Uma imagem com sentado, preto, computador

Descrição gerada automaticamente

**Faculdade de Ciências Exatas e de Engenharia**

2019/2020

**Programação Orientada por Objetos**

**Festival de Cinema**

**Docentes:**

Mónica Cameirão

Sergi Bermúdez

**Trabalho realizado por:**

Diego Andrés da Silva Briceño (nº 2043818)

Sílvia da Silva Fernandes (nº 2043118)

Rúben José Gouveia Rodrigues (nº 2046018)

Funchal, 17 de maio de 2020

INDICE

1. **Introdução**

Este relatório tem por objetivo demonstrar e explicar o funcionamento de uma aplicação que permite gerir um festival de cinema, usando para tal a plataforma *NetBeans IDE* e a linguagem de programação *Java*.

Este relatório explicará os objetivos principais deste trabalho, tal como a forma de implementação e os procedimentos realizados para tal fim e a justificação pela qual decidiu-se implementá-los. E no final, é incluído, em anexo, o diagrama UML, que demonstra toda a estruturação do software desenvolvido.

1. **Objetivos**

Este trabalho tem por objetivos, aplicar os conhecimentos adquiridos na unidade curricular de Programação Orientada por Objetos com o propósito de criar uma aplicação que permita administrar várias edições de um festival de cinema, podendo o utilizador inserir e consultar a informação que pretende.

Resumidamente, em cada edição do festival, poderão participar vários filmes. Cada filme é caracterizado pelo género, tem um realizador e nele participam vários atores. Um filme não poderá ter mais do que um ator ou atriz principais. Os atores distinguem-se pelo nome e anos de carreira, e só poderão participar no máximo de dois filmes em uma edição. No festival, tanto os filmes quanto os atores competem para ganhar prémios. A cada prémio competem 4 candidatos, cujo vencedor é decidido com base na média das pontuações (1-10) dadas por um conjunto de peritos.

1. **Procedimento e implementação do código**
   1. **Classes**
      1. **Pessoa**

Esta classe, serve para especificar os métodos e atributos básicos que distinguem uma pessoa, ou seja, nome e género. Por esse mesmo motivo, a classe pessoa foi declarada como abstrata (*abstract*), pois apenas idealiza um rascunho. Ao declarar esta classe como abstrata estamos a impedir que se criem instâncias da mesma, o que faz sentido, uma vez que no festival apenas existem atores, realizadores e peritos.

O nome é do tipo *String* e o género do tipo boolean, ou seja, se a pessoa for do género masculino, este atributo será true, e se for do género feminino, false. Sendo assim, no construtor desta classe, definiu-se esses mesmos atributos. Definiu-se dois métodos *getter*: **getNome** e **getGenero**. E também implementou-se os métodos **toString** e **equals**. O **toString** apenas irá imprimir o nome e género da pessoa e o método **equals** serve para comparar se um objeto é igual a outro desta classe.

* + 1. **Edição**

A classe Edição controla toda a informação da respetiva edição do festival.

No início desta classe, efetuou-se uma chamada da *package* “java.util.ArrayList” para podermos utilizar todas as funcionalidades ligadas com ArrayLists.

Esta classe tem como atributos: “numEdicao” e “ano” do tipo int, o ArrayList<Filme> “filmes” contendo os filmes (classe Filme) que participam na edição, o ArrayList<Perito> “peritos” onde são guardados os peritos (classe Perito) que irão pontuar os candidatos aos prémios e o ArrayList<Premio> “premios” que contém todos os prémios (classe Premio) que serão concedidos naquela edição.

O construtor desta classe é declarado como sendo do tipo *protect*, com o propósito de apenas poder ser alterado ou acedido somente por classes da mesma *package*. E apenas recebe como parâmetros o número e o ano da Edição, que serão guardados nas variáveis “numEdicao” e “ano”, respetivamente. Para além disto, inicializa os restantes atributos para, posteriormente, ser inserida a informação nos respetivos ArrayLists, e também chama o método **inserePremios**.

