مستندات پروژه پایانی درس برنامه‌نویسی پیشرفته

عنوان پروژه: MelodyStream (پلتفرم استریم و مدیریت محتوای صوتی)

۱. مقدمه و هدف

هدف از این پروژه، پیاده‌سازی هسته مرکزی (Backend) و رابط کاربری خط فرمان (CLI) برای یک سرویس پخش موسیقی مشابه Spotify یا SoundCloud است. در این پروژه شما باید با چالش‌های مدیریت داده‌ها در حافظه، ذخیره‌سازی فایل، روابط بین اشیاء و پیاده‌سازی منطق‌های پیچیده (Business Logic) دست و پنجه نرم کند.

---

۲. معماری سیستم (System Architecture)

پروژه باید از **معماری لایه‌بندی شده (Layered Architecture)** پیروی کند تا اصول "Separation of Concerns" رعایت شود. درهم‌آمیختن منطق و رابط کاربری باعث کسر نمره خواهد شد.

ساختار اجباری پکیج‌ها (هر کدام از موارد زیر یک پوشه هستند):

۱. **model**: شامل کلاس‌های موجودیت (Entity) مثل User, Song, Playlist. این کلاس‌ها فقط حاوی داده‌ها (Fields) و متدهای دسترسی (Getters/Setters) هستند و هیچ منطق پیچیده‌ای ندارند.

۲. **service**: مغز متفکر برنامه. تمام محاسبات، جستجوها، مدیریت صف پخش، احراز هویت و لاجیک‌های اصلی در این لایه قرار می‌گیرند (مثلاً PlayerService, AuthService).

۳. **repository** (یا data): مسئول خواندن و نوشتن داده‌ها در فایل (File Handler). لایه سرویس برای ذخیره اطلاعات فقط با این لایه صحبت می‌کند.

۴. **view** (یا cli): مسئول نمایش منوها و دریافت ورودی از کاربر. هیچ منطق شرطی (if/else) مربوط به بیزینس نباید در این لایه باشد، فقط Input/Output.

۵. **exception**: کلاس‌های خطای سفارشی که توسط دانشجو طراحی شده‌اند.

---

۳. تشریح موجودیت‌ها و شی‌گرایی (OOP Logic)

الف) سلسله مراتب محتوا (Polymorphism & Inheritance)

- یک کلاس انتزاعی (Abstract) یا اینترفیس به نام Audio باید وجود داشته باشد که فیلدهای مشترک مثل id, title, artist, duration, likes را در خود دارد.
- کلاس‌های Song و Podcast از Audio ارث‌بری می‌کنند.
  - کلاس Podcast فیلد اختصاصی episodeNumber و caption دارد.
  - کلاس Song فیلد اختصاصی lyrics و genre دارد.
- **چندریختی (Polymorphism):** متد ()play باید در کلاس‌های فرزند بازنویسی (Override) شود. مثلاً برای پادکست، آخرین دقیقه شنیده شده ذخیره شود اما برای موزیک فقط تعداد پلی (Play Count) اضافه شود.

ب) کاربران (Users)

- کلاس والد User (شامل username, password, balance, signupDate).
- دو نوع کاربر وجود دارد که از User ارث می‌برند:

۱. **Listener**: کاربر عادی که قابلیت شارژ حساب، ساخت پلی‌لیست و گوش دادن دارد.

۲. **Artist**: کاربری که علاوه بر قابلیت‌های Listener، قابلیت آپلود آهنگ و آلبوم و مشاهده درآمد را دارد.

- *نکته طراحی:* اگر شما از Composition استفاده کنید (یک User دارای یک نقش Role باشد) نمره مثبت دارد، اما ارث‌بری ساده (Artist extends User) نیز کاملاً قابل قبول است.

ج) کپسوله‌سازی (Encapsulation)

- تمام فیلدها باید private باشند.
- لیست آهنگ‌ها (ArrayList یا Map) نباید public باشد. افزودن آهنگ به دیتابیس باید از طریق متدی مثل addAudio(Audio audio) در کلاس مدیریت انجام شود تا اعتبارسنجی صورت گیرد.

---

۴. استراتژی استفاده از ساختمان داده‌ها (Collections Strategy)

این بخش قلب تپنده پروژه است. شما نباید صرفاً از ArrayList استفاده کنید. استفاده از موارد زیر **الزامی** است:

۱. **HashMap<String, Audio>**: دیتابیس اصلی موزیک‌ها در RAM.
   - *هدف:* جستجو بر اساس ID آهنگ باید با پیچیدگی زمانی O(۱) باشد.

۲. **LinkedList**: برای پیاده‌سازی **PlayQueue** (صف پخش یا همون پلی لیست).
   - *هدف:* کاربر ممکن است بخواهد آهنگی را به "وسط" صف اضافه کند یا از اول صف حذف کند. لینکد لیست برای Insert/Delete بهینه‌تر از آرایه‌هاست.

