last saved library (JSON)");
    System.out.println("4. Save album library (JSON)");
    System.out.println("5. Add new album");
    System.out.println("6. Remove album");
    System.out.println("7. Add new song to album");
    System.out.println("8. Remove song from album");
    System.out.println("9. Exit");
    System.out.println("==============================");

    System.out.print("Choose an action: ");
    int choice = scanner.nextInt();
    System.out.println();

    switch (choice) {
        case 1: { // PRINT
            Print();
            break;
        }

        case 2: { // CLEAR
            ClearLibrary();
            break;
        }

        case 3: {
```

Am folosit un while loop-ul de afisare al meniului. Se iese din structura “while” cand “printMenu” devine “false” in “switch”.

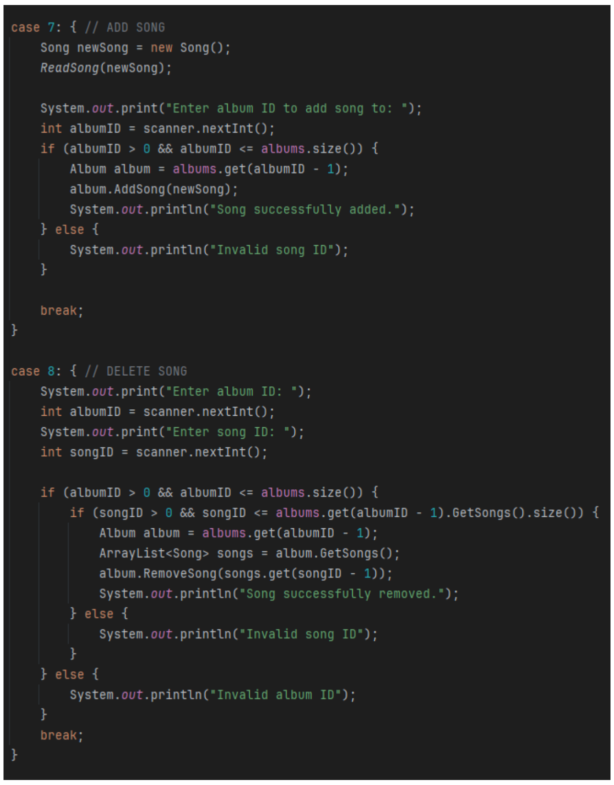
Opris Liviu Vlad

- **switch**

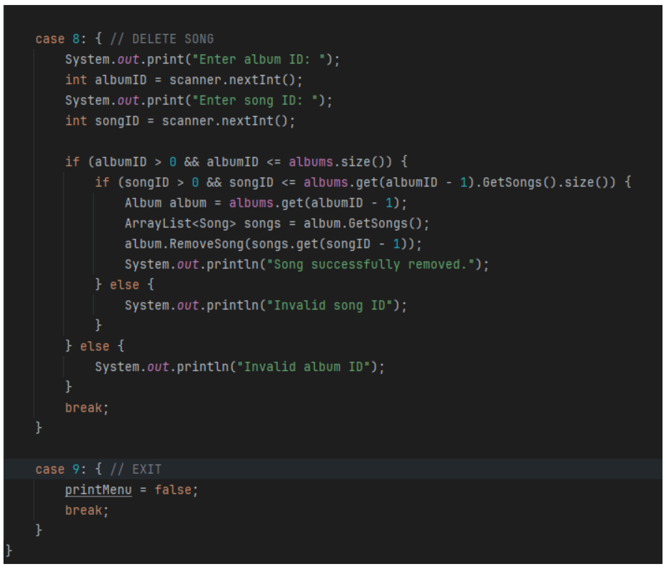


```java
switch (choice) {
    case 1: { // PRINT
        Print();
        break;
    }

    case 2: { // CLEAR
        ClearLibrary();
        break;
    }

    case 3: {
        ReadFromJson( JSONFilePath: "libraryInput.json");
        break;
    }

    case 4: {
        WriteToJson( JSONFilePath: "libraryOutput.json");
        break;
    }

    case 5: { // ADD ALBUM
        Album newAlbum = new Album();
        ReadAlbum(newAlbum);
        AddAlbum(newAlbum);
        break;
    }

    case 6: { // DELETE ALBUM
        System.out.print("Enter album ID: ");

        int albumID = scanner.nextInt();
        if (albumID > 0 && albumID <= albums.size()) {
            RemoveAlbum(albums.get(albumID - 1));
            System.out.println("Album successfully removed.");
        } else {
            System.out.println("Invalid album ID");
        }
```

Opris Liviu Vlad



```java
case 7: { // ADD SONG
    Song newSong = new Song();
    ReadSong(newSong);

    System.out.print("Enter album ID to add song to: ");
    int albumID = scanner.nextInt();
    if (albumID > 0 && albumID <= albums.size()) {
        Album album = albums.get(albumID - 1);
        album.AddSong(newSong);
        System.out.println("Song successfully added.");
    } else {
        System.out.println("Invalid song ID");
    }

    break;
}

case 8: { // DELETE SONG
    System.out.print("Enter album ID: ");
    int albumID = scanner.nextInt();
    System.out.print("Enter song ID: ");
    int songID = scanner.nextInt();

    if (albumID > 0 && albumID <= albums.size()) {
        if (songID > 0 && songID <= albums.get(albumID - 1).GetSongs().size()) {
            Album album = albums.get(albumID - 1);
            ArrayList<Song> songs = album.GetSongs();
            album.RemoveSong(songs.get(songID - 1));
            System.out.println("Song successfully removed.");
        } else {
            System.out.println("Invalid song ID");
        }
    } else {
        System.out.println("Invalid album ID");
    }
    break;
}
```

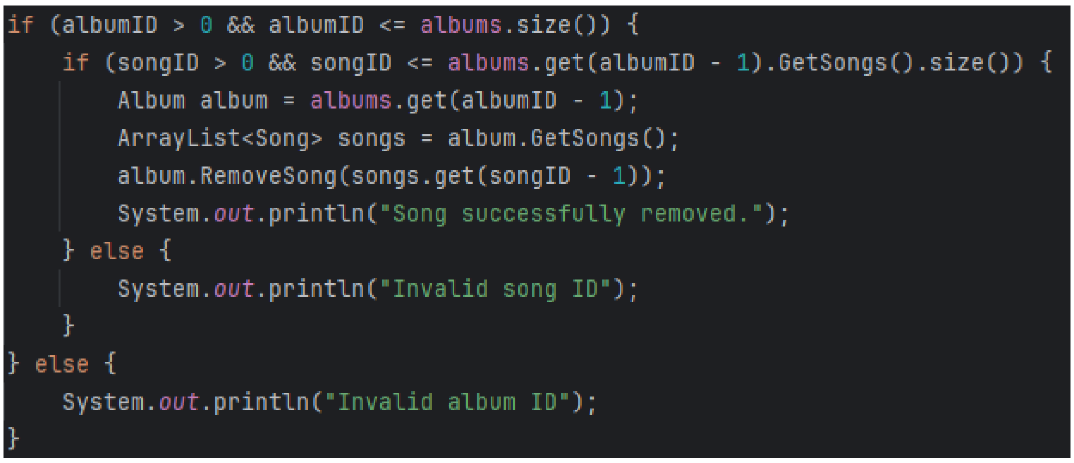


```java
    case 8: { // DELETE SONG
        System.out.print("Enter album ID: ");
        int albumID = scanner.nextInt();
        System.out.print("Enter song ID: ");
        int songID = scanner.nextInt();

        if (albumID > 0 && albumID <= albums.size()) {
            if (songID > 0 && songID <= albums.get(albumID - 1).GetSongs().size()) {
                Album album = albums.get(albumID - 1);
                ArrayList<Song> songs = album.GetSongs();
                album.RemoveSong(songs.get(songID - 1));
                System.out.println("Song successfully removed.");
            } else {
                System.out.println("Invalid song ID");
            }
        } else {
            System.out.println("Invalid album ID");
        }
        break;
    }

    case 9: { // EXIT
        printMenu = false;
        break;
    }
}
```

Opris Liviu Vlad

-if:

ex:

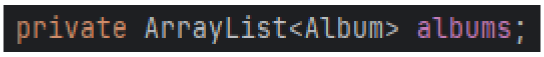


```java
if (albumID > 0 && albumID <= albums.size()) {
    if (songID > 0 && songID <= albums.get(albumID - 1).GetSongs().size()) {
        Album album = albums.get(albumID - 1);
        ArrayList<Song> songs = album.GetSongs();
        album.RemoveSong(songs.get(songID - 1));
        System.out.println("Song successfully removed.");
    } else {
        System.out.println("Invalid song ID");
    }
} else {
    System.out.println("Invalid album ID");
}
```

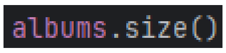
Laboratorul 3:

- **Colectii JAVA:**

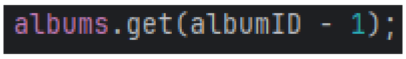
  ex:



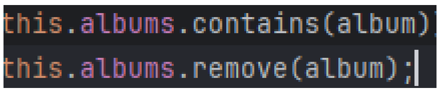
```java
private ArrayList<Album> albums;
```