O método **inserePremios,** tal como o nome indica, cria todos os prémios que os filmes e os atores irão competir e, insere-os no ArrayList correspondente aos prémios da edição. Deste modo, todas as edições terão os mesmos nove prémios em jogo:

* Melhor Ator Principal
* Melhor Atriz Principal
* Melhor Ator Secundário
* Melhor Atriz Secundária
* Melhor Filme
* Melhor Realizador
* Melhor Argumento
* Melhor Cinematografia
* Prémio Carreira

O método **insereFilmes**, insere o novo filme que o utilizador pretende que participe no festival, no ArrayList de filmes.

O método **inserePerito**, insere o novo perito criado no ArrayList de peritos da Edição, e também o adiciona ao ArrayList de ArrayLists de pontuações de cada objeto da classe Prémio existente na Edição, inicializando a zero cada posição do ArrayList correspondente a esse Perito (será explicado melhor na classe Prémio).

Nesta classe definiu-se vários métodos *getter*: o método **getPeritos** que retorna a lista com todos os peritos da Edição, o método **getNumEdicao** que retorna o número da Edição, o método **getFilmes** que retorna a lista com todos os filmes que participam na Edição, e por último o método **getPremios** que retorna a lista de todos os prémios.

Para imprimir as informações sobre os filmes participantes da Edição e as categorias que serão premiadas, implementou-se, respetivamente, os métodos **imprimeFilmes** (caso ainda não existam filmes registados, imprime uma mensagem com essa informação) e **imprimePremios,** que imprime o nome de todas as categorias.

Para além disto, esta classe também possui outros quatro métodos para imprimir informação importante:

O método **listarFilmesMaisPremiados**, tal como o nome indica, imprime os filmes que possuem um ou mais prémios. A classe Filme tem um atributo do tipo int intitulado de “numeroPremios”, caso este atributo seja superior a zero, irá ser imprimido o nome do filme em questão e o número de prémios que este já ganhou.

O método **listarCandidatos** imprime os quatro candidatos a cada um dos prémios. Caso um prémio não tenha nenhum candidato, aparece uma mensagem a dizer que esse prémio não tem candidatos.

O método **listarVencedores** lista o vencedor de cada prémio, e o método **listarPontuaçõesOrdenadas** lista os candidatos de cada prémio pela ordem da sua pontuação (de maior para menor pontuação).

E por fim, o método **toString** que imprime o ano e número da respetiva Edição.

O método **equals** serve para comparar se um objeto é igual a outro desta classe.

* + 1. **Perito**

Os peritos são aqueles que avaliam os candidatos aos prémios, sendo a sua avaliação dada numa escala de 1 a 10.

Esta classe é uma extensão da classe Pessoa, e herda todos os métodos dessa mesma classe. Deste modo, no construtor faz-se uma chamada ao construtor da superclasse Pessoa, através do método **super**, para guardar no nome e género os parâmetros recebidos do perito criado.

Para podermos inserir a pontuação dada por um perito a um dado candidato em um dado prémio, definiu-se o método **inserePontuacao**. Este método recebe como parâmetros: o “premio” (classe Premio) que está sendo avaliado, o parâmetro do tipo int “indiceCandidato”, que corresponde à posição da ArrayList das pontuações que indica a qual candidato estamos a atribuir a pontuação, o “indicePerito” (do tipo int) que indica a posição da ArrayList da posição da Arraylist mencionada anteriormente pontuações que indica o perito, e por último o parâmetro “scan” que servirá para o programa ler o que é escrito no teclado. Caso a pontuação inserida esteja entre os limites válidos, é inserido na ArrayList da posição “indiceCandidato” da ArrayList das pontuações, na posição “indicePerito” a pontuação inserida.

E por fim, esta classe também possui os métodos **toString** e **equals**. O método **toString** faz uma chamada ao método **toString** da superclasse para imprimir o nome e género do perito e o método **equals** serve para comparar se um objeto é igual a outro desta classe.

* + 1. **Realizador**

Todos os filmes têm que ter obrigatoriamente um realizador.

À semelhança do perito, o realizador é distinguindo apenas pelo seu nome e género. Sendo assim, o construtor desta classe recebe como parâmetros o nome e género, e faz uma chamada ao construtor da superclasse Pessoa, através do método super com esses mesmos parâmetros.

O método **toString** é idêntico ao da classe Perito e o método **equals** compara se um objeto é igual a outro desta classe.

