Guia passo a passo — Biblioteca de CDs em Java

Este guia ensina, do zero, a construir a biblioteca de CDs em Java com as classes **Artista**, **CD**, **Musica**, **BibliotecaCD** e uma classe **Main** para testes. É ideal para estudar orientação a objetos, listas, composição/associação entre classes e boas práticas básicas.

---

1) Visão geral do que vamos construir

Objetivo: cadastrar CDs herdados pelo Adriano, com artista, título, ano e lista de músicas (nome, duração e link do YouTube).

Classes e responsabilidades: - **Musica**: representa uma faixa (nome, duração, link do YouTube). - **Artista**: representa cantor/dupla/banda e mantém sua discografia (lista de CDs). - **CD**: representa um álbum (título, ano, artista e faixas). - **BibliotecaCD**: "repositório" de CDs; permite adicionar, listar e buscar CDs. - **Main**: ponto de entrada para montar um cenário de teste e executar as operações.

Relacionamentos (UML em texto):

```
Artista 1 --- * CD 1 --- * Musica
```

- Um **Artista** tem **muitos CDs**. - Um **CD** tem **muitas Músicas**. - Um **CD** pertence a **um Artista**.

---

2) Pré-requisitos e ambiente

- **JDK** (Java Development Kit) instalado (Java 8+).
- Um **editor/IDE** (IntelliJ IDEA, VS Code com extensão Java, Eclipse) **ou** apenas um terminal.

Estrutura de pastas sugerida (sem `package`)

```
projeto-biblioteca/
├── Artista.java
├── CD.java
├── Musica.java
├── BibliotecaCD.java
└── Main.java
```

Se quiser usar `package`, veja a seção **8. Boas práticas e extensões**.

---

3) Criando a classe `Musica`

Objetivo: encapsular os dados de uma faixa.

Atributos mínimos: - `nome: String` - `duracao: String` (ex.: "3:45"). *Dica:* você pode evoluir para `int segundos` depois. - `linkYoutube: String` (pode ser vazio/nulo se não tiver).

Passos: 1. Crie o arquivo **Musica.java**. 2. Declare a classe e os atributos como `private`. 3. Crie um **construtor** para preencher os campos. 4. Forneça ao menos um **getter** para `nome` (usaremos na busca de músicas). 5. Implemente um método utilitário para exibir/formatar os dados (ex.: `exibirInfo()` ou `toString()`).

Esqueleto (simplificado):

```
public class Musica {
    private String nome;
    private String duracao;
    private String linkYoutube;

    public Musica(String nome, String duracao, String linkYoutube) {
        this.nome = nome;
        this.duracao = duracao;
        this.linkYoutube = linkYoutube;
    }

    public String getNome() { return nome; }

    public String exibirInfo() {
        return "Música: " + nome + " | Duração: " + duracao +
               (linkYoutube != null && !linkYoutube.isEmpty() ? " | YouTube:
" + linkYoutube : "");
    }
}
```

---

4) Criando a classe `Artista`

Objetivo: representar cantor/dupla/banda e manter seus CDs.

Atributos mínimos: - `nome: String` - `tipo: String` (ex.: "Cantor", "Dupla", "Banda"). *Dica:* pode virar um `enum` futuramente. - `cds: List<CD>` (lista de álbuns deste artista).

Passos: 1. Crie o arquivo **Artista.java**. 2. Inicialize a lista `cds` no construtor (`new ArrayList<>()`). 3. Crie um método `adicionarCD(CD cd)` para manter a relação. 4. Crie métodos utilitários: `listarCDs()` e `toString()`.

Esqueleto:

```
import java.util.ArrayList;
import java.util.List;
```

```java
public class Artista {
    private String nome;
    private String tipo; // Cantor, Dupla ou Banda
    private List<CD> cds;

    public Artista(String nome, String tipo) {
        this.nome = nome;
        this.tipo = tipo;
        this.cds = new ArrayList<>();
    }

    public void adicionarCD(CD cd) { cds.add(cd); }

    public void listarCDs() {
        System.out.println("CDs do artista " + nome + ":");
        for (CD cd : cds) {
            System.out.println(" - " + cd.getTitulo());
        }
    }

    public String getNome() { return nome; }

    @Override
    public String toString() {
        return "Artista: " + nome + " | Tipo: " + tipo;
    }
}
```

Observação importante (associação bidirecional controlada): - Vamos chamar `artista.adicionarCD(this)` dentro do **construtor de** `CD` para manter a lista no artista automaticamente quando um CD novo for criado.

---

5) Criando a classe `CD`

Objetivo: representar o álbum e suas músicas.