```java
albums.size()
```



```java
albums.get(albumID - 1);
```



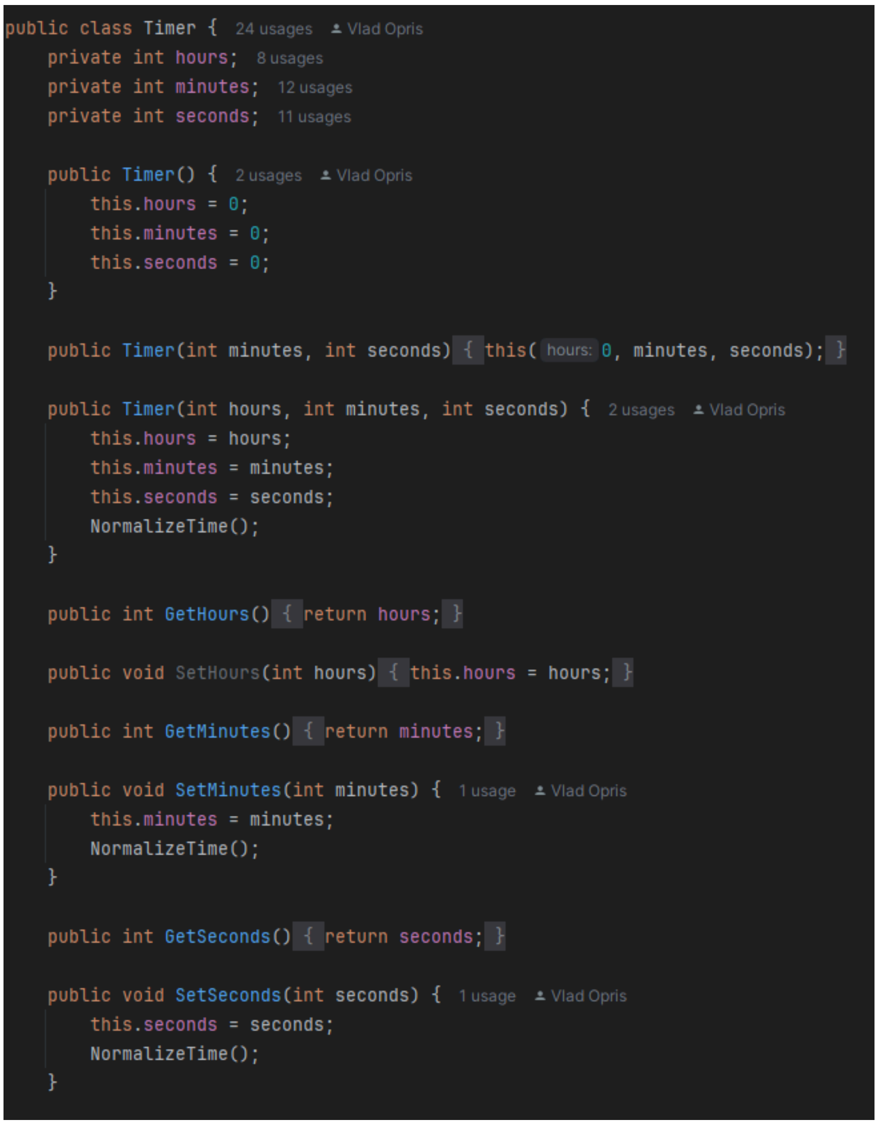
```java
this.albums.contains(album)
this.albums.remove(album);
```

Opris Liviu Vlad

Laboratorul 4:

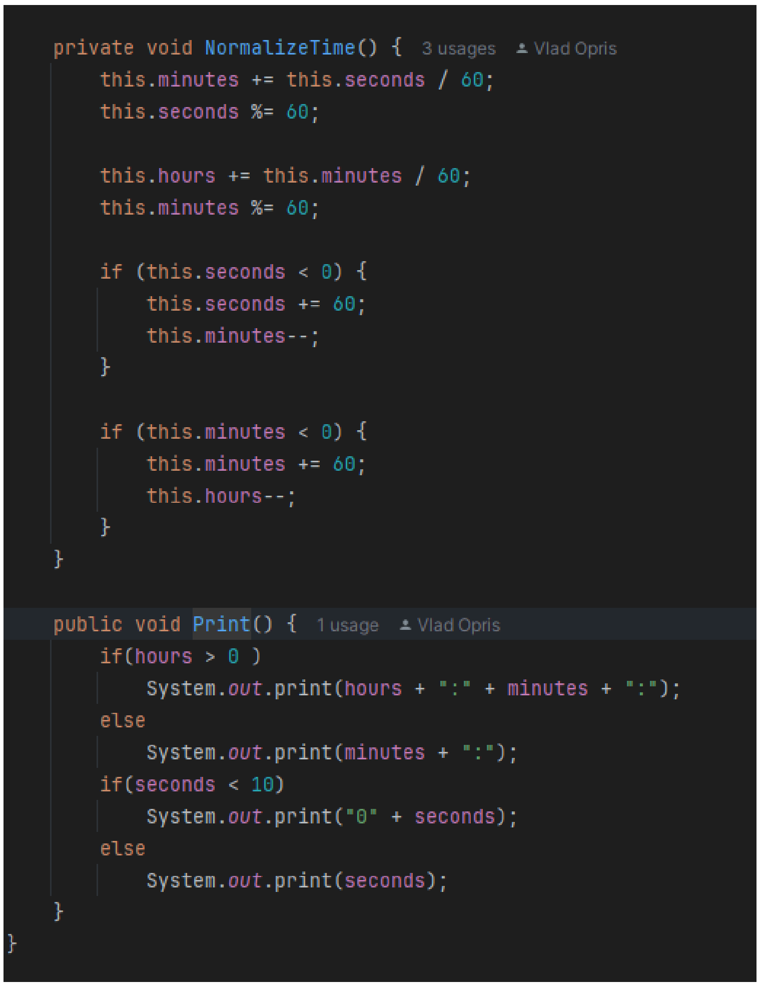
- **Clase:**

  ex:



```
public class Timer {  24 usages  ≗ Vlad Opris
    private int hours;  8 usages
    private int minutes;  12 usages
    private int seconds;  11 usages

    public Timer() {  2 usages  ≗ Vlad Opris
        this.hours = 0;
        this.minutes = 0;
        this.seconds = 0;
    }

    public Timer(int minutes, int seconds) { this( hours: 0, minutes, seconds); }

    public Timer(int hours, int minutes, int seconds) {  2 usages  ≗ Vlad Opris
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
        NormalizeTime();
    }

    public int GetHours() { return hours; }

    public void SetHours(int hours) { this.hours = hours; }

    public int GetMinutes() { return minutes; }

    public void SetMinutes(int minutes) {  1 usage  ≗ Vlad Opris
        this.minutes = minutes;
        NormalizeTime();
    }

    public int GetSeconds() { return seconds; }

    public void SetSeconds(int seconds) {  1 usage  ≗ Vlad Opris
        this.seconds = seconds;
        NormalizeTime();
    }
```

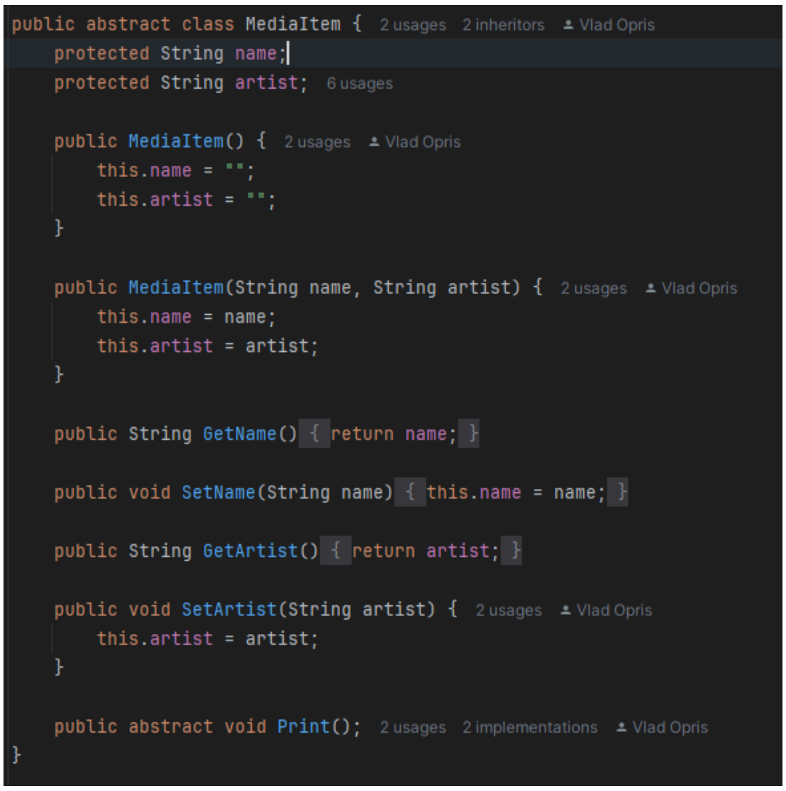
Opris Liviu Vlad


