Java 1. Beadandó

Kertész Domonkos PB8JV3

Contents

Main

A main-ben a program működését irányító 5 metódus van meghívva.

```java
public static void main(String[] args) {
    try {
        /**
         * Generates directories and files if they are non-existent, fills the data file with songs in random order if generated
         */
        FileTreeGenerator.init();

        /**
         * Initializes menu system
         * Loads songs into song database
         * Generates album data from loaded songs, loads it into album database
         * Loads playlists into playlist database
         */
        MainMenuController menu = new MainMenuController(new DataManager(ReadFromFile.readSongs(), ReadFromFile.readPlaylists()));

        /**
         * Initializes the menu system
         */
        menu.init();

        /**
         * Writes songs data to data file
         */
        WriteToFile.write(menu.getDataManager().getSongManager());

        /**
         * Writes each playlist into their corresponding file
         */
        WriteToFile.write(menu.getDataManager().getPlaylistManager());

    } catch (FailedDirectoryCreationException | FailedFileCreationException | FileNotFoundException ex) {
        System.out.println(ex);
    }
}
```

Figure 1:Main Osztály

Adat tárolás

Single Entry adat tárolás

Song

```java
public class Song {
    private String name;
    private String artist;
    private String style;
    private Integer length;
    private String albumName;

    public Song() {
    }

    public Song(String name, String artist, String style, Integer lenght, String albumName) {
        this.name = name;
        this.artist = artist;
        this.style = style;
        this.length = lenght;
        this.albumName = albumName;
    }
```

Figure 2:Song Osztály

A Song osztályban egy zeneszám adatait tároljuk. Az osztálynak van egy üres és egy minden adatot bekérő konstruktora, az adattagokhoz settere és gettere.

public String printable(): Visszaadja az adattagokat egy stringben ’;’-vel elválasztva.

public void printAllData(): Kinyomtatja az adattagokat.

Album

```java
public class Album {
    private final String name;
    private final List<Song> songs;

    public Album(String name, Song song) {
        this.name = name;
        songs = new ArrayList<>();
        songs.add(song);
    }

    public Album addSong(Song song) {
        songs.add(song);
        return this;
    }

    public void listSongs() {
        songs.forEach(song -> {System.out.println(song.getName());});
    }

    public void playSongs() {
        songs.forEach(song -> {System.out.println("Now playing: " + song.getName());});
    }

    public String getName() {
        return name;
    }

    public List<Song> getSongs() {
        return songs;
    }
}
```

Az Album osztály egy album adatait tárolja, az album nevét Stringben és zenéket listában. Az osztály konstruktora az album nevét és egy zeneszámot vár. Az adattagonak van gettere.

public void listSongs(): Kilistázza az albumban lévő zenék neveit.

public void playSongs(): Formázva kilistázza az albumban lévő zenék neveit.

Playlist

```java
public class Playlist {
    private String name;
    private final List<Song> songs;

    public Playlist(String name){
        this.name = name;
        songs = new ArrayList<>();
    }

    public Playlist(String name, List<Song> songs){
        this.name = name;
        this.songs = songs;
    }

    public String getName() {
        return name;
    }

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

    public List<Song> getSongs() {
        return songs;
    }

    public void addSong(Song song){
        songs.add(song);
    }

    public void removeSong(Song song){
        songs.stream().filter(songInList -> (songInList.equals(song))).forEachOrdered(songInList -> {songs.remove(songInList);});
    }

    public void listSongs(){
        if(!songs.isEmpty())
            songs.forEach(song -> {System.out.println(song.getName());});
        else
            System.out.println("Playlist is empty, consider addig songs to it from the data editor menu");
    }

    public void playSongs(){
        if(!songs.isEmpty())
            songs.forEach(song -> {System.out.println("Now playing: " + song.getName());});
        else
            System.out.println("Playlist is empty, consider addig songs to it from the data editor menu");
    }
```

Figure 3:Playlist Osztály

A Playlist osztály egy lajátszási lista adatait tárolja, a nevét Stringben, és a zenéket listában. Az osztálynak két konstruktora van, az egyik egy Stringet vár névnek, a másik egy Stringet névnek és egy listát, ami zenéket tartalmaz. Az adattagoknak van gettere, és a String name-nek van settere.

public void addSong(Song song): Hozzáadja a paraméterben lévő zenét a listához.

public void removeSong(Song song): Kitörli a paraméterben lévő zenét a listából.

public void listSongs(): Kilistázza a zenék neveit, ha üres a lista akkor arról tájékoztat.

public void playSongs(): Kilistázza a zenék neveit formázva, ha üres a lista akkor arról tájékoztat.