۳. **Stack**: برای پیاده‌سازی قابلیت **Previous Song**.
   - *هدف:* هر آهنگی که پخشش تمام می‌شود، وارد استک می‌شود. اگر کاربر دکمه Back را زد، آخرین آهنگ از استک پاپ (Pop) شده و پخش می‌شود.

۴. **HashSet**: برای لیست **FavoriteSongs**.

- *هدف:* جلوگیری از تکراری بودن. یک آهنگ نمی‌تواند دوبار لایک شود و Set این تضمین را می‌دهد.

۵. **PriorityQueue**: برای بخش **"Trending" (محبوب‌ترین‌ها)**.

- *هدف:* آهنگ‌ها باید بر اساس تعداد لایک یا تعداد پلی در این صف قرار بگیرند تا همیشه محبوب‌ترین آهنگ در بالای صف باشد ((access to max O(۱)).

---

۵. منطق‌های کلیدی و سناریوها (Business Logic)

سناریوی ۱: پخش کننده (The Player State Machine)

سیستم باید بداند الان چه آهنگی پخش می‌شود.

- قابلیت‌های: Play, Pause, Next, Previous.
- **Shuffle:** باید صف پخش (LinkedList) را به صورت تصادفی بهم بریزد (استفاده از Collections.shuffle).
- **Skip:** پرش به آهنگ بعدی. اگر صف خالی بود باید EmptyQueueException پرتاب شود.

سناریوی ۲: سیستم پیشنهاد دهنده (Recommendation Engine)

سیستم باید سلیقه کاربر را تحلیل کند.

- یک Map<Integer, Genre> برای هر کاربر وجود دارد.
- هر بار که کاربر آهنگی را لایک می‌کند، امتیاز آن ژانر (مثلاً "Rock") در مپ کاربر افزایش می‌یابد.
- در منوی "For You"، سیستم باید ۵ آهنگ از دیتابیس کل پیدا کند که ژانرشان با ژانر مورد علاقه کاربر یکی است و قبلاً آن‌ها را نشنیده است.

سناریوی ۳: جستجوی پیشرفته (Advanced Search)

جستجو نباید ساده باشد. کاربر باید بتواند فیلتر ترکیبی اعمال کند.

- ورودی کلمات کلیدی از کاربر گرفته شود.
- سیستم باید قابلیت فیلتر همزمان روی "نام خواننده" **AND** "ژانر" را داشته باشد. (مثلاً: تمام آهنگ‌های "PostRock" از "Da Voile").

سناریوی ۴: متن آهنگ هماهنگ (Synced Lyrics)

- هر آهنگ دارای یک متن (String) است که خطوط با \n جدا شده‌اند.
- هنگام انتخاب گزینه "Show Lyrics"، برنامه باید متن را خط به خط چاپ کند و بین هر خط یک وقفه زمانی (مثلاً ۲ ثانیه) با استفاده از Tread.sleep() ایجاد کند تا حس پخش زنده و کارائوکه منتقل شود.

سناریوی ۵: اقتصاد و آرتیست

- آرتیست‌ها می‌توانند آلبوم بسازند و آهنگ‌ها را درون آن قرار دهند.
- هر بار که یک آهنگ توسط کاربران پلی می‌شود، به حساب Artist مبلغ مشخصی (مثلاً ۰.۵ دلار) اضافه می‌شود.
- تراکنش‌های مالی باید لاگ شوند و در فایل ذخیره شوند.

---

۶. مدیریت خطا (Error Handling)

برنامه نباید تحت هیچ شرایطی کرش کند (Crash). ورودی‌های اشتباه کاربر باید مدیریت شوند. ساختن و استفاده از کلاس‌های زیر (Custom Exceptions) الزامی است:

- WrongPasswordException / UserNotFoundException: برای لاگین.
- InvalidAudioFormatException: اگر فرمت فایل آپلودی نامعتبر بود.
- EmptyQueueException: تلاش برای زدن دکمه Next وقتی صف خالی است.
- PermissionDeniedException: تلاش کاربر عادی برای دسترسی به پنل آرتیست.

---

۷. فایل هندلینگ (Data Persistence)

هنگام بسته شدن برنامه، تمام داده‌ها باید ذخیره شوند و هنگام باز شدن، بازیابی گردند.

۱. **Serialization:** استفاده از ObjectOutputStream برای ذخیره کردن کلیِ آبجکت‌های Library و UserList.

۲. **Text Logging:** یک فایل transactions.txt باید وجود داشته باشد که تمام رخدادهای مالی (شارژ حساب، واریز به آرتیست) را به صورت متنی و خوانا (CSV) ذخیره کند تا قابل پرینت گرفتن باشد. (CSV فرمت عجیبی نیست فقط با کاما (,) دیتا های توی هر خط رو از هم جدا می کنید)

- فرمت: Date, UserID, TransactionType, Amount

---

۸. رابط کاربری و نمونه صفحات (CLI Walkthrough)

رابط کاربری باید تمیز، خوانا و دارای کادربندی باشد. در زیر فلو (Flow) دقیق برنامه آورده شده است که شما باید آن را پیاده‌سازی کنید.