```
    private void NormalizeTime() {  3 usages  Vlad Opris
        this.minutes += this.seconds / 60;
        this.seconds %= 60;

        this.hours += this.minutes / 60;
        this.minutes %= 60;

        if (this.seconds < 0) {
            this.seconds += 60;
            this.minutes--;
        }

        if (this.minutes < 0) {
            this.minutes += 60;
            this.hours--;
        }
    }

    public void Print() {  1 usage  Vlad Opris
        if(hours > 0 )
            System.out.print(hours + ":" + minutes + ":");
        else
            System.out.print(minutes + ":");
        if(seconds < 10)
            System.out.print("0" + seconds);
        else
            System.out.print(seconds);
    }
}
```

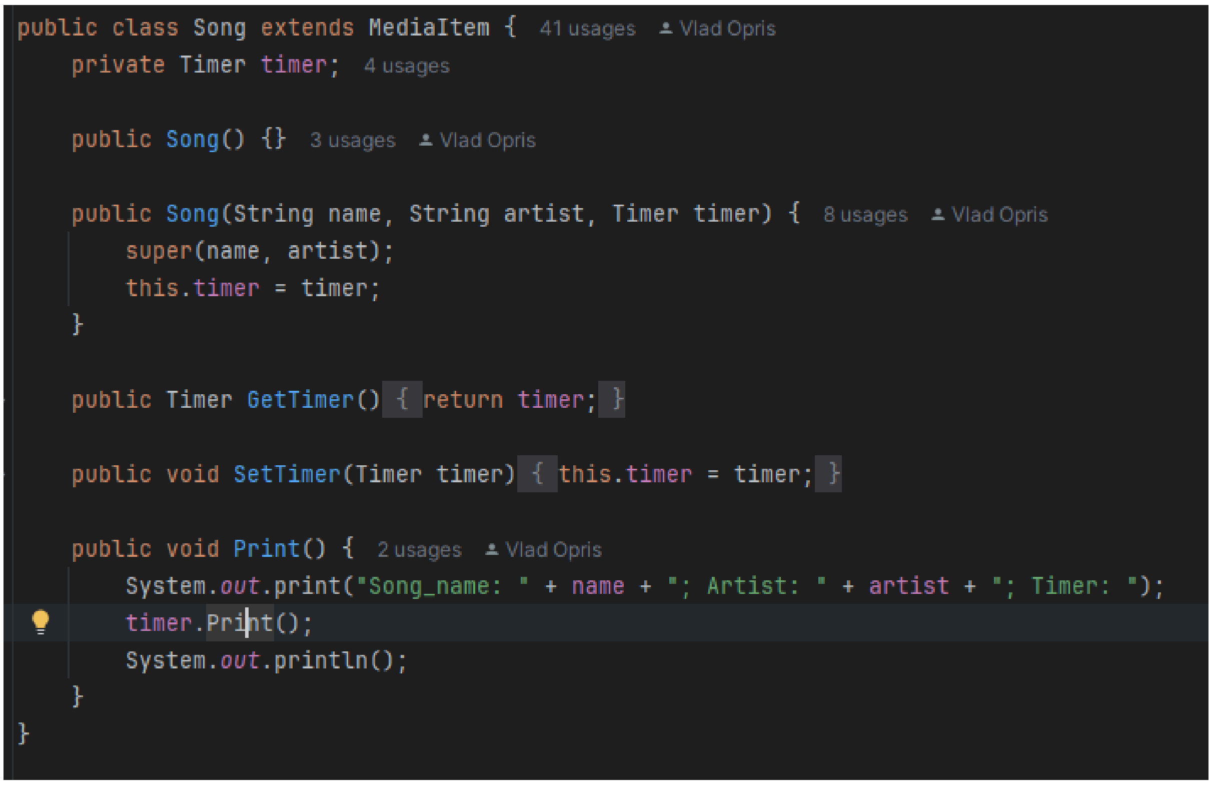
Opris Liviu Vlad

Laboratorul 5:

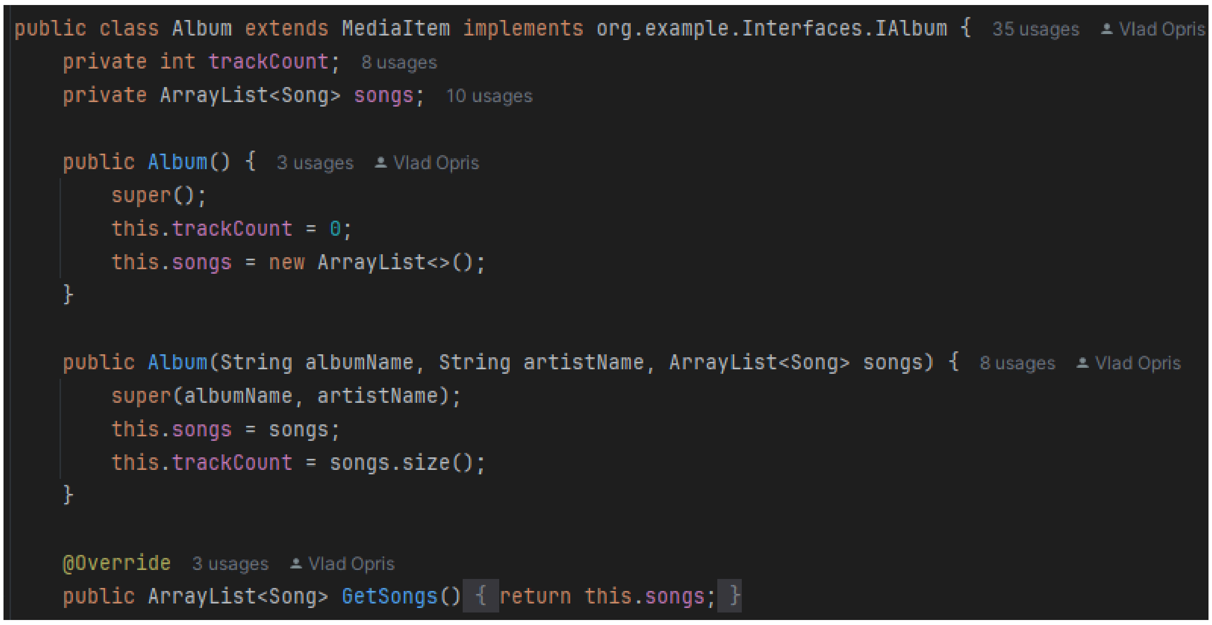
- **Moștenire în Java, clase abstracte:**



```java
public abstract class MediaItem {  2 usages  2 inheritors  Vlad Opris
    protected String name;
    protected String artist;  6 usages

    public MediaItem() {  2 usages  Vlad Opris
        this.name = "";
        this.artist = "";
    }

    public MediaItem(String name, String artist) {  2 usages  Vlad Opris
        this.name = name;
        this.artist = artist;
    }

    public String GetName() { return name; }

    public void SetName(String name) { this.name = name; }

    public String GetArtist() { return artist; }

    public void SetArtist(String artist) {  2 usages  Vlad Opris
        this.artist = artist;
    }

    public abstract void Print();  2 usages  2 implementations  Vlad Opris
}
```



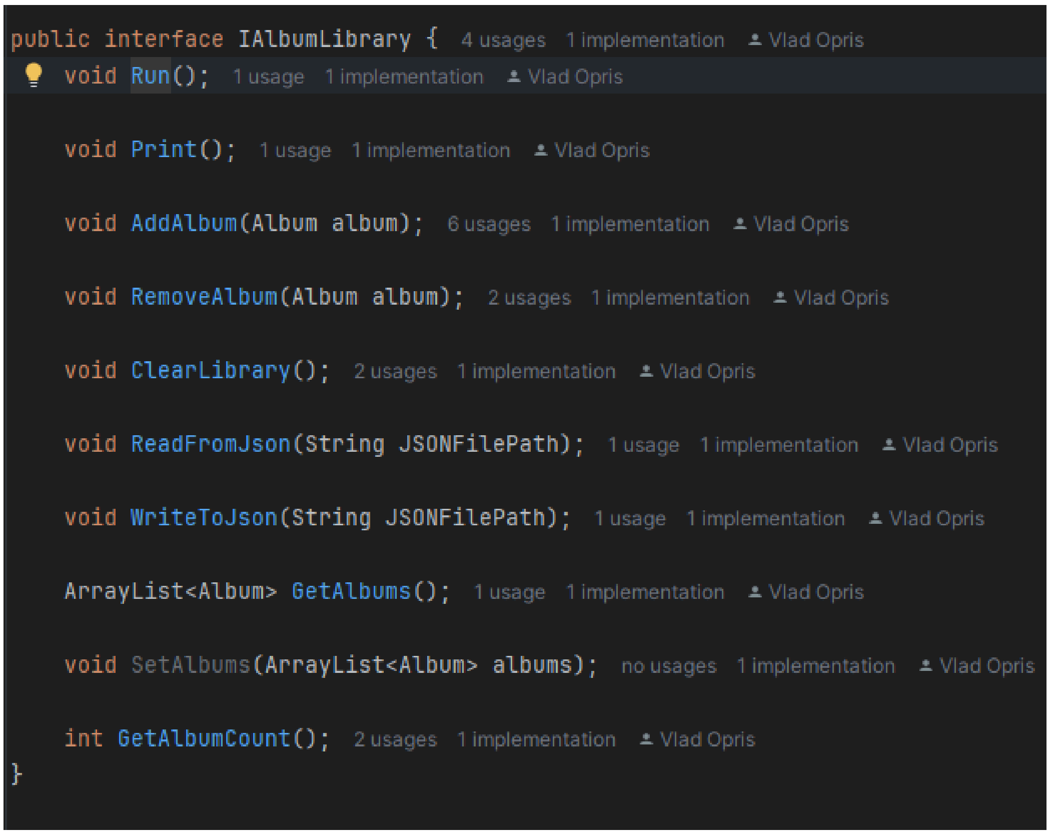
```java
public class Song extends MediaItem {  41 usages  Vlad Opris
    private Timer timer;  4 usages

    public Song() {}  3 usages  Vlad Opris

    public Song(String name, String artist, Timer timer) {  8 usages  Vlad Opris
        super(name, artist);
        this.timer = timer;
    }

