SISTEMA DE GERENCIAMENTO DE COLEÇÃO DE DISCOS UTILIZANDO PARADIGMA ORIENTADO A OBJETOS COM JAVA

Giovanni Braga Soares Vasconcelos[1], Antônio Heitor Gomes Azevedo[1], Cauã Maia de Souza Nara[1]

[1] Escola de Negócios, Tecnologia e Inovação - ARGO, Centro Universitário do Estado do Pará, Brasil, 5º Período de Ciência da Computação.

{giovanni23070008, antonio23070017, caua23070005}@aluno.cesupa.br

Paradigmas de Linguagens de Programação.

Profª. Suzana Lustosa de Souza

Belém, Pará. Maio de 2025.

RESUMO

Este artigo apresenta o desenvolvimento de um sistema de gerenciamento para coleção de discos utilizando o paradigma orientado a objetos, especificamente com a linguagem Java. O objetivo é demonstrar como o paradigma orientado a objetos pode ser aplicado para solucionar problemas reais de organização e gestão de informações, onde o foco está na modelagem de entidades do mundo real através de classes e objetos. O sistema desenvolvido permite gerenciar artistas, discos e coleções, possibilitando operações de adição, remoção e consulta de forma intuitiva e organizadas através de encapsulamento, herança e polimorfismo.

Palavras-chave: Paradigma Orientado a Objetos; Java; Sistema de Informação; Gerenciamento de Coleção de Discos.

ABSTRACT

This paper presents the development of a record collection management system using the object-oriented programming paradigm, specifically with the Java language. The goal is to demonstrate how the object-oriented paradigm can be applied to solve real-world information organization and management problems, where the focus is on modeling real-world entities through classes and objects. The developed system allows managing artists, records, and collections, enabling addition, removal, and query operations in an intuitive and organized manner through encapsulation, inheritance, and polymorphism.

Keywords: Object-Oriented Paradigm; Java; Information System; Record Collection Management.

1. INTRODUÇÃO

Na era digital atual, a organização e gestão de coleções pessoais tornaram-se uma necessidade crescente para muitos entusiastas da música. Colecionadores de discos, sejam eles de vinil, CD ou outros formatos, frequentemente enfrentam desafios para catalogar e gerenciar suas coleções de forma eficiente. Neste contexto, o presente artigo tem como objetivo desenvolver um sistema de gerenciamento para coleção de discos utilizando o paradigma orientado a objetos, especificamente com a linguagem Java.

Como destacado por Deitel e Deitel (2017), o paradigma orientado a objetos revolucionou a forma como pensamos sobre programação, permitindo que modelemos sistemas computacionais de maneira mais próxima ao mundo real. Este paradigma se baseia na criação de objetos que encapsulam dados e comportamentos, facilitando a manutenção, reutilização e extensibilidade do código.

Um sistema de gerenciamento para coleção de discos representa um caso ideal para aplicação do paradigma orientado a objetos, pois envolve entidades claras e bem definidas, como discos, artistas e coleções, cada uma com suas características e comportamentos específicos. Ao utilizar Java, podemos aproveitar recursos como encapsulamento, herança e polimorfismo para criar um sistema robusto e de fácil manutenção, como observado por Horstmann (2019).

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Paradigmas de Programação

Os paradigmas de programação representam diferentes filosofias e abordagens para estruturar e resolver problemas computacionais. Cada paradigma oferece uma perspectiva única sobre como organizar código e dados. Segundo Sebesta (2018), os principais paradigmas incluem:

Paradigma Imperativo: Baseia-se na execução sequencial de comandos que modificam o estado do programa. O programador especifica explicitamente os passos que o computador deve seguir. Linguagens como C e Pascal são exemplos clássicos deste paradigma.

Paradigma Funcional: Enfatiza o uso de funções matemáticas puras, evitando mudanças de estado e dados mutáveis. Linguagens como Haskell e Lisp seguem este paradigma.

Paradigma Orientado a Objetos: Organiza o código em torno de objetos que encapsulam dados e comportamentos. Este paradigma busca modelar sistemas computacionais de forma similar ao mundo real, utilizando conceitos como classes, objetos, herança e polimorfismo.

2.2 Paradigma Orientado a Objetos

O paradigma orientado a objetos surgiu da necessidade de criar sistemas mais organizados, reutilizáveis e de fácil manutenção. Booch et al. (2007) definem os pilares fundamentais deste paradigma:

Encapsulamento: Permite ocultar detalhes internos de implementação, expondo apenas interfaces necessárias para interação com o objeto. Isso promove segurança e modularidade no código.