Atributos mínimos: - `titulo: String` - `ano: int` - `artista: Artista` - `musicas: List<Musica>`

Passos: 1. Crie o arquivo **CD.java**. 2. No construtor, receba `titulo`, `ano` e `artista`. 3. Inicialize `musicas` com `new ArrayList<>()`. 4. **Mantenha a associação:** após setar o `artista`, chame `artista.adicionarCD(this)`. 5. Implemente métodos: `adicionarMusica`, `listarMusicas`, `buscarMusica(String nome)`. 6. Implemente `getTitulo()` e `getArtista()` para permitir buscas e impressões.

Esqueleto:

```java
import java.util.ArrayList;
import java.util.List;

public class CD {
    private String titulo;
    private int ano;
    private Artista artista;
    private List<Musica> musicas;

    public CD(String titulo, int ano, Artista artista) {
        this.titulo = titulo;
        this.ano = ano;
        this.artista = artista;
        this.musicas = new ArrayList<>();
        artista.adicionarCD(this); // mantém a lista do artista atualizada
    }

    public void adicionarMusica(Musica musica) { musicas.add(musica); }

    public void listarMusicas() {
        System.out.println("Músicas do CD \"" + titulo + "\" do artista " +
artista.getNome() + ":");
        for (Musica m : musicas) {
            System.out.println("  - " + m.exibirInfo());
        }
    }

    public Musica buscarMusica(String nome) {
        for (Musica m : musicas) {
            if (m.getNome().equalsIgnoreCase(nome)) {
                return m;
            }
        }
        return null;
    }

    public String getTitulo() { return titulo; }
    public Artista getArtista() { return artista; }

    @Override
    public String toString() {
        return "CD: " + titulo + " | Ano: " + ano + " | Artista: " +
artista.getNome();
    }
}
```

Boas práticas opcionais: - **Validação** no construtor (ex.: `ano > 0`, `titulo` não vazio). - **Imutabilidade parcial:** evitar `setters` públicos; expor só o que precisa. - **Proteção da lista:** se criar um `getMusicas()`, retorne uma cópia ou `Collections.unmodifiableList(musicas)`.

6) Criando a classe `BibliotecaCD`

Objetivo: ser um repositório simples para manipular a coleção de CDs.

Atributos: - `cds: List<CD>`

Passos: 1. Crie o arquivo **BibliotecaCD.java**. 2. Inicie `cds` no construtor. 3. Implemente `adicionarCD`, `listarCDs` e `buscarCD(String titulo)`.

Esqueleto:

```
import java.util.ArrayList;
import java.util.List;

public class BibliotecaCD {
    private List<CD> cds;

    public BibliotecaCD() {
        this.cds = new ArrayList<>();
    }

    public void adicionarCD(CD cd) { cds.add(cd); }

    public void listarCDs() {
        System.out.println("Lista de CDs cadastrados:");
        for (CD cd : cds) {
            System.out.println(" - " + cd);
        }
    }

    public CD buscarCD(String titulo) {
        for (CD cd : cds) {
            if (cd.getTitulo().equalsIgnoreCase(titulo)) {
                return cd;
            }
        }
        return null;
    }
}
```

Ideias de evolução: - Buscar por **artista** (`List<CD> buscarPorArtista(String nome)`). - Evitar **duplicatas** (verificar antes de adicionar). - Usar **Streams** (`cds.stream().filter(...)`).

---

7) Criando a classe `Main` (teste de uso)

Objetivo: montar um cenário, popular a biblioteca e exercitar as operações.

Passos: 1. Crie o arquivo **Main.java** com o método `public static void main(String[] args)`. 2. Instancie `BibliotecaCD`. 3. Crie alguns `Artista`s. 4. Crie CDs para esses artistas. 5. Adicione músicas aos CDs. 6. Adicione CDs à biblioteca. 7. Liste e busque para validar.

Exemplo (completo e funcional):

```
public class Main {
    public static void main(String[] args) {
        BibliotecaCD biblioteca = new BibliotecaCD();

        Artista queen = new Artista("Queen", "Banda");
        Artista beatles = new Artista("The Beatles", "Banda");

        CD cd1 = new CD("Greatest Hits", 1981, queen);
        cd1.adicionarMusica(new Musica("Bohemian Rhapsody", "5:55", "https://
youtu.be/fJ9rUzIMcZQ"));
        cd1.adicionarMusica(new Musica("Another One Bites The Dust", "3:35",
"https://youtu.be/rYOWxgSXdEE"));

        CD cd2 = new CD("Abbey Road", 1969, beatles);
        cd2.adicionarMusica(new Musica("Come Together", "4:20", "https://
youtu.be/45cYwDMibGo"));
        cd2.adicionarMusica(new Musica("Something", "3:03", "https://
youtu.be/UelDrZ1aFeY"));

        biblioteca.adicionarCD(cd1);
        biblioteca.adicionarCD(cd2);

        // Listagem geral
        biblioteca.listarCDs();

        // Listagem por artista
        System.out.println("\n>>> CDs do artista Queen:");
        queen.listarCDs();

        // Busca de CD e suas faixas
        System.out.println("\n>>> Músicas do CD Abbey Road:");
        CD buscado = biblioteca.buscarCD("Abbey Road");
        if (buscado != null) {
            buscado.listarMusicas();
        }
    }
}
```

