

Apresentação de nossos códigos!

By Joaquim and Hericky

20/10/2023

Contendo a página de Agendamentos e...

Apresentaremos o vídeo ao final, para
mostrar totalmente os códigos

Começando com a tela de Agendamentos
Falaremos sobre:

Construtores:

Cada classe possui construtores que inicializam os atributos quando um evento é criado.

```
1  class Evento {  
2      public Evento(String nome, Date data) {  
3          this.nome = nome;  
4          this.data = data;  
5      }  
6  }  
7  
8  class ShowDeMusica extends Evento {  
9      public ShowDeMusica(String nome, Date data, int capacidade, double precoIngresso) {  
10         super(nome, data);  
11         this.capacidade = capacidade;  
12         this.precoIngresso = precoIngresso;  
13     }  
14 }  
15  
16 class ShowDeComedia extends Evento {  
17     public ShowDeComedia(String nome, Date data, int faixaEtaria) {  
18         super(nome, data);  
19         this.faixaEtaria = faixaEtaria;  
20     }  
21 }  
22
```



Logo em seguida temos:

ArrayList:

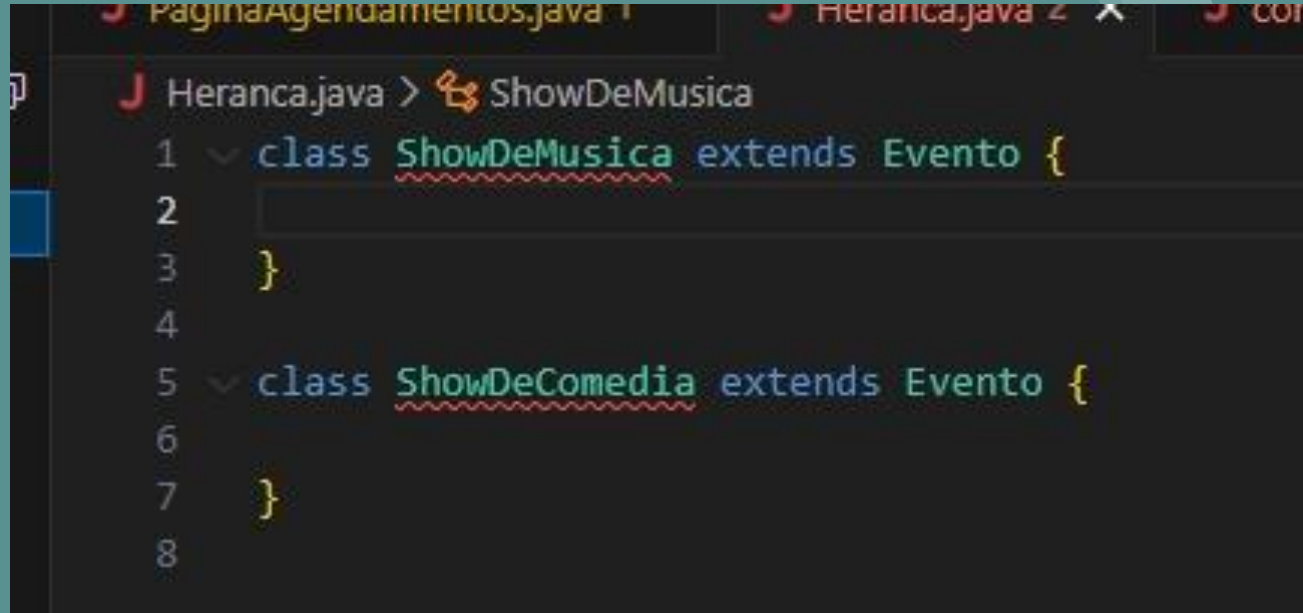
A estrutura de dados ArrayList é usada para armazenar uma lista dinâmica de eventos, permitindo o agendamento de múltiplos eventos.

```
1  
2 public class PaginaAgendamentos {  
3     private static ArrayList<Evento> eventos = new ArrayList<Evento>();  
4 }
```

Run | Debug

Herança:

A relação de herança entre Evento, ShowDeMusica e ShowDeComedia permite que as subclasses herdem atributos e métodos da superclasse, promovendo a reutilização de código.