    public Timer GetTimer() { return timer; }

    public void SetTimer(Timer timer) { this.timer = timer; }

    public void Print() {  2 usages  Vlad Opris
        System.out.print("Song_name: " + name + "; Artist: " + artist + "; Timer: ");
        timer.Print();
        System.out.println();
    }
}
```



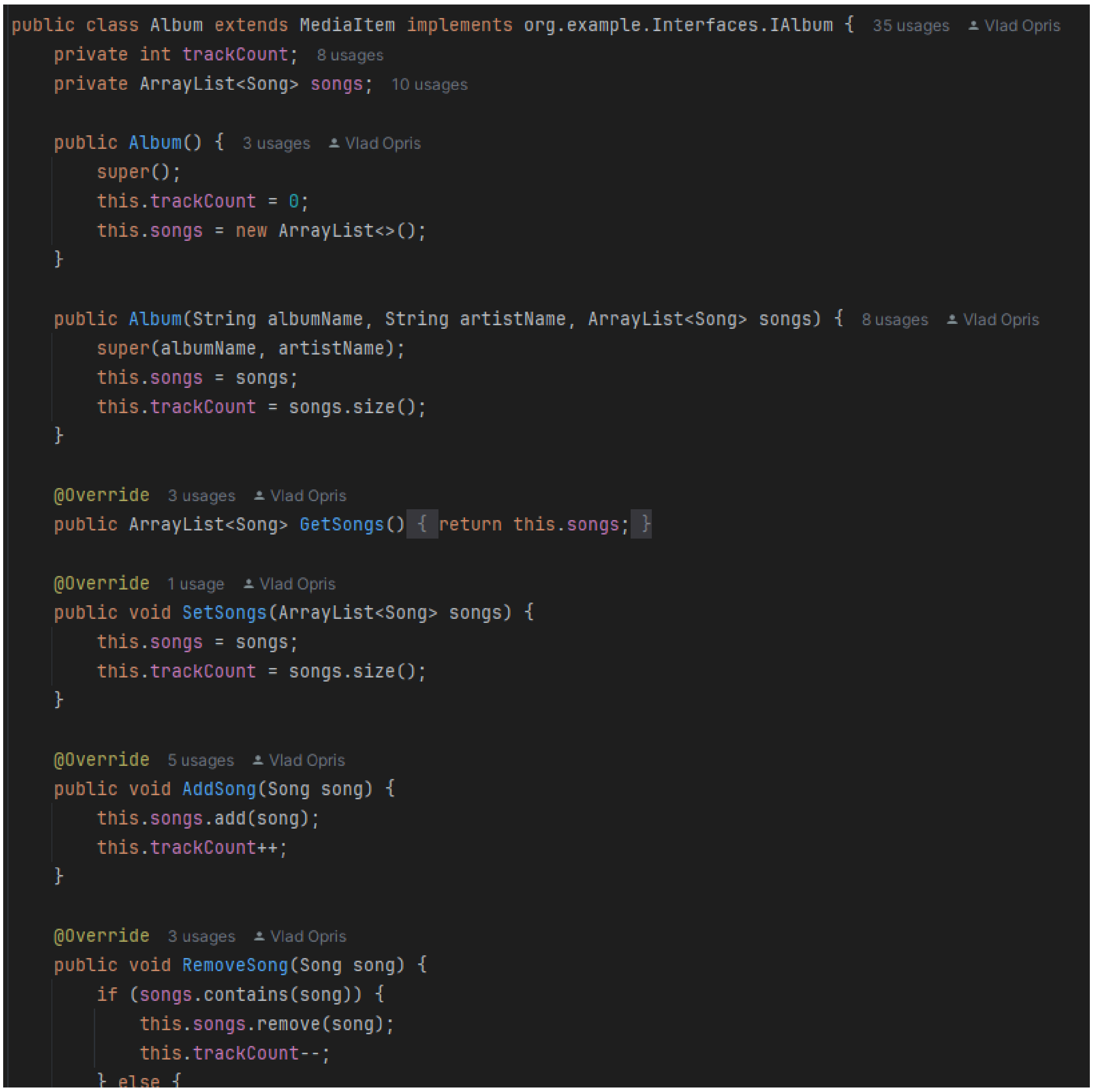
```
public class Album extends MediaItem implements org.example.Interfaces.IAlbum {  35 usages  Vlad Opris
    private int trackCount;  8 usages
    private ArrayList<Song> songs;  10 usages

    public Album() {  3 usages  Vlad Opris
        super();
        this.trackCount = 0;
        this.songs = new ArrayList<>();
    }

    public Album(String albumName, String artistName, ArrayList<Song> songs) {  8 usages  Vlad Opris
        super(albumName, artistName);
        this.songs = songs;
        this.trackCount = songs.size();
    }

    @Override  3 usages  Vlad Opris
    public ArrayList<Song> GetSongs() { return this.songs; }
```

Laboratorul 6:

- **Interfețe în Java**
  ex:



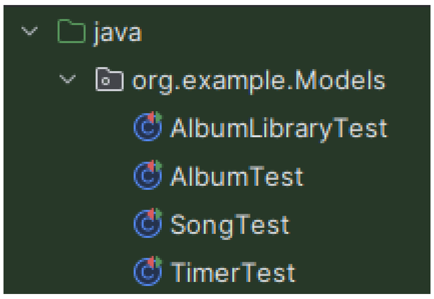
```
public interface IAlbumLibrary {  4 usages  1 implementation  Vlad Opris
    void Run();  1 usage  1 implementation  Vlad Opris

    void Print();  1 usage  1 implementation  Vlad Opris

    void AddAlbum(Album album);  6 usages  1 implementation  Vlad Opris

    void RemoveAlbum(Album album);  2 usages  1 implementation  Vlad Opris

    void ClearLibrary();  2 usages  1 implementation  Vlad Opris

    void ReadFromJson(String JSONFilePath);  1 usage  1 implementation  Vlad Opris

    void WriteToJson(String JSONFilePath);  1 usage  1 implementation  Vlad Opris

    ArrayList<Album> GetAlbums();  1 usage  1 implementation  Vlad Opris

    void SetAlbums(ArrayList<Album> albums);  no usages  1 implementation  Vlad Opris

    int GetAlbumCount();  2 usages  1 implementation  Vlad Opris
}
```



```
public class Album extends MediaItem implements org.example.Interfaces.IAlbum {  35 usages  Vlad Opris
    private int trackCount;  8 usages
    private ArrayList<Song> songs;  10 usages

    public Album() {  3 usages  Vlad Opris
        super();
        this.trackCount = 0;
        this.songs = new ArrayList<>();
    }

    public Album(String albumName, String artistName, ArrayList<Song> songs) {  8 usages  Vlad Opris
        super(albumName, artistName);
        this.songs = songs;
        this.trackCount = songs.size();
    }

    @Override  3 usages  Vlad Opris
    public ArrayList<Song> GetSongs() { return this.songs; }

    @Override  1 usage  Vlad Opris
    public void SetSongs(ArrayList<Song> songs) {
        this.songs = songs;
        this.trackCount = songs.size();
    }

    @Override  5 usages  Vlad Opris
    public void AddSong(Song song) {
        this.songs.add(song);
        this.trackCount++;
    }