Entry tárolás

SongManager

```java
public class SongManager {
    private final List<Song> songs;

    public SongManager() {
        songs = new ArrayList<>();
    }

    public SongManager(List<Song> songs) {
        this.songs = songs;
    }

    public SongManager addSong(Song song) {
        songs.add(song);
        return this;
    }

    public List<Song> getSongs() {
        return songs;
    }

    public void listSongs(){
        songs.forEach(song -> System.out.println(song.getName()));
    }
}
```

Figure 4: SongManager osztály

A SongManager osztály az összes Song objektumot tárolja egy listában. Az osztálynak két konstruktora van, az egyik üres konstruktor, a másik egy zenékből álló listát vár. A listának van gettere.

public SongManager addSong(Song song): Hozzáad a listához egy zenét.

public void listSongs(): Kilistázza a zenéket

AlbumManager

```
public class AlbumManager {
    private Map<String, Album> albums;

    public AlbumManager(SongManager songManager) {
        albums = new HashMap();
        songManager.getSongs().forEach(song -> {
            if(albums.containsKey(song.getAlbumName())){
                albums.replace(song.getAlbumName(), albums.get(song.getAlbumName()).addSong(song));
            } else {
                albums.put(song.getAlbumName(), new Album(song.getAlbumName(), song));
            }
        });
    }

    public Album getAlbum(String name){
        if(albums.containsKey(name))
            return albums.get(name);
        else
            return null;
    }

    public void listAlbums() {
        albums.keySet().forEach(albumName -> {System.out.println(albumName);});
    }
}
```

Figure 5:AlbumManager Osztály

Az album manager a SongManagerben lévő zenék alapján kigenerálja az albumokat, és menti őket egy Map-be, aminek az kulcsa az Album neve, értéke pedig maga az album. Az osztály konstruktora a SongManager-t várja, végig iterál a zenéken és a megfelelő albumba menti őket, vagy új albumot készít nekik.

public Album getAlbum(String name): Név alapján le lehet kérni albumot.

public void listAlbums(): Kilistázza az albumokat.

PlaylistManager

```java
public class PlaylistManager {
    Map<String, Playlist> playlists;

    public PlaylistManager(Map<String, Playlist> playlists) {
        this.playlists = playlists;
    }

    public Map<String, Playlist> getPlaylists() {
        return playlists;
    }

    public PlaylistManager addPlaylist(String name, List<Song> songs){
        playlists.put(name, new Playlist(name, songs));
        return this;
    }

    public Playlist getPlaylist(String name){
        if(playlists.containsKey(name))
            return playlists.get(name);
        else
            return null;
    }

    public void removePlaylist(String name){
        playlists.remove(name);
    }

    public PlaylistManager modifyPlaylist(String name, List<Song> songs){
        playlists.replace(name, new Playlist(name, songs));
        return this;
    }

    public void listPlaylists(){
        playlists.keySet().forEach(name -> System.out.println(name));
    }
}
```

Figure 6:PlaylistManager Osztály

A PlaylistManager osztály az összes playlistet tárolja egy Map-ben, aminek a kulcsa a lejátszási lista neve, az értéke pedig maga a lejátszási lista. Az osztály konstruktora a Map-ot várja és elmenti. A Map-nek van gettere.

public PlaylistManager addPlaylist(String name, List<Song> songs): Elmenti egy playlistet a Map-be a megadott névből és zene listából.

public Playlist getPlaylist(String name): Visszaérít playlistet a megadott név alapján, vagy null-t ha nem létezik.

public void removePlaylist(String name): Kitöröl egy playlistet a megadott név alapján.

public PlaylistManager modifyPlaylist(String name, List<Song> songs): Modosítja név alapján a playlistet a megadott listára, majd visszatéríti a PlaylistManager objektumot.

public void listPlaylists(): Kilistázza a playlistek neveit.

