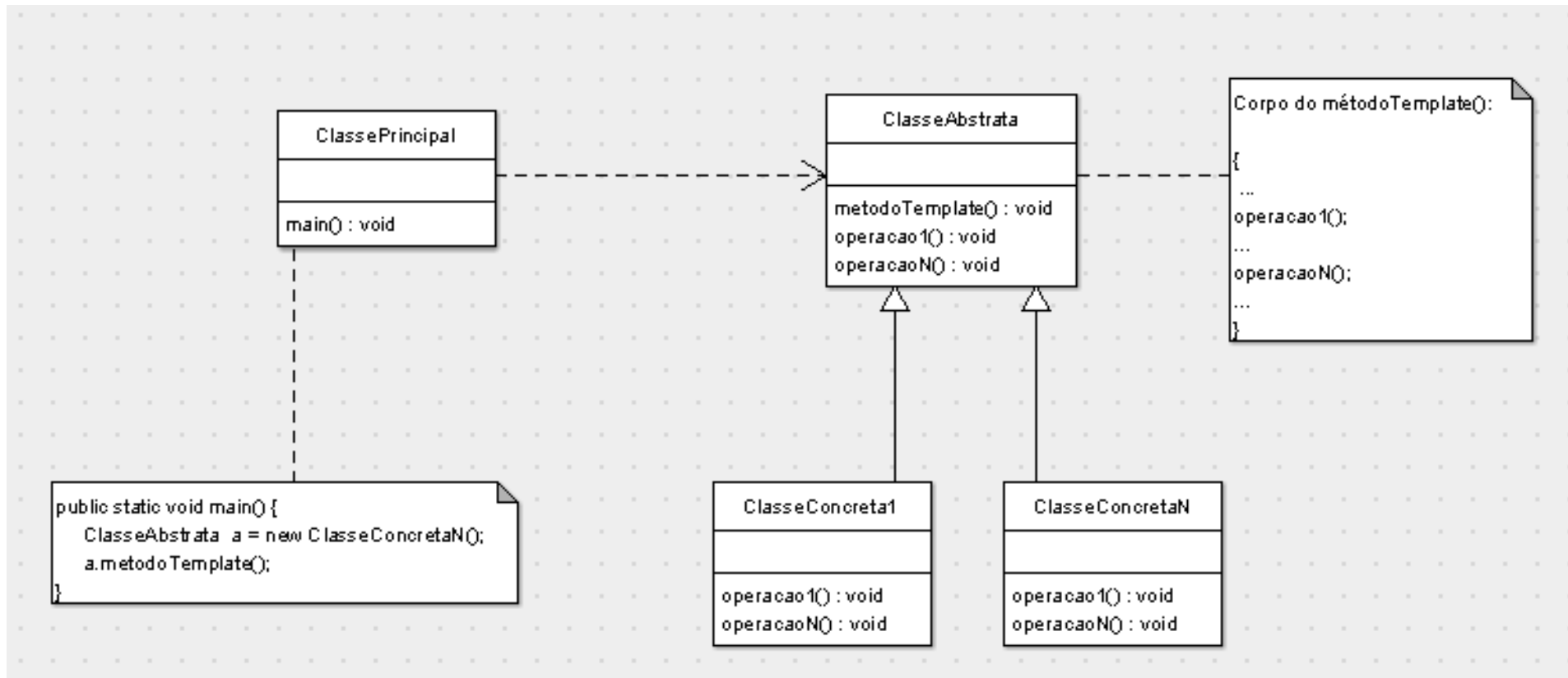


# Template Method

# Elementos Essenciais

- **Nome:** *Template Method*
- **Problema:** Flexibilizar partes de um algoritmo mais geral em comportamentos definidos posteriormente.
- **Solução:** Definir o esqueleto de um algoritmo em uma operação, postergando alguns passos para as subclasses. Template Method permite que subclasses redefinam certos passos de um algoritmo sem mudar a estrutura do mesmo.
- **Consequências:** Flexibilidade na redefinição de partes do comportamento de um método. Necessidade do cliente do código conhecer que métodos substituir para conseguir o efeito desejado. Uma redefinição de um método errado (que não deveria ser redefinido) pode gerar muitos problemas.