    @Override  3 usages  Vlad Opris
    public void RemoveSong(Song song) {
        if (songs.contains(song)) {
            this.songs.remove(song);
            this.trackCount--;
        } else {
```

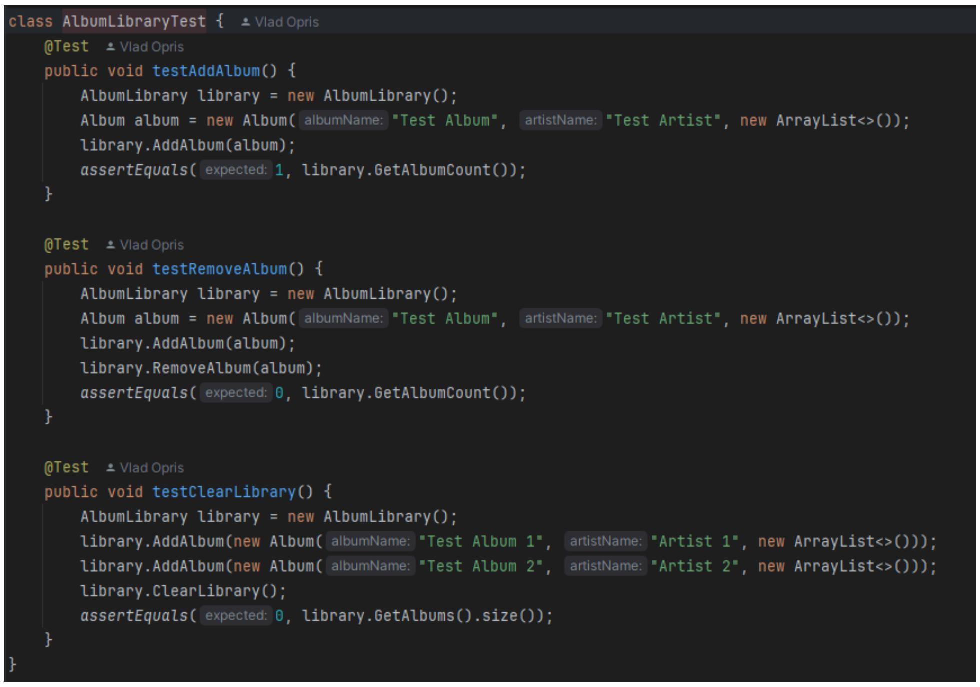
Laboratorul 7:

- **teste:**
  Am facut teste pentru aproape toate functiile din clase.
  ex:

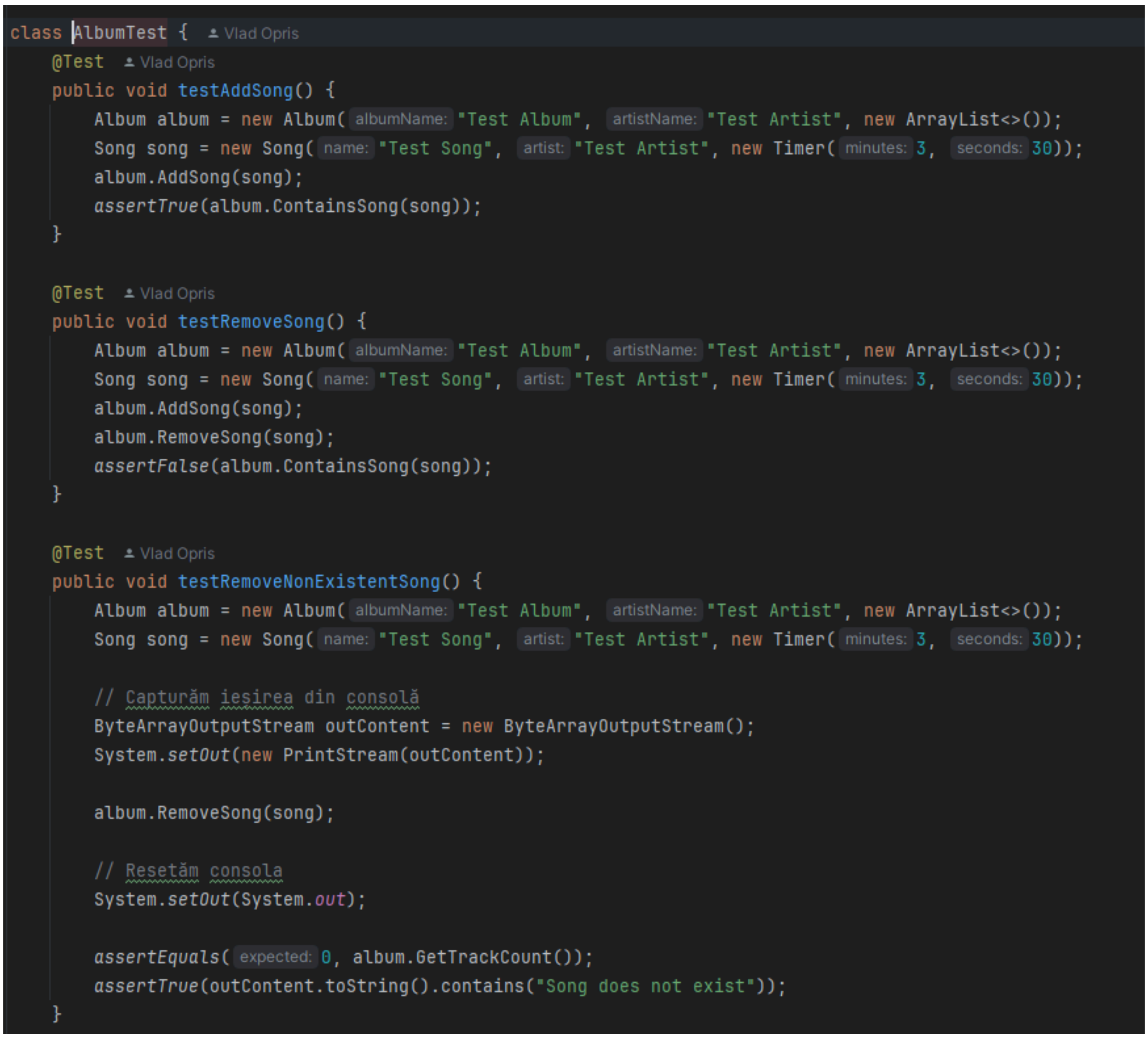


```
java
  org.example.Models
    AlbumLibraryTest
    AlbumTest
    SongTest
    TimerTest
```

Opris Liviu Vlad

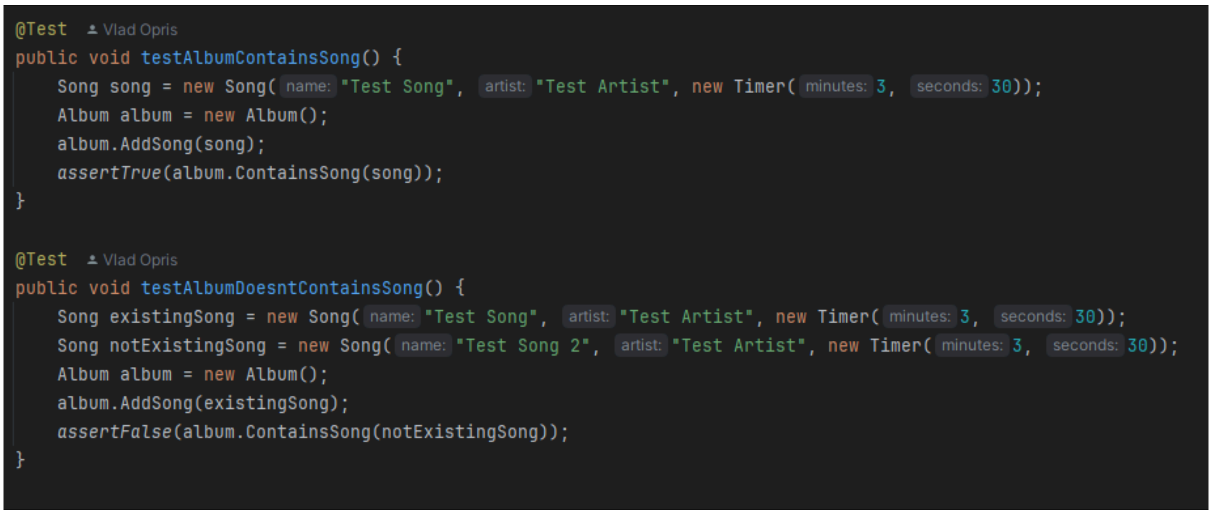


```
class AlbumLibraryTest {  Vlad Opris
    @Test  Vlad Opris
    public void testAddAlbum() {
        AlbumLibrary library = new AlbumLibrary();
        Album album = new Album( albumName: "Test Album",  artistName: "Test Artist", new ArrayList<>());
        library.AddAlbum(album);
        assertEquals( expected: 1, library.GetAlbumCount());
    }

    @Test  Vlad Opris
    public void testRemoveAlbum() {
        AlbumLibrary library = new AlbumLibrary();
        Album album = new Album( albumName: "Test Album",  artistName: "Test Artist", new ArrayList<>());
        library.AddAlbum(album);
        library.RemoveAlbum(album);
        assertEquals( expected: 0, library.GetAlbumCount());
    }

    @Test  Vlad Opris
    public void testClearLibrary() {
        AlbumLibrary library = new AlbumLibrary();
        library.AddAlbum(new Album( albumName: "Test Album 1",  artistName: "Artist 1", new ArrayList<>()));
        library.AddAlbum(new Album( albumName: "Test Album 2",  artistName: "Artist 2", new ArrayList<>()));
        library.ClearLibrary();
        assertEquals( expected: 0, library.GetAlbums().size());
    }
}
```

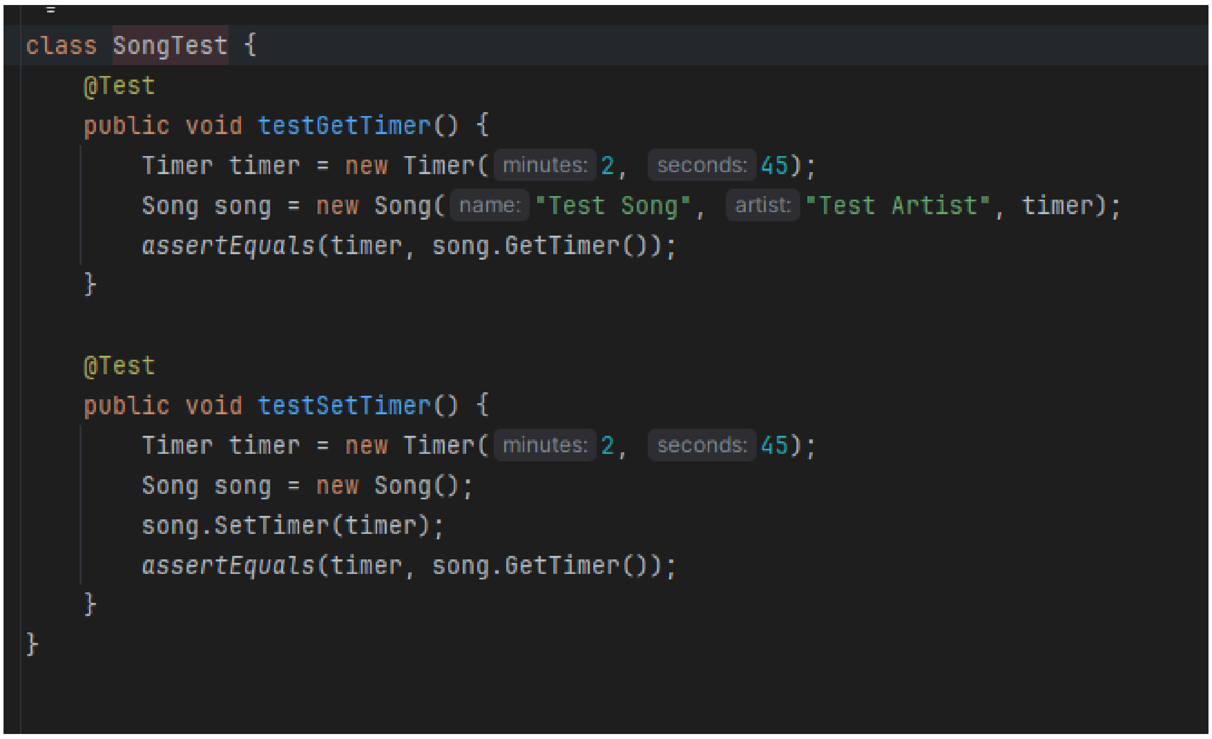