Herança: Possibilita que classes filhas herdem características e comportamentos de classes pais, promovendo reutilização de código e criação de hierarquias lógicas.

Polimorfismo: Permite que objetos de diferentes classes sejam tratados de forma uniforme através de interfaces comuns, aumentando a flexibilidade do sistema.

Abstração: Facilita a modelagem de conceitos complexos do mundo real através de representações simplificadas em classes e objetos.

2.3 A Linguagem Java

Java, desenvolvida pela Sun Microsystems (atualmente Oracle), é uma linguagem que implementa completamente o paradigma orientado a objetos. Horstmann (2019) destaca as principais características que tornam Java adequada para desenvolvimento de sistemas orientados a objetos:

1. **Sintaxe clara e legível:** Java possui uma sintaxe próxima à linguagem natural, facilitando a leitura e compreensão do código.
2. **Garbage Collection automático:** O gerenciamento de memória é automatizado, reduzindo a possibilidade de vazamentos de memória.
3. **Portabilidade:** O conceito "write once, run anywhere" permite que programas Java executem em diferentes plataformas sem modificações.
4. **Robustez:** Java inclui verificações rigorosas de tipos e tratamento de exceções, contribuindo para a criação de software mais confiável.
5. **Bibliotecas extensas:** A linguagem oferece um vasto conjunto de bibliotecas padrão que aceleram o desenvolvimento.

2.4 Sistemas de Gerenciamento de Coleções

Um sistema de gerenciamento de coleções é uma aplicação projetada para organizar, catalogar e facilitar o acesso a conjuntos de itens relacionados. No contexto de coleções de discos, estes sistemas tipicamente incluem funcionalidades como:

1. **Catalogação:** Registro detalhado de discos com informações sobre artista, título, ano de lançamento e formato.
2. **Organização:** Estruturação lógica da coleção para facilitar busca e navegação.

3. **Estatísticas:** Geração de informações agregadas sobre a coleção, como valor total e distribuição por artista.
4. **Manutenção:** Operações de adição, remoção e atualização de itens da coleção.

3. METODOLOGIA

O desenvolvimento do sistema de gerenciamento para coleção de discos seguiu uma abordagem sistemática baseada nos princípios do paradigma orientado a objetos. As etapas da metodologia incluíram:

1. **Análise de requisitos:** Identificação das funcionalidades essenciais para um sistema de coleção de discos, incluindo gestão de artistas, discos e operações de coleção.
2. **Modelagem orientada a objetos:** Seguindo as recomendações de Booch et al. (2007), foi realizada a identificação das classes principais, seus atributos e métodos, estabelecendo relacionamentos apropriados entre as entidades.
3. **Implementação das classes:** Desenvolvimento das classes em Java, aplicando os princípios de encapsulamento e definindo interfaces claras para interação entre objetos.
4. **Desenvolvimento da lógica de negócio:** Implementação dos métodos que realizam as operações principais do sistema, como adição, remoção e consulta de discos.
5. **Testes e validação:** Verificação da corretude das funcionalidades implementadas através de casos de teste na classe principal.

Para implementação, foi escolhida a linguagem Java em sua versão mais recente, aproveitando suas características de orientação a objetos e robustez, conforme recomendado por Deitel e Deitel (2017).

4. PARADIGMA ESCOLHIDO

4.1 Justificativa da Escolha do Paradigma Orientado a Objetos

O paradigma orientado a objetos foi escolhido para este trabalho devido às seguintes razões, alinhadas com as observações de Horstmann (2019):

1. **Modelagem natural do domínio:** O sistema de coleção de discos envolve entidades claras (discos, artistas, coleções) que podem ser facilmente representadas como classes e objetos, tornando o código mais intuitivo e próximo ao problema real.
2. **Reutilização de código:** A orientação a objetos promove a criação de componentes reutilizáveis, permitindo que classes sejam utilizadas em diferentes contextos sem modificações.
3. **Manutenibilidade:** O encapsulamento permite modificar implementações internas sem afetar outras partes do sistema, facilitando evoluções e correções.
4. **Extensibilidade:** A estrutura orientada a objetos facilita a adição de novas funcionalidades e tipos de entidades sem comprometer o código existente.

5. **Organização lógica:** A separação clara de responsabilidades entre classes resulta em código mais organizado e compreensível.

