Projeto de Avaliação POO - Fase 2

Ana Paula Canuto da Silva

Nº 24178 – Regime Pós-laboral

Docentes: Luís G. Ferreira & Ernesto Casanova

Ano letivo 2024/2025

Licenciatura em Engenharia de Sistemas Informáticos

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

Identificação do Aluno

Ana Paula Canuto da Silva

Aluno número 24178, pós-laboral

Licenciatura em Engenharia de Sistemas Informáticos

RESUMO

O projeto POOProjectMusicAndPodcast é uma aplicação de console desenvolvida em C# com o objetivo de praticar os conceitos de Programação Orientada a Objetos (POO). O sistema permite gerenciar artistas, álbuns, músicas e podcasts, utilizando uma estrutura organizada baseada em classes que representam o domínio do problema. O projeto destaca o uso de herança, onde uma classe base Menu é utilizada para definir comportamentos genéricos e reutilizáveis, sendo estendida por diferentes menus específicos. Além disso, o uso de interfaces proporciona uma navegação modular e escalável, facilitando a adição de novas funcionalidades. O trabalho evidencia a aplicação prática de POO em um contexto real, explorando a criação de objetos, encapsulamento e polimorfismo.

1. Introdução

O projeto POOProjectMusicAndPodcast é uma aplicação de console em C#, que simula um sistema de gerenciamento de músicas, álbuns, podcasts e artistas. Utilizando conceitos de Programação Orientada a Objetos (POO), o projeto explora a criação de classes, herança e interfaces, além de proporcionar uma experiência de interação com o usuário por meio de um sistema de menus. O objetivo deste relatório é descrever a estrutura do código, explicar o funcionamento das classes, o uso da herança e a implementação das interfaces no programa.

2. Trabalho Desenvolvido

Estrutura Geral do Projeto

O projeto foi organizado em duas pastas principais:

1. **Models**: Contém as classes principais do domínio, como Artist, Album, Music, Podcast, e outras relacionadas.
2. **Menus**: Contém as classes responsáveis pelos diferentes menus de interação com o usuário, implementando o comportamento para navegação e execução das funcionalidades.

A seguir, detalharemos as principais classes e como elas se relacionam.

Classes do Projeto (Models)

1. Classe Artist

- **Descrição**: Representa um artista, armazenando uma lista de álbuns e uma lista de notas (ranking).
- **Propriedades**:
  - Name: Nome do artista.
  - Albuns: Lista de álbuns registrados para o artista.
  - Average: Calcula a média das notas atribuídas ao artista.
- **Métodos**:
  - AddAlbum(Album album): Adiciona um álbum à lista de álbuns.
  - AddScore(Ranking score): Adiciona uma nota à lista de avaliações.
  - ShowDiscograph(): Exibe a discografia do artista.

2. Classe Album

- **Descrição**: Representa um álbum contendo uma lista de músicas.
- **Propriedades**:
  - Name: Nome do álbum.
  - Musics: Lista de músicas que fazem parte do álbum.
  - AlbumTime: Duração total do álbum.
- **Métodos**:
  - AddMusic(Music music): Adiciona uma música ao álbum.
  - ShowMusicsAlbum(): Exibe todas as músicas do álbum.

3. Classe Music

- **Descrição**: Representa uma música, contendo informações como nome, artista, duração e disponibilidade.
- **Propriedades**:
  - Name: Nome da música.
  - Artist: Artista responsável pela música.
  - MusicTime: Duração da música em minutos.
  - Available: Indica se a música está disponível para reprodução.
- **Métodos**:
  - ExibirFichaTecnica(): Exibe os detalhes da música.

4. Classe Podcast

- **Descrição**: Representa um podcast, contendo uma lista de episódios.
- **Propriedades**:
  - Name: Nome do podcast.
  - Episodes: Lista de episódios do podcast.
  - PodcastTime: Duração total de todos os episódios.
- **Métodos**:
  - AdicionarMusica(Episode episode): Adiciona um episódio ao podcast.
  - ShowPodcastEpisodes(): Exibe todos os episódios do podcast.