۱. صفحه ورود (Welcome Screen)

```
==============================================
      ♫  WELCOME TO MELODY STREAM  ♫
==============================================
1. Login
2. Register (Listener)
3. Register (Artist)
4. Exit
==============================================
> Select option: _
```

۲. داشبورد شنونده (Listener Home)

پس از لاگین موفق، این منو نمایش داده می‌شود. بخش Trending از PriorityQueue خوانده می‌شود.

```
==============================================
 User: Ali_Reza | Balance: $15.0
==============================================
 [ 🔥 TRENDING TOP 3 ]
 1. Skyfall - Adele (Plays: 500)
 2. Numb - Linkin Park (Plays: 450)
 3. Dream On - Aerosmith (Plays: 410)
==============================================
 [ MENU ]
 1. Search Music (Advanced)
 2. My Library (Playlists & Likes)
 3. Player Control (Open Player)
 4. Recommendations (For You)
 5. Account Settings
 6. Logout
==============================================
> Select option: _
```

۳. صفحه پخش (The Player Interface)

هنگامی که آهنگی پلی می‌شود، کاربر وارد این حالت می‌شود. (استفاده از Thread برای سینک لیریک در اینجا نمود پیدا می‌کند).

```
-------------------------------------------
 NOW PLAYING ♫
 Title:  Bohemian Rhapsody
 Artist: Queen
 Album:  A Night at the Opera
-------------------------------------------
 [======.........................] 1:25
-------------------------------------------
 Queue Next: We Will Rock You
 History (Prev): Love of My Life
-------------------------------------------
```

```
-------------------------------------------
 [COMMANDS]
 [P] Play/Pause    [N] Next Track    [B] Previous
 [S] Shuffle       [R] Repeat        [L] Like
 [K] Karaoke (Lyrics)                [E] Exit Player
-------------------------------------------
> Enter command: K

[LYRICS MODE - Press Ctrl+C to stop]
Is this the real life?
(waiting 2s...)
Is this just fantasy?
(waiting 2s...)
Caught in a landslide...
```

۴. پنل مدیریت آرتیست (Artist Panel)

```
================================================
 Artist Panel: Hans Zimmer | Earnings: $5,400
================================================
 1. Upload New Song
 2. Create New Album
 3. View Statistics (Total Plays per Song)
 4. Withdraw Money
 5. Switch to Listener Mode
================================================
> Select option: 1

Enter Song Title: Time
Enter Genre: Soundtrack
Enter Lyrics (End with 'END'): ...
```

۹. بارم‌بندی (مجموع ۱۰۰ نمره)

نمرات بر اساس کیفیت کد، رعایت اصول و اجرای صحیح سناریوها داده می‌شود.

بخش پروژه	جزئیات ارزیابی	نمره
معماری و OOP	رعایت لایه‌بندی، ارث‌بری صحیح، کپسوله‌سازی و عدم وجود منطق در View	**۲۰**
Collections Strategy	استفاده بجا و صحیح از Map, LinkedList, Stack, PriorityQueue, Set (هر مورد ۱ نمره)	**۲۵**
& Logic Scenarios	اجرای صحیح پلیر، سرچ پیشرفته، الگوریتم پیشنهاد دهنده، محاسبه درآمد و متن آهنگ	**۳۰**
File Handling	ذخیره و بازیابی بدون باگ (Serialization) + لاگ متنی تراکنش‌ها	**۱۵**
& CLI Exceptions	رابط کاربری تمیز (طبق داکیومنت)، مدیریت خطاهای ورودی و اکسپشن‌های اختصاصی	**۱۰**

---

۱۰. بخش نمرات اضافه (Bonus)

موارد زیر جزو پروژه اصلی **نیستند** و صرفاً برای دریافت نمره تشویقی یا جبران نمرات از دست رفته می‌باشند:

۱. **اتصال به دیتابیس (+۱۵ نمره):**

- حذف فایل هندلینگ و استفاده از **MySQL** یا **PostgreSQL**.

- استفاده از JDBC برای کوئری زدن.
- طراحی ERD صحیح و جداول نرمال‌سازی شده.

۲. **رابط گرافیکی GUI (+۱۵ نمره):**

- پیاده‌سازی برنامه با **JavaFX** یا **Swing**.
- برنامه باید دارای دکمه‌های گرافیکی Play/Pause، نوار لغزنده (Slider) برای موزیک و جداول برای نمایش لیست‌ها باشد.
- *نکته:* اگر GUI زده شود، نیازی به طراحی منوهای CLI نیست.