Entry tároló tárolás

```
public class DataManager {
    SongManager songManager;
    AlbumManager albumManager;
    PlaylistManager playlistManager;

    public DataManager(SongManager songManager, PlaylistManager playlistManager) {
        this.songManager = songManager;
        this.albumManager = new AlbumManager(songManager);
        this.playlistManager = playlistManager;
    }

    public SongManager getSongManager() {
        return songManager;
    }

    public AlbumManager getAlbumManager() {
        return albumManager;
    }

    public PlaylistManager getPlaylistManager() {
        return playlistManager;
    }

    public void setSongManager(SongManager songManager) {
        this.songManager = songManager;
    }

    public void setAlbumManager(AlbumManager albumManager) {
        this.albumManager = albumManager;
    }

    public void setPlaylistManager(PlaylistManager playlistManager) {
        this.playlistManager = playlistManager;
    }
```

Figure 7:DataManager osztály

A DataManager osztály tárolja a SongManagert, az AlbumManagert, és a PlaylistManagert. Az osztály konstruktora beállítja az adattagokat. Az adattagoknak van settere és gettere.

Fájl kezelés

Fáljból olvasás

```
public class ReadFromFile {
    private ReadFromFile(){
    }

    public static SongManager readSongs() throws FileNotFoundException{
        File songsFile = new File(FileLocation.getDataFileLocation());
        List<Song> songs = new ArrayList<>();
        Scanner fileScanner = new Scanner(songsFile);
        while(fileScanner.hasNextLine()) {
            String line = fileScanner.nextLine();
            if(!line.isEmpty()){
                songs.add(ReadFromFile.songFromString(line));
            }
        }
        return new SongManager(songs);
    }

    public static PlaylistManager readPlaylists(){
        Map<String, Playlist> playlists = new HashMap();
        try {
            File folder = new File(FileLocation.getPlaylistPath().toString());
            File[] listOfPlaylists = folder.listFiles();

            for(File playlist : listOfPlaylists) {
                playlists.put(playlist.getName().substring(0, playlist.getName().length()-4), ReadFromFile.playlistFromFile(playlist.getName().substring(0, playlist.getName().length()-4), playlist));
            }

            return new PlaylistManager(playlists);
        } catch (NullPointerException e){
            return null;
        } catch (FileNotFoundException e){
            System.out.println("Some error occured while reading playlists: " + e.getMessage());
            return null;
        }
    }
```

Figure 8:ReadFromFile Osztály

A fáljból olvasást a ReadFromFile osztály végzi, az osztály nem inicializálható, private a konstruktora, és static függvényeket tartamaz.

public static SongManager readSongs(): Beolvassa a zenéket a az adatot tároló fájlból, elmenti SongManagerbe és visszadja. FileNotFoundExceptiont dob.

public static PlaylistManager readPlaylists(): Beolvassa a fájlokat a playlisteket tartalmazó mappából playlistFromFile() segítségével, elmenti őket PlaylistManagerbe és visszaadja.

private static Song songFromString(String rawString): Egy sor adatot kap Stringként, szétbontja menti egy Song-ba majd visszatéríti.

private static Playlist playlistFromFile(String name, File playlistFile): Megkap egy fálj nevet és egy filet, elmenti a fileban lévő zenéket egy playlistbe majd visszatéríti.

Fáljba írás

```java
public class WriteToFile {

    private WriteToFile() {
    }

    public static void write(SongManager songManager) {
        try (FileWriter writer = new FileWriter(FileLocation.getDataFileLocation())) {
            songManager.getSongs().forEach(song -> {
                try {
                    writer.write(song.printable());
                    writer.write(System.lineSeparator());
                } catch (IOException e) {
                    System.out.println("An error occurred: " + e.getMessage());
                }
            });
            System.out.println("Songs written to " + FileLocation.getDataFileName());
            writer.close();
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }

    public static void write(PlaylistManager playlistManager) {
        playlistManager.getPlaylists().values().forEach(playlist -> {
            try (FileWriter writer = new FileWriter(FileLocation.getPlaylistPath().toString() + "/" + playlist.getName() + ".txt")) {
                playlist.getSongs().forEach(song -> {
                    try {
                        writer.write(song.printable());
                        writer.write(System.lineSeparator());
                    } catch (IOException e) {
                        System.out.println("An error occurred: " + e.getMessage());
                    }
                });
                System.out.println("Songs written to " + FileLocation.getPlaylistPath().toString() + "/" + playlist.getName() + ".txt");
                writer.close();
            } catch (IOException e) {
                System.out.println("An error occurred: " + e.getMessage());
            }
        });
    }
}
```