---

8) Boas práticas e extensões (para ir além)

- **Packages (organização):**
- Estruture o projeto em pacotes, por exemplo:

```
src/
└── br/com/seunome/biblioteca/
    ├── model/
    │   ├── Artista.java
    │   ├── CD.java
    │   └── Musica.java
    ├── repo/
    │   └── BibliotecaCD.java
    └── app/
        └── Main.java
```

- Dentro de cada arquivo, declare o `package` correspondente (ex.: `package br.com.seunome.biblioteca.model;`).

- **Enum para tipo de artista:**

```
public enum TipoArtista { CANTOR, DUPLA, BANDA }
```

Substitua `String tipo` por `TipoArtista tipo`.

- **Validação e exceções:**

- Verifique campos obrigatórios no construtor e lance `IllegalArgumentException` quando inválidos.

- **Comparação e coleções:**

- Implemente `equals` / `hashCode` em `CD` para evitar duplicatas (ex.: por `titulo + artista + ano`).

- **Duração como número:**

- Armazene a duração em **segundos (`int`)** e gere a string `mm:ss` sob demanda. Alternativa: `java.time.Duration` (Java 8+).

- **Persistência simples:**

- Salvar/carregar em arquivo texto/CSV/JSON.

- **Testes unitários (JUnit):**

- Crie testes para `buscarCD`, `buscarMusica`, validações e fluxos de erro.

- **Interface (CLI/GUI):**

- Evolua para um menu no console (Scanner) ou para uma GUI (JavaFX/Swing).

---

9) Como compilar e executar

Opção A — via terminal (sem packages)

No diretório do projeto, rode:

```
javac *.java
java Main
```

Opção B — via terminal (com packages e pasta `src`)

```
# dentro da pasta raiz (que contém src/)
javac -d out $(find src -name "*.java")
java -cp out br.com.seunome.biblioteca.app.Main
```

> No Windows (PowerShell), use: `Get-ChildItem -Recurse src -Filter *.java | ForEach-Object { $_.FullName } | javac -d out @-`

Opção C — via IDE

- **IntelliJ IDEA**: New Project > Add Java files > Botão run no `Main`.
- **VS Code**: instale **Extension Pack for Java**, abra a pasta do projeto, crie os arquivos e clique em "Run" no `main`.

---

10) UML (opcional com PlantUML)

Cole em um arquivo `.puml` e renderize num plugin/online:

```
@startuml
class Musica {
  - nome: String
  - duracao: String
  - linkYoutube: String
  + Musica(nome, duracao, linkYoutube)
  + getNome(): String
  + exibirInfo(): String
}

class Artista {
  - nome: String
  - tipo: String
  - cds: List<CD>
  + Artista(nome, tipo)
  + adicionarCD(cd: CD)
  + listarCDs()
```

```
  + getNome(): String
}

class CD {
  - titulo: String
  - ano: int
  - artista: Artista
  - musicas: List<Musica>
  + CD(titulo, ano, artista)
  + adicionarMusica(musica: Musica)
  + listarMusicas()
  + buscarMusica(nome: String): Musica
  + getTitulo(): String
  + getArtista(): Artista
}

class BibliotecaCD {
  - cds: List<CD>
  + BibliotecaCD()
  + adicionarCD(cd: CD)
  + listarCDs()
  + buscarCD(titulo: String): CD
}

Artista "1" -- "*" CD
CD "1" -- "*" Musica
@enduml
```

---

11) Checklist rápido de estudo

- [ ] Entendi o **papel** de cada classe.
- [ ] Sei **como e por que** usar `List<>` (ArrayList) para guardar coleções.
- [ ] Consigo **manter as associações** (CD atualiza o Artista ao ser criado).
- [ ] Sei **compilar e executar** pelo terminal ou IDE.
- [ ] Consigo **evoluir** o projeto (enum, validações, buscar por artista, etc.).

Pronto! Com esses passos, você cria e entende a biblioteca de CDs, com base sólida para evoluir o projeto e treinar os principais conceitos de orientação a objetos em Java.