# Template Method



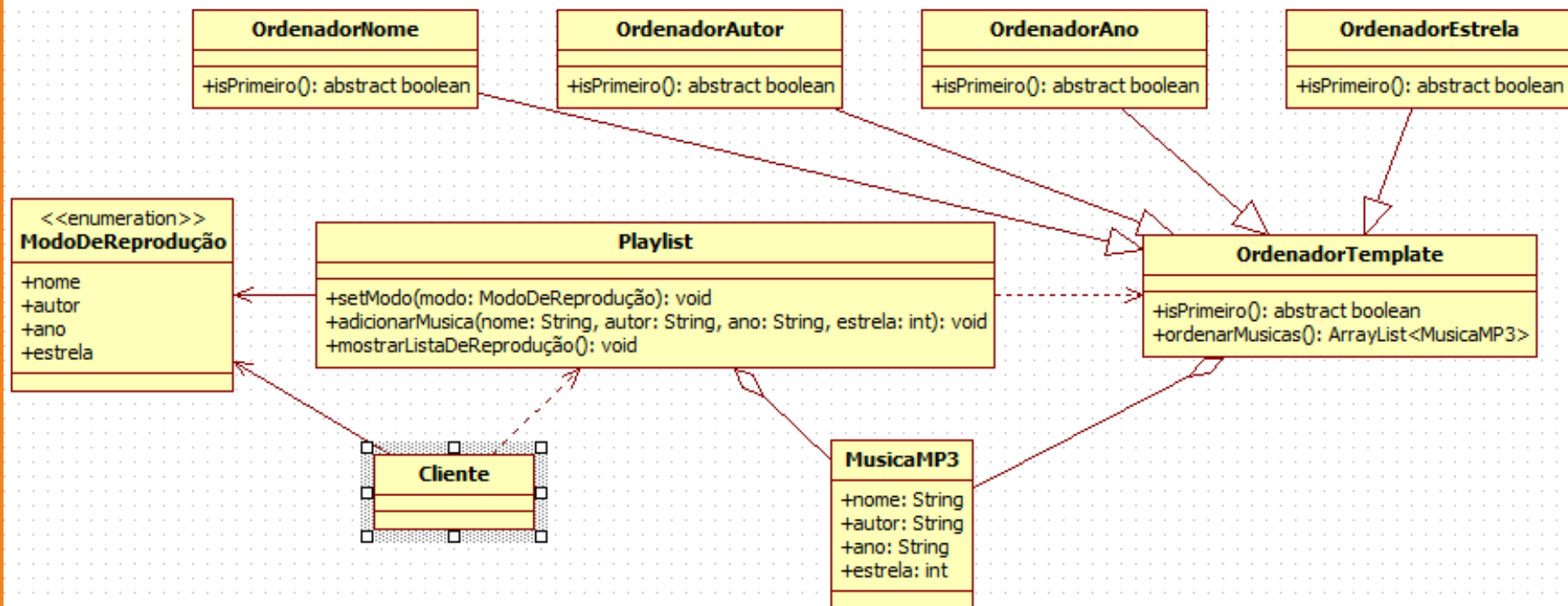
# Problema

- Suponha um player de música que oferece várias maneiras de reproduzir as músicas de uma playlist. Para exemplificar suponha que podemos reproduzir a lista de músicas da seguinte maneira:
  - Ordenado por nome da música
  - Ordenado por nome do Autor
  - Ordenado por ano
  - Ordenado por estrela (preferência do usuário)

Fonte: <https://brizenowordpress.com/category/padroes-de-projeto/template-method/> (acessado em 02/03/2020)

# Diagrama de classes

```
public abstract class OrdenadorTemplate {
    public abstract boolean isPrimeiro(MusicaMP3 musical, MusicaMP3 musica2);
    protected ArrayList<MusicaMP3> ordenarMusica(ArrayList<MusicaMP3> lista) {
        ArrayList<MusicaMP3> novaLista = new ArrayList<MusicaMP3>();
        for (MusicaMP3 musicaMP3 : lista) {
            novaLista.add(musicaMP3);
        }
        for (int i = 0; i < novaLista.size(); i++) {
            for (int j = i; j < novaLista.size(); j++) {
                if (!isPrimeiro(novaLista.get(i), novaLista.get(j))) {
                    MusicaMP3 temp = novaLista.get(j);
                    novaLista.set(j, novaLista.get(i));
                    novaLista.set(i, temp);
                }
            }
        }
        return novaLista;
    }
}
```