Figure 9:WriteToFile Osztály

A fáljba írást a WriteToFile osztály végzi, az osztály nem inicializálható, private a konstruktora, és egy overoladolt static függvényt tartamaz.

public static void write(SongManager songManager): Megkapja a SongManagert és kiírja a zenéket tároló fájlba.

public static void write(PlaylistManager playlistManager): Megkapja a PlaylistManagert és a playlisteket elmenti a megfelelő fáljba, ha nincs még fájl akkor generál egyet, a fálj neve a playlist neve lesz.

Fálj kezelés Utility-k

```java
public class FileTreeGenerator {

    private FileTreeGenerator() {
    }

    public static void init() throws FailedDirectoryCreationException, FailedFileCreationException{
        try {
            Files.createDirectories(FileLocation.getPlaylistPath());
            System.out.println("Data directory created or already exists");
            System.out.println("Playlist directory created or already exists");
        } catch (IOException e) {
            throw new FailedDirectoryCreationException("Failed to create directory! " + e.getMessage());
        }

        try {
            File songsFile = new File(FileLocation.getDataFileLocation());
            if(songsFile.createNewFile()){
                System.out.println("File created in data directory: songs.txt");
                WriteToFile.write(new SongManager(SongsGenerator.generateSongs()));
            } else {
                System.out.println("File lready exists: songs.txt");
            }
        } catch (IOException e) {
            throw new FailedFileCreationException("Failed to create file: songs.txt "+ e.getMessage());
        }
    }
}
```

Figure 10:FileTreeGenerator Osztály

public class FailedDirectoryCreationException: Akkor kerül dobásra, ha nem sikerül legenerálni a mappát.

Public class FailedFileCreationException: Akkor kerül dobásra, ha nem sikerül legenerálni a fájlt.

public class FileLocation: Tárolja a fájlok a mappák nevét, generál hozzájuk elérési útvonalat.

public class FileTreeGenerator: Kigenerálja a mappákat és ha nincs zenéket tároló fájl generál egyet és feltölti zenékkel random sorrendben a SongsGenerator segítségével. FailedDirectoryCreationException-t, FailedFileCreationException-t tud dobni.

public class SongsGenerator: egy static függvénye van ami visszatérít egy listányi zenét random sorrendben.

Menü rendszer

A menüpontok közül számok megadásával lehet választani. A menük While ciklusokkal és Switch-Case-ekkel működnek. Minden menüpontnak és minden submenünek saját osztálya van. A Menük kiírását a MenuPrinter osztály végzi.

public class MainMenuController: Megkapja a DataManagert, majd tovább adja a választott submenünek.

public class MenuPrinter: A maüpontok kiírását végzi.

public class PlayMenuController: a lejátszással kapcsolatos submenüket irányítja, megkapja a DataManagert és tovább adja a választott submenünek.

public class PlayAllSubmenuController: az összes zene lejátszásával kapcsolatos menüpontokat irányítja, megkapja a DataManagert. Le tud játszani random sorrendben, legújabb legelöl, és legújabb leghátul sorrendben.

public class PlaylistSubmenuController: A playlistek lejátszásával kapcsolatos menüpontokat tartalmazza. Megkapja a DataManagert. Bekér egy Playlist nevet és lejátsza.

public class AlbumSubmenuController: Az albumok lejátszásával kapcsolatos menüpontokat tartalmazza. Megkapja a DataManagert. Bekér egy Playlist nevet és lejátsza.

public class public class DataEditorMenuController: Az adatmódosítással kapcsolatos menüpontokat tartalmazza, megkapja a DataManagert és továbbadja a választott menüpontnak, kilépés esetén visszatéríti.

public class SongDataEditorSubmenuController: Zenék adatainak módosításával kapcsolatos menüpontokat tartalmaz. Megkapja a DataManagert, kilépés esetén visszatéríti. Zene adat módosítás után újra generálja az album adatokat.

public class PlaylistDataEditorSubmenuController: Playlistek adatainak módosításával kapcsolatos menüpontokat tartalmaz. Megkapja a DataManagert, kilépés esetén visszatéríti. Van lehetőség új playlist létrehozására, és zene hozzáadására már meglévő playlisthez