DSCM031Semantic Technologies

Coursework 3: Program in Python

Student: Zornitsa Hristova Hristova
FN: 120168

I've chosen to implement the logic from my OWL file in Python since it has a **owlready2** library.

Tasks:

- 4 core functionalities which demonstrate the ontology basics
  - **`list_songs()`** - Lists all songs with details (artist, duration, rating, play count).

```
def list_songs():
    print("♪ Songs in ontology:")
    for song in Song.instances():
        artist = getattr(song, "hasArtist", None)
        artist_name = artist.name if artist else "Unknown"
        duration = getattr(song, "duration", "Unknown")
        rating = getattr(song, "rating", "Unknown")
        play_count = getattr(song, "playCount", "Unknown")
        print(f"- {song.name} | Artist: {artist_name} | Duration:
{duration}s | Rating: {rating} | Plays: {play_count}")
```

  - **`list_artists()`** - Lists all artists with their awards.

```
def list_artists():
    print("\n🎤 Artists in ontology:")
    for artist in Artist.instances():
        award = getattr(artist, "hasAward", ["None"])
        print(f"- {artist.name} | Award: {award}")
```

  - **`list_inferred_classes()`** - Lists inferred classes like PopularSong, ShortSong, AwardWinningSong.

```
def list_inferred_classes():
    print("\n📊 Inferred classes:")
    if PopularSong:
        print("Popular Songs:")
        for song in PopularSong.instances():
            print(f"  - {song.name}")
    if ShortSong:
        print("Short Songs:")
        for song in ShortSong.instances():
            print(f"  - {song.name}")
    if AwardWinningSong:
        print("Award Winning Songs:")
        for song in AwardWinningSong.instances():
            print(f"  - {song.name}")
```

- **`list_playlists()`** – Lists playlists and which songs are in them.

```
def list_playlists():
    print("\n🎧 Playlists and songs:")
    for playlist in Playlist.instances():
        songs_in_playlist = [s.name for s in Song.instances() if
playlist in getattr(s, "inPlaylist", [])]
        print(f"- {playlist.name}: {songs_in_playlist}")
```

- 3 additional functionalities
  - **`recommend_songs_for_preson()`** – Recommendation feature for each person which is based on the songs they like.

```
def recommend_songs_for_person(person_name_or_object):
    """
    Recommend songs for a person based on their likes.

    Args:
        person_name_or_object: Either a string (person name like "Me",
"Mariya")
                               or a Person object from the ontology
    """
    # Handle both string (name) and Person object inputs
    if isinstance(person_name_or_object, str):
        # Find the person by name
        person = None
        for p in Person.instances():
            if p.name == person_name_or_object:
                person = p
                break

        if person is None:
            print(f"❌ Error: Person '{person_name_or_object}' not
found!")
            return
        person_name = person_name_or_object
    else:
        # Assume it's a Person object
        person = person_name_or_object
        if person is None:
            print("Person not found!")
            return
        person_name = person.name

    liked_songs = getattr(person, "likes", [])
    recommendations = []
    for song in Song.instances():
        if song not in liked_songs:
            recommendations.append(song.name)
    print(f"\n🎶 Recommended songs for {person_name}:
```

- **`add_favorite_to_playlist()-`** For each user there is an option to add a favorite song to a playlist called “Name_of_UserFavorites”.

```
def add_favorite_to_playlist(person_name, song_name):

    with onto:
        # Find the person
        person = None
        for p in Person.instances():
            if p.name == person_name:
                person = p
                break

        if person is None:
            print(f"❌ Error: Person '{person_name}' not found!")
            return False

        # Find the song
        song = None
        for s in Song.instances():
            if s.name == song_name:
                song = s
                break

        if song is None:
            print(f"❌ Error: Song '{song_name}' not found!")
            return False

        # Create or find the favorites playlist name
        favorites_playlist_name = f"{person_name}Favorites"

        # Check if the favorites playlist already exists
        favorites_playlist = None
        for pl in Playlist.instances():
            if pl.name == favorites_playlist_name:
                favorites_playlist = pl
                break

        # Create the favorites playlist if it doesn't exist
        if favorites_playlist is None:
            favorites_playlist = Playlist(favorites_playlist_name)
            print(f"✅ Created new favorites playlist: {favorites_playlist_name}")
        else:
            print(f"📝 Using existing favorites playlist: {favorites_playlist_name}")

        # Check if the song is already in the playlist
        current_playlists = getattr(song, "inPlaylist", [])
        if favorites_playlist in current_playlists:
            print(f"ℹ️  Song '{song_name}' is already in {person_name}'s favorites playlist")
            return True

        # Add the song to the playlist
        current_playlists.append(favorites_playlist)
        song.inPlaylist = current_playlists

        # Also mark that the person likes this song (if not already)
        liked_songs = getattr(person, "likes", [])
        if song not in liked_songs:
            liked_songs.append(song)
            person.likes = liked_songs

        print(f"✅ Added '{song_name}' to {person_name}'s favorites playlist")
        return True
```

- **`show_user_favorites()` –** List the songs from the playlist "Name_of_UserFavorites"

```
def show_user_favorites(person_name):
    """
    Display all songs in a user's favorites playlist.

    Args:
        person_name: Name of the person (e.g., "Me", "Mariya", "Miya")
    """
    favorites_playlist_name = f"{person_name}Favorites"

    # Find the favorites playlist
    favorites_playlist = None
    for pl in Playlist.instances():
        if pl.name == favorites_playlist_name:
            favorites_playlist = pl
            break

    if favorites_playlist is None:
        print(f"😢 {person_name} doesn't have a favorites playlist yet")
        return

    # Get all songs in this playlist
    songs_in_favorites = [song for song in Song.instances()
                          if favorites_playlist in getattr(song, "inPlaylist", [])]

    if songs_in_favorites:
        song_names = [song.name for song in songs_in_favorites]
        print(f"\n★ {person_name}'s Favorites Playlist ({len(song_names)} songs):")
        for i, song_name in enumerate(song_names, 1):
            print(f"   {i}. {song_name}")
    else:
        print(f"😢 {person_name}'s favorites playlist is empty")
```