```
class AlbumTest {  Vlad Opris
    @Test  Vlad Opris
    public void testAddSong() {
        Album album = new Album( albumName: "Test Album",  artistName: "Test Artist", new ArrayList<>());
        Song song = new Song( name: "Test Song",  artist: "Test Artist", new Timer( minutes: 3,  seconds: 30));
        album.AddSong(song);
        assertTrue(album.ContainsSong(song));
    }

    @Test  Vlad Opris
    public void testRemoveSong() {
        Album album = new Album( albumName: "Test Album",  artistName: "Test Artist", new ArrayList<>());
        Song song = new Song( name: "Test Song",  artist: "Test Artist", new Timer( minutes: 3,  seconds: 30));
        album.AddSong(song);
        album.RemoveSong(song);
        assertFalse(album.ContainsSong(song));
    }

    @Test  Vlad Opris
    public void testRemoveNonExistentSong() {
        Album album = new Album( albumName: "Test Album",  artistName: "Test Artist", new ArrayList<>());
        Song song = new Song( name: "Test Song",  artist: "Test Artist", new Timer( minutes: 3,  seconds: 30));

        // Capturăm ieșirea din consolă
        ByteArrayOutputStream outContent = new ByteArrayOutputStream();
        System.setOut(new PrintStream(outContent));

        album.RemoveSong(song);

        // Resetăm consola
        System.setOut(System.out);

        assertEquals( expected: 0, album.GetTrackCount());
        assertTrue(outContent.toString().contains("Song does not exist"));
    }
```

Opris Liviu Vlad

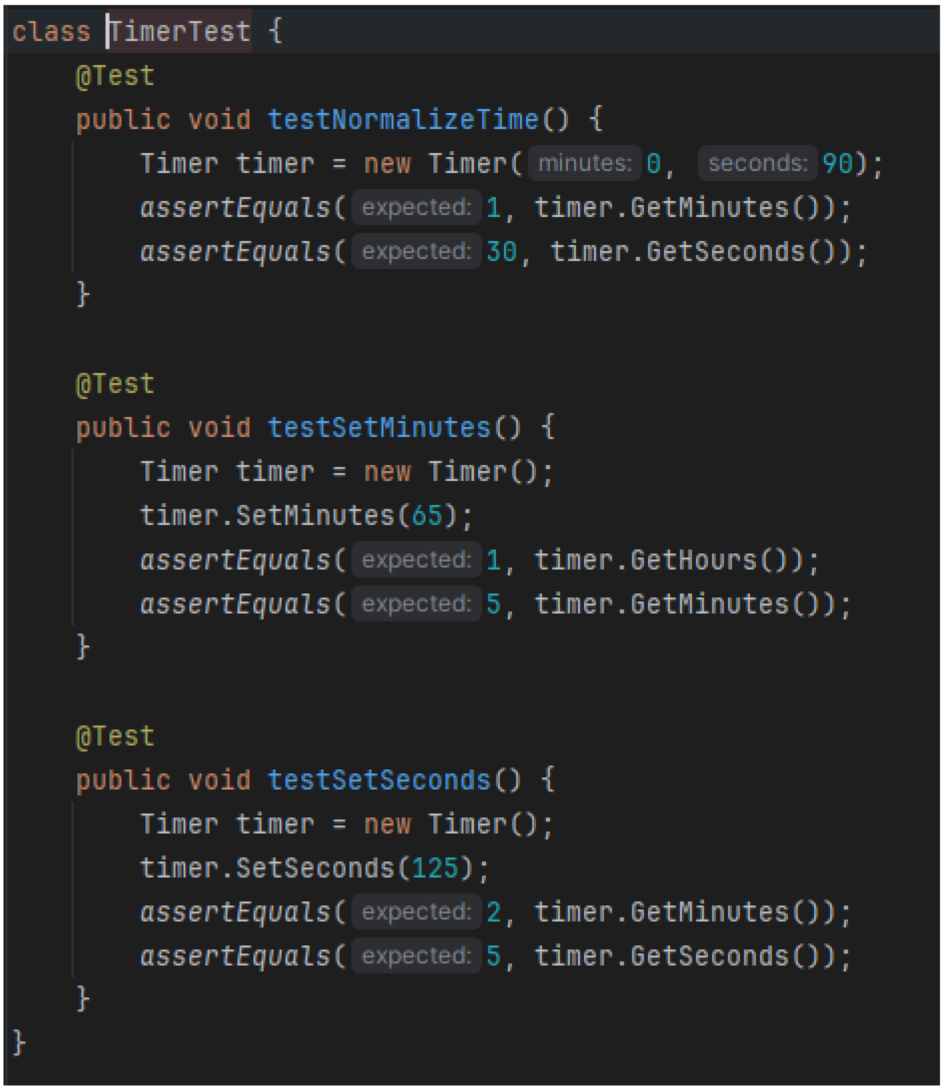


```
@Test  Vlad Opris
public void testAlbumContainsSong() {
    Song song = new Song( name: "Test Song",  artist: "Test Artist", new Timer( minutes: 3,  seconds: 30));
    Album album = new Album();
    album.AddSong(song);
    assertTrue(album.ContainsSong(song));
}

@Test  Vlad Opris
public void testAlbumDoesntContainsSong() {
    Song existingSong = new Song( name: "Test Song",  artist: "Test Artist", new Timer( minutes: 3,  seconds: 30));
    Song notExistingSong = new Song( name: "Test Song 2",  artist: "Test Artist", new Timer( minutes: 3,  seconds: 30));
    Album album = new Album();
    album.AddSong(existingSong);
    assertFalse(album.ContainsSong(notExistingSong));
}
```



```
class SongTest {
    @Test
    public void testGetTimer() {
        Timer timer = new Timer( minutes: 2,  seconds: 45);
        Song song = new Song( name: "Test Song",  artist: "Test Artist", timer);
        assertEquals(timer, song.GetTimer());
    }

    @Test
    public void testSetTimer() {
        Timer timer = new Timer( minutes: 2,  seconds: 45);
        Song song = new Song();
        song.SetTimer(timer);
        assertEquals(timer, song.GetTimer());
    }
}
```

Opris Liviu Vlad



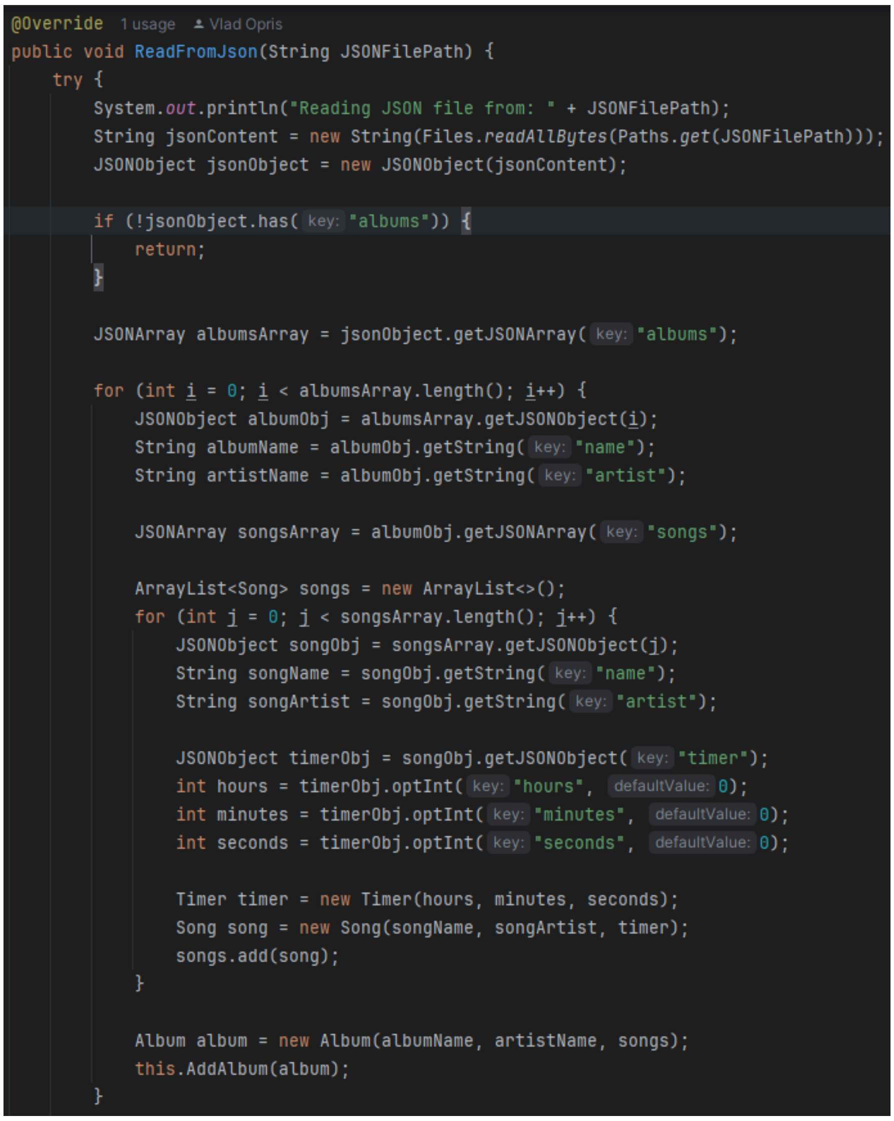
```
class TimerTest {
    @Test
    public void testNormalizeTime() {
        Timer timer = new Timer( minutes: 0,  seconds: 90);
        assertEquals( expected: 1, timer.GetMinutes());
        assertEquals( expected: 30, timer.GetSeconds());
    }