4.2 Critérios de Linguagem

A escolha de Java como linguagem para implementação do sistema foi baseada nos seguintes critérios, conforme discutido por Sebesta (2018):

1. **Legibilidade:** Java possui sintaxe clara e expressiva, com convenções de nomenclatura que facilitam a compreensão do código.
2. **Facilidade de escrita:** A linguagem oferece construções intuitivas para implementar conceitos orientados a objetos, reduzindo a complexidade de desenvolvimento.
3. **Confiabilidade:** O sistema de tipos forte e o tratamento obrigatório de exceções contribuem para a criação de software mais robusto.
4. **Eficiência:** Embora interpretada, Java oferece desempenho adequado para aplicações de gerenciamento de dados através da máquina virtual otimizada (JVM).
5. **Portabilidade:** Programas Java podem executar em diferentes sistemas operacionais sem modificações, aumentando a versatilidade da solução.
6. **Suporte a orientação a objetos:** Java foi projetada especificamente para suportar completamente o paradigma orientado a objetos, oferecendo todos os recursos necessários.
7. **Comunidade e recursos:** Java possui vasta comunidade de desenvolvedores e abundante documentação, facilitando o desenvolvimento e resolução de problemas.

5. IMPLEMENTAÇÃO DO SISTEMA

O sistema de gerenciamento para coleção de discos foi implementado utilizando Java para criar um conjunto de classes que representam as entidades do domínio e suas interações. A implementação seguiu as melhores práticas descritas por Deitel e Deitel (2017) e consistiu em uma abordagem estruturada que priorizou o encapsulamento adequado, definição clara de responsabilidades e implementação de operações eficientes.

5.1 Arquitetura do Sistema

O sistema foi estruturado em quatro classes principais, cada uma com responsabilidades bem definidas:

Classe Artist: Representa artistas musicais, encapsulando informações como nome, gênero, país de origem e ano de formação. Esta classe demonstra o princípio de encapsulamento ao manter dados privados e expor apenas métodos necessários para interação.

Classe Record: Modela discos individuais, incluindo título, artista associado, ano de lançamento, formato e preço. A classe implementa um sistema de identificação automática através de incremento sequencial, demonstrando encapsulamento de lógica de negócio.

Classe RecordCollection: Gerencia coleções de discos, implementando operações como adição, remoção, busca e geração de estatísticas. Esta classe exemplifica o princípio de composição, mantendo uma lista de objetos Record.

Classe Main: Serve como ponto de entrada da aplicação, demonstrando o uso das demais classes e testando as funcionalidades implementadas.

5.2 Implementação de Funcionalidades

As principais funcionalidades implementadas incluem:

1. **Gestão de discos:** Adição e remoção de discos da coleção, com validação automática e feedback ao usuário.
2. **Sistema de busca:** Implementação de métodos de busca por título, artista e formato, utilizando algoritmos de busca linear adequados para o tamanho esperado da coleção.
3. **Geração de estatísticas:** Cálculo automático de métricas como total de discos, valor total e médio da coleção, demonstrando operações de agregação em coleções.
4. **Visualização de dados:** Métodos para exibição formatada da coleção e estatísticas, implementando interfaces amigáveis ao usuário.

O sistema utiliza estruturas de dados apropriadas, como ArrayList, para armazenar coleções de objetos, aproveitando as vantagens da biblioteca padrão Java para operações eficientes de manipulação de listas.

5.3 Código Fonte e Disponibilidade

O código completo do sistema está organizado em classes distintas, seguindo convenções de nomenclatura Java. A implementação inclui:

Encapsulamento adequado: Todos os atributos são declarados como privados, com acesso controlado através de métodos getter e setter quando apropriado.

Construtores parametrizados: Cada classe possui construtores que inicializam objetos em estados válidos, garantindo integridade dos dados.

Métodos de utilidade: Implementação de métodos toString() para representação textual dos objetos, facilitando depuração e exibição.

Tratamento de casos especiais: Validação de operações como remoção de discos inexistentes, com mensagens informativas ao usuário.

6. RESULTADOS E DISCUSSÃO

A implementação do sistema de gerenciamento para coleção de discos utilizando o paradigma orientado a objetos com Java demonstrou resultados significativos, que confirmam as vantagens deste paradigma destacadas por Booch et al. (2007) e Horstmann (2019).

