Lista 9 - Revisão de POO

Prazo de entrega: 22/06/2024 até 23:59 (Sábado)

Instruções de entrega: Você deve ter um repositório em seu github para a disciplina de POO chamado **Programacao-Orientado-Objetos**.
Crie uma pasta dentro dele chamada Listas e dentro dela crie uma pasta chamada Lista9.
Para cada exercício da lista deve ser criado um projeto java.
Ao terminar a lista, suba seus exercícios no github e envie o link no formulário abaixo.

Link para o formulário de envio: https://forms.gle/9GQ9ANPseQVpobxNA

MONITORIAS: Serão realizados quatro dias de monitorias para auxiliar com a resolução dos exercícios. As datas e horários das monitorias são:

Dia	Horários		
Segunda-Feira	19h até 21h		
Terça-Feira	19h até 21h	-	-
Quarta-Feira	09h até 11h	19h até 21h	-
Quinta-Feira	19h até 21h	-	-
Sexta-Feira	09h até 11h	19h até 21h	22h até 23:59

OBS: Se você não conseguir participar das monitorias nos dias e horários propostos, você pode solicitar uma monitoria em uma data e horário diferente, desde que seja comunicado com pelo menos um dia de antecedência 🙂

EXERCÍCIO

Você foi contratado pela prefeitura da cidade para desenvolver um sistema de gestão de eventos culturais abertos para toda a população, ficou sob sua responsabilidade modelar o que é um evento para o sistema, e também suas implementações para show e stand-up..

1. Crie um pacote chamado `eventos`.

2. Dentro do pacote `eventos`, crie a classe abstrata `Evento`.

- **Atributos**:
  - `nome: String` - Representa o nome do evento.
  - `local: String` - Representa o local onde o evento será realizado.
  - `data: String` - Representa a data do evento.
- Os atributos devem ser privados.
- **Construtor**:
  - Crie um construtor que inicialize os atributos `nome`, `local` e `data` com os valores fornecidos.
- **Métodos**:
  - `public void exibirInformacoes()`: Este método deve imprimir o nome, local e data do evento.
  - `public abstract void comecarEvento()`: Método abstrato para iniciar o evento.
- **Getters e Setters**:
  - Desenvolva getters e setters para cada um dos atributos.

3. Crie a classe `Show` que herda de `Evento`.

- **Atributo adicional**:
  - `artistas: List<String>` - Representa a lista de artistas que vão realizar o show.
- **Construtor**:

    - Crie um construtor que inicialize os atributos `nome`, `local`, `data`(atributos da superclasse) e `artistas`(atributo desta classe) com os valores fornecidos.
- **Métodos**:
    - Sobrescreva o método `exibirInformacoes()` para exibir também a lista de artistas.
    - Sobrescreva o método `começarEvento()` para iniciar o show, exibindo uma mensagem específica de início de show.
    - **Sobrecarregado**:
        - `public void começarEvento(boolean comAplausos)`: Método para iniciar o show com ou sem aplausos.

4. Crie a classe final `StandUp` que herda de `Evento`.

- **Atributo adicional**:
    - `comediante: String` - Representa o nome do comediante da apresentação de stand-up.
- **Construtor**:
    - Crie um construtor que inicialize os atributos `nome`, `local`, `data`(atributos da superclasse) e `comediante`(atributo desta classe) com os valores fornecidos.
- **Métodos**:
    - Sobrescreva o método `exibirInformacoes()` para exibir também o nome do comediante.
    - Sobrescreva o método `começarEvento()` para iniciar a apresentação de stand-up, exibindo uma mensagem específica de início da apresentação.
    - **Sobrecarregado**:
        - `public void começarEvento(String piada)`: Método para iniciar a apresentação com uma piada.

5. Crie a classe `App` com o método `main`.

- **Método main**:
    - Crie instâncias de `Show` e `StandUp` utilizando os construtores com parâmetros.
    - Utilize o método `começarEvento` (com e sem sobrecarga) para iniciar os eventos de diferentes tipos.
    - Exiba as informações dos eventos utilizando o método `exibirInformacoes`.