    @Test
    public void testSetMinutes() {
        Timer timer = new Timer();
        timer.SetMinutes(65);
        assertEquals( expected: 1, timer.GetHours());
        assertEquals( expected: 5, timer.GetMinutes());
    }

    @Test
    public void testSetSeconds() {
        Timer timer = new Timer();
        timer.SetSeconds(125);
        assertEquals( expected: 2, timer.GetMinutes());
        assertEquals( expected: 5, timer.GetSeconds());
    }
}
```

Opris Liviu Vlad

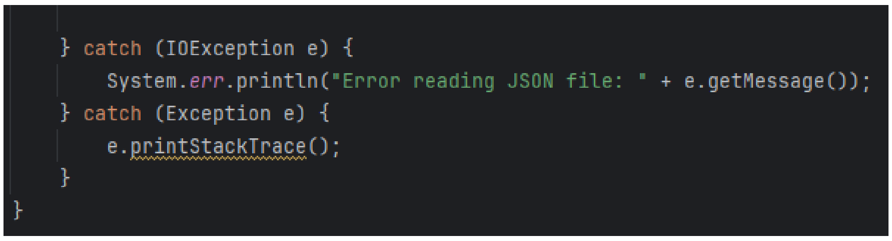
Laboratorul 8:

- **persistenta datelor:**

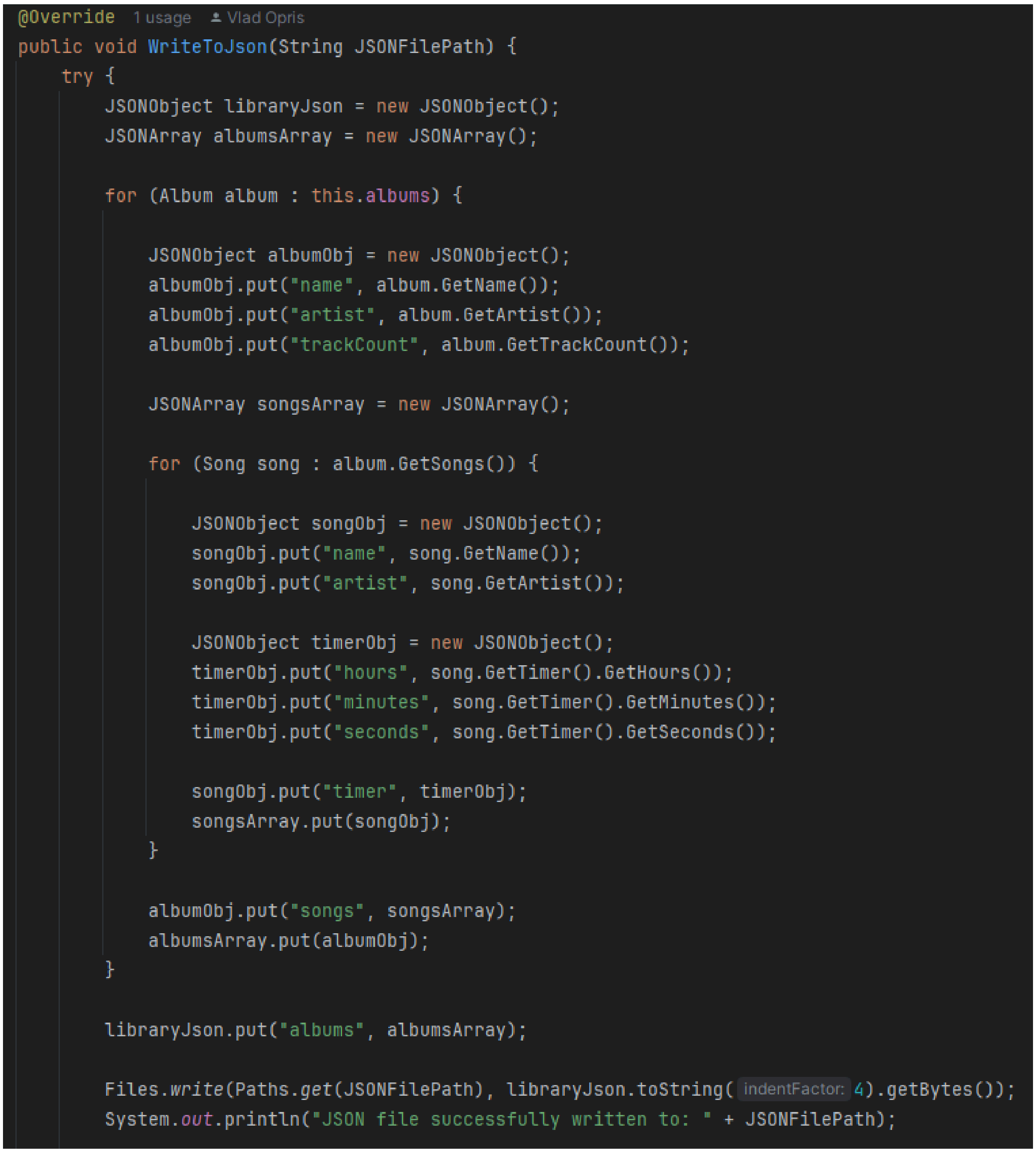
  Am creat doua functii pentru salvarea datelor si anume: ReadFromJson() si WriteToJson().



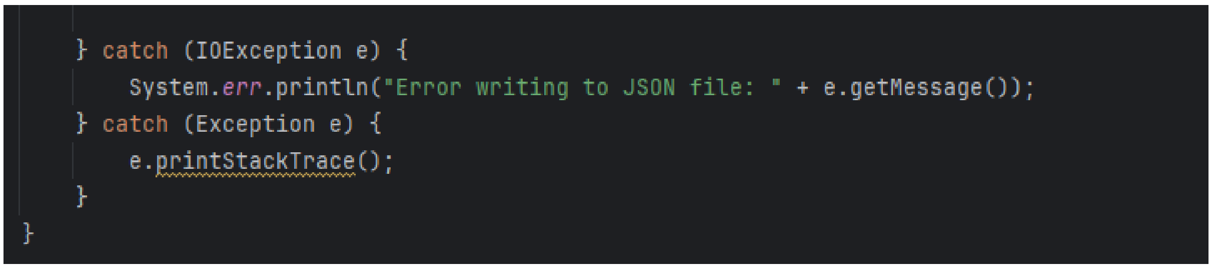
```
@Override  1 usage  Vlad Opris
public void ReadFromJson(String JSONFilePath) {
    try {
        System.out.println("Reading JSON file from: " + JSONFilePath);
        String jsonContent = new String(Files.readAllBytes(Paths.get(JSONFilePath)));
        JSONObject jsonObject = new JSONObject(jsonContent);

        if (!jsonObject.has( key: "albums")) {
            return;
        }

        JSONArray albumsArray = jsonObject.getJSONArray( key: "albums");

        for (int i = 0; i < albumsArray.length(); i++) {
            JSONObject albumObj = albumsArray.getJSONObject(i);
            String albumName = albumObj.getString( key: "name");
            String artistName = albumObj.getString( key: "artist");

            JSONArray songsArray = albumObj.getJSONArray( key: "songs");

            ArrayList<Song> songs = new ArrayList<>();
            for (int j = 0; j < songsArray.length(); j++) {
                JSONObject songObj = songsArray.getJSONObject(j);
                String songName = songObj.getString( key: "name");
                String songArtist = songObj.getString( key: "artist");

                JSONObject timerObj = songObj.getJSONObject( key: "timer");
                int hours = timerObj.optInt( key: "hours",  defaultValue: 0);
                int minutes = timerObj.optInt( key: "minutes",  defaultValue: 0);
                int seconds = timerObj.optInt( key: "seconds",  defaultValue: 0);

                Timer timer = new Timer(hours, minutes, seconds);
                Song song = new Song(songName, songArtist, timer);
                songs.add(song);
            }

            Album album = new Album(albumName, artistName, songs);
            this.AddAlbum(album);
        }
```



```
    } catch (IOException e) {
        System.err.println("Error reading JSON file: " + e.getMessage());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



```java
@Override  1 usage  Vlad Opris
public void WriteToJson(String JSONFilePath) {
    try {
        JSONObject libraryJson = new JSONObject();
        JSONArray albumsArray = new JSONArray();

        for (Album album : this.albums) {

            JSONObject albumObj = new JSONObject();
            albumObj.put("name", album.GetName());
            albumObj.put("artist", album.GetArtist());
            albumObj.put("trackCount", album.GetTrackCount());

            JSONArray songsArray = new JSONArray();

            for (Song song : album.GetSongs()) {

                JSONObject songObj = new JSONObject();
                songObj.put("name", song.GetName());
                songObj.put("artist", song.GetArtist());

                JSONObject timerObj = new JSONObject();
                timerObj.put("hours", song.GetTimer().GetHours());
                timerObj.put("minutes", song.GetTimer().GetMinutes());
                timerObj.put("seconds", song.GetTimer().GetSeconds());

                songObj.put("timer", timerObj);
                songsArray.put(songObj);
            }

            albumObj.put("songs", songsArray);
            albumsArray.put(albumObj);
        }

        libraryJson.put("albums", albumsArray);

        Files.write(Paths.get(JSONFilePath), libraryJson.toString( indentFactor: 4).getBytes());
        System.out.println("JSON file successfully written to: " + JSONFilePath);
```