# Código do main

```
public class TemplateMethod {  
    public static void main(String[] args) {  
  
        Playlist minhaPlaylist = new Playlist(ModoDeReproducao.porNome);  
        minhaPlaylist.adicionarMusica("Everlong", "Foo Fighters", "1997", 5);  
        minhaPlaylist.adicionarMusica("Song 2", "Blur", "1997", 4);  
        minhaPlaylist.adicionarMusica("American Jesus", "Bad Religion", "1993",  
            3);  
        minhaPlaylist.adicionarMusica("No Cigar", "Milencollin", "2001", 2);  
        minhaPlaylist.adicionarMusica("Ten", "Pearl Jam", "1991", 1);  
  
        System.out.println("=== Lista por Nome de Musica ===");  
        minhaPlaylist.mostrarListaDeReproducao();  
  
        System.out.println("\n=== Lista por Autor ===");  
        minhaPlaylist.setModoDeReproducao(ModoDeReproducao.porAutor);  
        minhaPlaylist.mostrarListaDeReproducao();  
  
        System.out.println("\n=== Lista por Ano ===");  
        minhaPlaylist.setModoDeReproducao(ModoDeReproducao.porAno);  
        minhaPlaylist.mostrarListaDeReproducao();  
  
        System.out.println("\n=== Lista por Estrela ===");  
        minhaPlaylist.setModoDeReproducao(ModoDeReproducao.porEstrela);  
        minhaPlaylist.mostrarListaDeReproducao();  
    }  
}
```

# Vejamos a Playlist

```
public enum ModoDeReproducao {  
    porNome, porAutor, porAno, porEstrela  
}
```

```
public class PlayList {  
    protected ArrayList<MusicaMP3> musicas;  
    protected OrdenadorTemplate ordenador;  
    public PlayList(ModoDeReproducao modo) {  
        musicas = new ArrayList<MusicaMP3>();  
        switch (modo) {  
            case porAno:  
                ordenador = new OrdenadorPorAno();  
                break;  
            case porAutor:  
                ordenador = new OrdenadorPorAutor();  
                break;  
            case porEstrela:  
                ordenador = new OrdenadorPorEstrela();  
                break;  
            case porNome:  
                ordenador = new OrdenadorPorNome();  
                break;  
            default:  
                break;  
        }  
    }  
  
    public void setModoDeReproducao(ModoDeReproducao modo) {  
        ordenador = null;  
        switch (modo) {  
            case porAno:  
                ordenador = new OrdenadorPorAno();  
                break;  
            case porAutor:  
                ordenador = new OrdenadorPorAutor();  
                break;  
            case porEstrela:  
                ordenador = new OrdenadorPorEstrela();  
                break;  
            case porNome:  
                ordenador = new OrdenadorPorNome();  
                break;  
            default:  
                break;  
        }  
    }  
}
```

# Mais Playlist



```
public void adicionarMusica(String nome, String autor, String ano,
    int estrela) {
    musicas.add(new MusicaMP3(nome, autor, ano, estrela));
}

public void mostrarListaDeReproducao() {
    ArrayList<MusicaMP3> novaLista = new ArrayList<MusicaMP3>();
    novaLista = ordenador.ordenarMusica(musicas);

    for (MusicaMP3 musica : novaLista) {
        System.out.println(musica.nome + " - " + musica.autor + "\n Ano: "
            + musica.ano + "\n Estrelas: " + musica.estrelas);
    }
}
}
```



# OrdenadorTemplate

```
public abstract class OrdenadorTemplate {  
    public abstract boolean isPrimeiro(MusicaMP3 musical, MusicaMP3 musica2);  
    protected ArrayList<MusicaMP3> ordenarMusica(ArrayList<MusicaMP3> lista) {  Método Template  
        ArrayList<MusicaMP3> novaLista = new ArrayList<MusicaMP3>();  
        for (MusicaMP3 musicaMP3 : lista) {  
            novaLista.add(musicaMP3);  
        }  
        for (int i = 0; i < novaLista.size(); i++) {  
            for (int j = i; j < novaLista.size(); j++) {  
                if (!isPrimeiro(novaLista.get(i), novaLista.get(j))) {  Usando uma operação  
                    MusicaMP3 temp = novaLista.get(j);  
                    novaLista.set(j, novaLista.get(i));  
                    novaLista.set(i, temp);  
                }  
            }  
        }  
        return novaLista;  
    }  
}
```

# Ordenadores

```
public class OrdenadorPorAno extends OrdenadorTemplate {  
    @Override  
    public boolean isPrimeiro(MusicaMP3 musical, MusicaMP3 musica2) {  
        if (musical.ano.compareToIgnoreCase(musica2.ano) <= 0) {  
            return true;  
        }  
        return false;  
    }  
}
```

```
public class OrdenadorPorAutor extends OrdenadorTemplate {  
    @Override  
    public boolean isPrimeiro(MusicaMP3 musical, MusicaMP3 musica2) {  
        if (musical.autor.compareToIgnoreCase(musica2.autor) <= 0) {  
            return true;  
        }  
        return false;  
    }  
}
```

# Dúvidas?