A screenshot of a Java IDE window titled 'Heranca.java'. The editor shows two class definitions. The first class is 'ShowDeMusica' which extends 'Evento'. The second class is 'ShowDeComedia' which also extends 'Evento'. The code is as follows:

```
1 class ShowDeMusica extends Evento {  
2  
3 }  
4  
5 class ShowDeComedia extends Evento {  
6  
7 }  
8
```

J encapsulamento.java > Evento > Evento(String, Date)

```
1  class Evento {
2      private String nome;
3      private Date data;
4
5      public Evento(String nome, Date data) {
6          this.nome = nome;
7          this.data = data;
8      }
9
10     public String getNome() {
11         return nome;
12     }
13
14     public void setNome(String nome) {
15         this.nome = nome;
16     }
17
18     public Date getData() {
19         return data;
20     }
21
22     public void setData(Date data) {
23         this.data = data;
24     }
25 }
```

Encapsulamento:

Os atributos nas classes são definidos como privados, garantindo que só possam ser acessados por meio de métodos get e set, aplicando o encapsulamento.

Os códigos completos aparecerão no vídeo

Polimorfismo:

A função AgendarEvento pode aceitar tanto objetos de ShowDeMusica quanto de ShowDeComedia, exemplificando o polimorfismo.

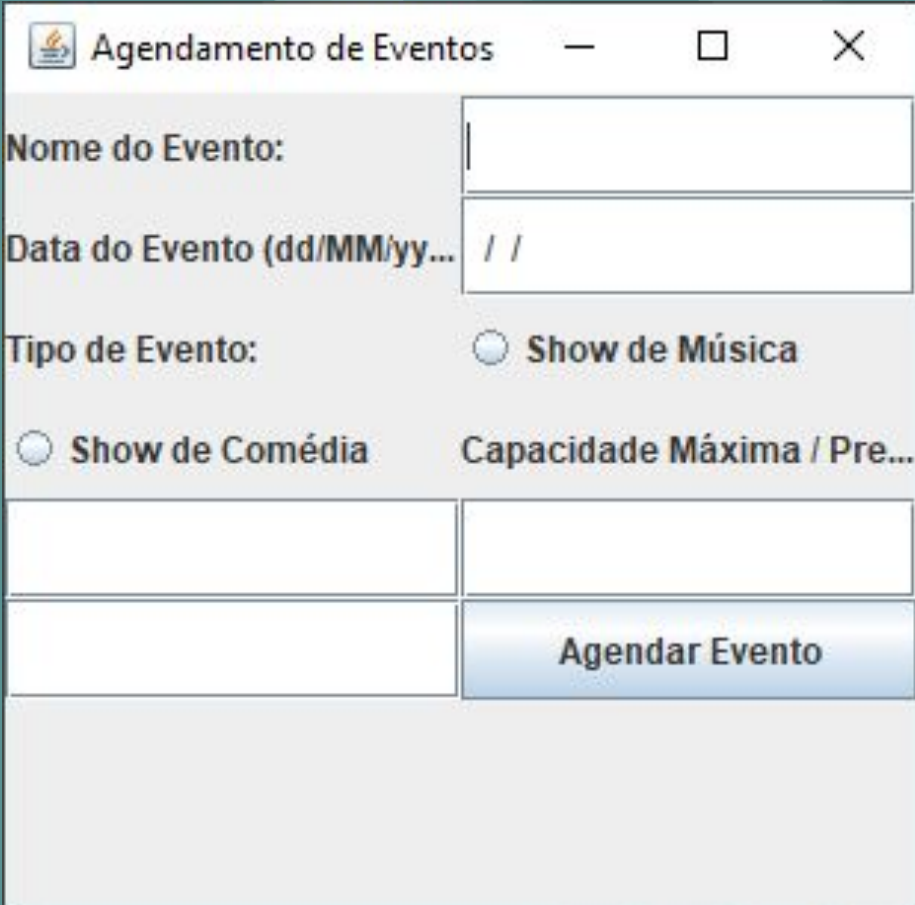
(Que não esta de forma explicita)

J polimorfismo.java

```
1  ArrayList<Evento> eventos = new ArrayList<Evento>();
2
3  eventos.add(new ShowDeMusica("Show de Música", data, capacidade, precoIngresso));
4  eventos.add(new ShowDeComedia("Show de Comédia", data, faixaEtaria));
5
6  for (Evento evento : eventos) {
7      System.out.println("Nome: " + evento.getNome());
8      System.out.println("Data: " + evento.getData());
9  }
10
11
```

Interface gráfica:

A interface gráfica é o que permite aos usuários inserir as informações sobre o evento, como nome, data, capacidade e preço do ingresso e agendar o evento.



The image shows a window titled "Agendamento de Eventos" (Event Scheduling). It contains several input fields and a button:

- Nome do Evento:** A text input field.
- Data do Evento (dd/MM/yy...):** A date input field with slashes as placeholders.
- Tipo de Evento:** Two radio button options: "Show de Música" (selected) and "Show de Comédia".
- Capacidade Máxima / Pre...:** A label for a field that is partially cut off.
- Agendar Evento:** A blue button to submit the form.