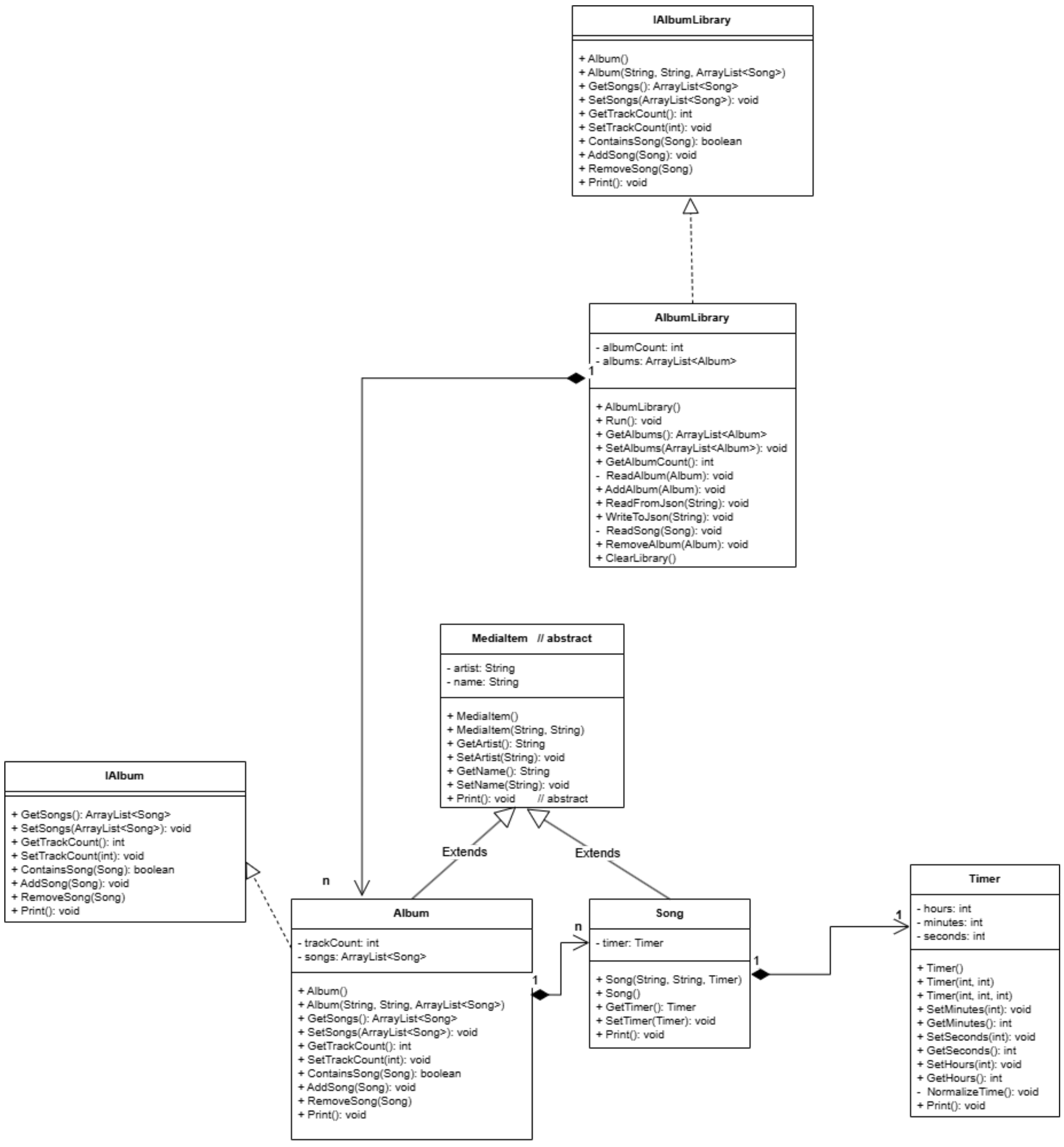
```java
    } catch (IOException e) {
        System.err.println("Error writing to JSON file: " + e.getMessage());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Pentru ca programul sa citeasca din JSON datele salvate la ultima rulare a codului, fișierele de input și de output trebuie sa coincida.

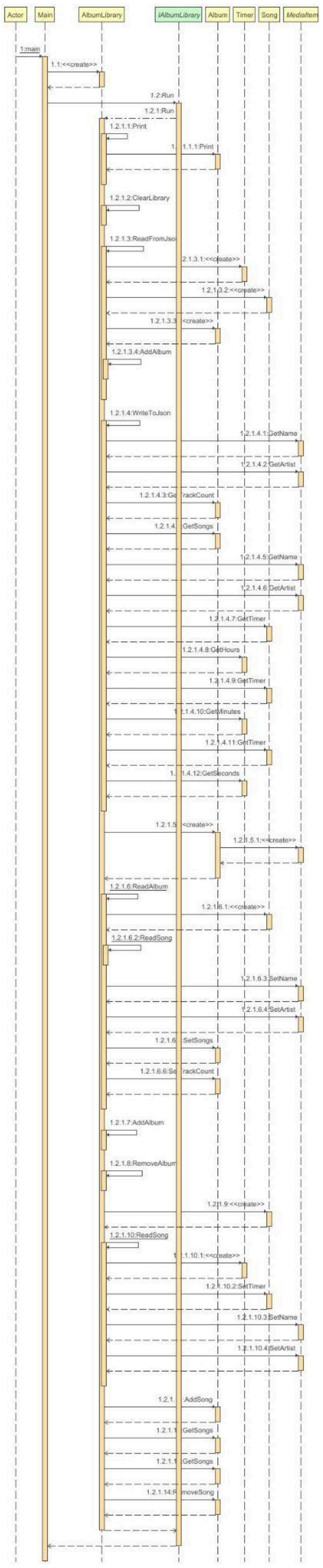
Opris Liviu Vlad

Laboratorul 9:

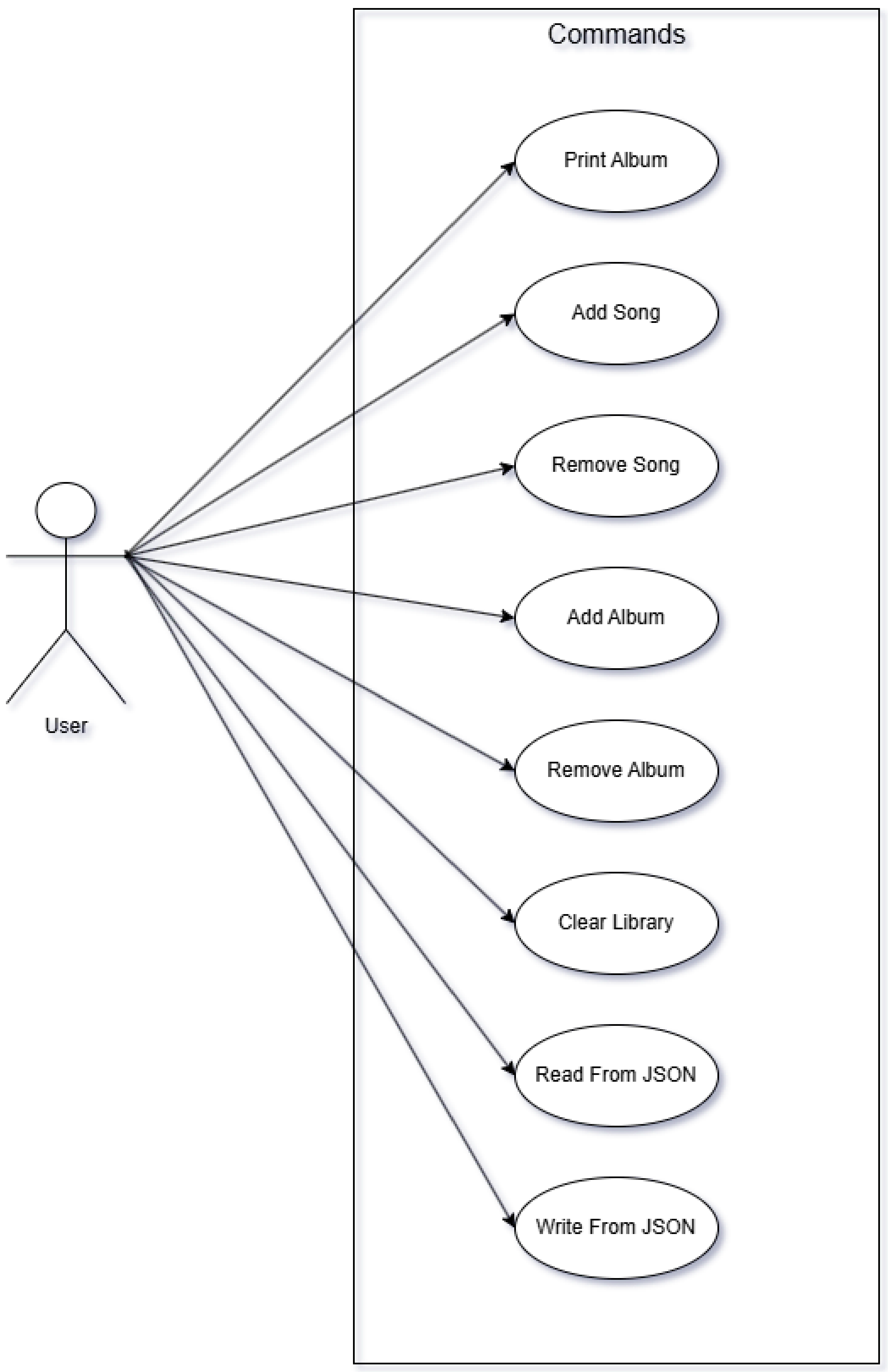
- **Diagrame UML:**



Opris Liviu Vlad



Opris Liviu Vlad


Commands
Print Album
Add Song
Remove Song
Add Album
Remove Album
Clear Library
Read From JSON
Write From JSON
User