6.1 Análise da Implementação

A clareza e organização do código desenvolvido evidenciam a adequação do paradigma orientado a objetos para o problema abordado. A separação de responsabilidades entre classes resultou em um sistema modular onde cada componente tem um propósito bem definido. Por exemplo, a classe RecordCollection encapsula toda a lógica de gerenciamento da coleção, enquanto a classe Record se concentra apenas na representação de discos individuais.

O uso de encapsulamento permitiu criar interfaces limpas entre os componentes, onde mudanças internas em uma classe não afetam outras partes do sistema. Isso ficou evidente na implementação do sistema de identificação automática da classe Record, que funciona de forma transparente para as demais classes.

6.2 Comparação com Outras Abordagens

Comparando com uma possível implementação utilizando paradigma procedural, a solução orientada a objetos apresentou vantagens significativas:

1. **Organização do código:** A estruturação em classes resulta em código mais organizado e de fácil navegação, facilitando manutenção e extensões futuras.
2. **Reutilização:** As classes desenvolvidas podem ser facilmente reutilizadas em outros contextos ou projetos similares, reduzindo esforço de desenvolvimento.
3. **Manutenibilidade:** Mudanças em uma classe específica não impactam outras partes do sistema, desde que as interfaces sejam mantidas, como observado por Deitel e Deitel (2017).
4. **Extensibilidade:** A adição de novas funcionalidades, como diferentes tipos de mídia ou novos critérios de busca, pode ser realizada através de herença ou composição sem modificar código existente.

6.3 Desafios Encontrados

Apesar das vantagens, alguns desafios foram encontrados durante a implementação:

1. **Complexidade inicial:** O paradigma orientado a objetos requer um investimento inicial maior em planejamento e design das classes, especialmente para desenvolvedores menos experientes.

2. **Overhead de memória:** A criação de múltiplos objetos pode resultar em maior uso de memória comparado a implementações procedurais mais diretas, embora este impacto seja mínimo para aplicações de pequeno porte.
3. **Curva de aprendizado:** Para desenvolvedores não familiarizados com orientação a objetos, pode haver uma curva de aprendizado para compreender completamente conceitos como encapsulamento e composição.

7. CONCLUSÕES

A implementação de um sistema de gerenciamento para coleção de discos utilizando o paradigma orientado a objetos com Java demonstrou a eficácia desta abordagem para o desenvolvimento de sistemas de informação, confirmando as observações teóricas de Booch et al. (2007) e Horstmann (2019). As principais conclusões obtidas são:

1. **Adequação do paradigma:** O paradigma orientado a objetos mostrou-se altamente adequado para o domínio do problema, permitindo modelar entidades do mundo real de forma natural e intuitiva através de classes e objetos.
2. **Clareza e organização:** A estruturação do código em classes bem definidas resultou em um sistema mais organizado e de fácil compreensão, facilitando tanto o desenvolvimento quanto a manutenção futura.
3. **Reutilização e extensibilidade:** A implementação orientada a objetos criou componentes reutilizáveis que podem ser facilmente adaptados para outros contextos ou estendidos com novas funcionalidades.
4. **Encapsulamento efetivo:** O uso adequado de encapsulamento resultou em interfaces limpas entre componentes, permitindo modificações internas sem impactar outras partes do sistema.
5. **Robustez do Java:** A linguagem Java provou ser uma escolha adequada, oferecendo recursos robustos para implementação de conceitos orientados a objetos, além de gerenciamento automático de memória e tratamento de exceções.

O sistema desenvolvido, embora simples, demonstra o potencial do paradigma orientado a objetos para aplicações de gerenciamento de dados. Em trabalhos futuros, o sistema poderia ser expandido com funcionalidades adicionais, como diferentes tipos de mídia, sistema de empréstimos, integração com APIs de informações musicais, e implementação de interfaces gráficas mais sofisticadas.

A experiência de desenvolvimento confirmou que a escolha adequada do paradigma de programação é fundamental para o sucesso de projetos de software, e que o paradigma orientado a objetos oferece uma base sólida para sistemas que modelam entidades do mundo real com suas características e comportamentos complexos.

REFERÊNCIAS

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: Guia do Usuário**. 2ª ed. Rio de Janeiro: Elsevier, 2007.

DEITEL, P.; DEITEL, H. **Java: Como Programar**. 10ª ed. São Paulo: Pearson, 2017.

HORSTMANN, C. S. **Core Java Volume I - Fundamentals**. 11ª ed. Upper Saddle River: Pearson, 2019.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 11ª ed. Porto Alegre: Bookman, 2018.