5. Classe Episode

- **Descrição**: Representa um episódio de podcast, contendo informações como nome, artista convidado, duração e disponibilidade.
- **Propriedades**:
  - Name: Nome do episódio.
  - Artist: Artista convidado no episódio.
  - EpTime: Duração do episódio.
- **Métodos**:
  - ExibirFichaTecnica(): Exibe os detalhes do episódio.

6. Classe Ranking

- **Descrição**: Representa uma avaliação de um artista, armazenando a pontuação dada.
- **Propriedades**:
    - Score: Nota atribuída ao artista.
- **Métodos**:
    - Parse(string texto): Converte uma string de texto para um objeto de pontuação (Ranking).

Herança e Interface (Menus)

Herança

- A herança é utilizada para criar uma estrutura organizada e reutilizável nos menus do programa. A classe base Menu fornece um método genérico Execute(), que é sobreposto (override) nas classes derivadas.
- **Classe Menu**:
    - É a classe base que define o método Execute() e o método estático ShowOptionsTitle(), responsável por formatar e exibir o título das opções do menu.
    - Métodos como Execute() podem ser sobrepostos pelas subclasses específicas para implementar diferentes funcionalidades.
- **Subclasses do Menu**:
    - MenuAlbumProfile: Permite registrar um novo álbum para um artista.
    - MenuArtistProfile: Permite registrar um novo artista.
    - MenuArtistScore: Permite avaliar um artista com uma pontuação de 0 a 10.
    - MenuLogOff: Finaliza o programa.
    - MenuShowArtists: Exibe todos os artistas registrados.
    - MenuShowInformations: Exibe informações sobre um artista específico.

Interface com o Usuário (Programa.cs)

- O programa utiliza um sistema de menus interativo. Cada opção do menu é mapeada para uma classe específica, e estas classes derivam da classe base Menu.
- A navegação é controlada por meio de um **dicionário** que mapeia números de opções para instâncias de menus específicos. O método ShowMenuOptions() gerencia essa navegação:
    - O usuário escolhe uma opção, e o programa verifica no dicionário qual menu deve ser executado.

- Cada menu realiza uma operação específica, como registrar um artista, adicionar um álbum ou avaliar um artista.

Funcionamento do Programa

1. O programa inicia exibindo um logo e apresentando as opções de menu ao usuário.
2. As opções incluem registrar um artista, adicionar álbuns, listar artistas, atribuir notas e visualizar informações.
3. Ao escolher uma opção, o programa chama o método Execute() da classe correspondente.
4. Dependendo da escolha, o programa executa operações como adicionar artistas/álbuns, exibir listas e calcular notas médias.

O sistema continua até que o usuário escolha a opção de **LogOff**, que encerra a aplicação.

3. Conclusões

O projeto **POOProjectMusicAndPodcast** é um exemplo didático e eficaz de aplicação dos conceitos de **Programação Orientada a Objetos**. Utilizando uma abordagem organizada, o projeto implementa as funcionalidades principais para um sistema de gerenciamento de músicas e podcasts, como registro de artistas, adição de álbuns e avaliação de artistas.

A utilização de **herança** nos menus permite uma expansão facilitada e reutilização de código, enquanto o uso de **interfaces** por meio da classe base Menu proporciona uma estrutura modular, permitindo que novos menus sejam adicionados de forma simples.

A interface de usuário baseada em menus é clara e intuitiva, oferecendo uma boa experiência de navegação para o usuário. O projeto pode ser expandido para incluir funcionalidades adicionais, como criação de playlists, integração com serviços de streaming e suporte a múltiplos usuários, tornando-o uma base sólida para um aplicativo mais robusto.

Bibliografia

https://cursos.alura.com.br/formacao-linguagem-c#

Pessoas Instrutoras

Guilherme Lima

Daniel Portugal

Luri, inteligência artificial (IA) da Alura – Dúvidas e sujestões de como integrar a herança ao código.

https://cursos.alura.com.br