* + 1. **Filme**

No festival de cinema participam vários filmes. Os filmes são caracterizados pelo género, e possuem um realizador e vários atores. Num filme apenas existe um ator e atriz principais, sendo os demais atores secundários.

Como atributos esta classe tem o “nome” e “género” do tipo String (nome e género do filme), a “edição” do tipo int (número da edição em que o filme participa) e “numeroPremios” também do tipo int (número de prémios que o filme ganhou). Para além disto, esta classe tem como parâmetros o “realizador” do filme (classe Realizador), o ArrayList “atoresSecundarios” contendo todos os atores secundários do filme (classe Ator), e por fim temos o “AtorPrincipal” e a “AtrizPrincipal”, ambos da classe Ator.

O construtor recebe como parâmetros: o nome, o género, a edição e o realizador do filme e inicializa o “numeroPremios” a zero, o “AtorPrincipal” e a “AtrizPrincipal” a null e o ArrayList dos “atoresSecundarios”, sendo assim possível criar um filme sem atores. Os atores são posteriormente adicionados através de métodos *setter*.

Todos os atributos desta classe, exceto a “edição”, possuem métodos *getter* que retornam os valores ou variáveis guardadas nesse atributo.

Definiu-se nesta classe o método **insereAtor**, que permite inserir qualquer tipo de ator (principal ou secundário) no filme. Deste modo este método recebe como parâmetros: o “ator” a ser inserido e a variável do tipo boolean “principal”. Caso este último parâmetro seja true, significa que pretende-se inserir no filme um ator ou atriz principais. Como um filme apenas pode ter um ator e atriz principais, o método verifica primeiramente o género do ator que pretende-se inserir e depois verifica se o atributo “AtorPrincipal” ou “AtrizPrincipal” (dependendo do género do ator) encontra-se a null. Caso isso aconteça é apresentada uma mensagem a dizer que o filme já possui um ator ou atriz principais. Caso contrário, insere o ator no filme e adiciona na lista de filmes que o ator participa esse mesmo filme. Já se o parâmetro “principal” for false, significa que pretende-se adicionar um ator secundário. Para inserir um ator secundário são necessárias duas verificações: verificar se o ator que pretende-se inserir já é um ator ou atriz principais e verificar se esse ator também já não se encontra na lista de atores secundários do filme. Estas duas verificações têm como propósito evitar que se repitam os mesmos atores num filme. Se o ator ainda não pertence á gama de atores do filme, então ele é adicionado á lista “atoresSecundarios”, e é adicionado na lista de filmes que o ator participa esse mesmo filme, caso contrário, é apresentada uma mensagem a dizer que esse ator já está no filme.

O método **incrementaNumeroPremios**, tal como o nome indica, incrementa o número de prémios do filme á medida que o filme os vai ganhando.

E por fim, como sempre, temos os métodos **toString** e **equals**. O método **toString** imprime as informações mais relevantes do filme: nome, género, realizador e todos os nomes dos atores (ator e atriz principais e atores secundários). E o método **equals** compara se um objeto é igual a outro desta classe.

* + 1. **Ator**

Assim como o perito e o realizador, o ator também é uma subclasse de Pessoa.

Esta classe tem como atributos do tipo int: “anosCarreira” (anos de carreira do ator), “numFilmesEdiçãoAtual” (número de filmes em que o ator participa na edição atual) e “numFilmesTotal” (número de filmes em que o ator participa no total, ou seja, considerando todas as edições do festival de cinema). E também possui como atributo o ArrayList contendo os filmes em que este participa.

O construtor tem como parâmetros: o “nome”, o “género” e os “anosCarreira” do ator. Aqui faz-se uma chamada ao construtor da superclasse Pessoa, inicializa-se os atributos “numFilmesEdiçãoAtual” e “numFilmesTotal” a zero e também atribuiu-se a “anosCarreira” o valor do parâmetro “anosCarreira” e inicializa-se o ArrayList dos filmes, para posteriormente serem inseridos os filmes.

Aqui implementou-se três métodos *getter*: **getAnosCarreira** (retorna os anos de carreira do ator), **getFilmes** (retorna o ArrayList contendo os filmes em que o ator participa) e **getnumFilmesEdiçãoAtual** (retorna o número de filmes em que o ator participa na edição atual).

O método **resetNumFilmesEdicaoAtual**, coloca novamente a zero o atributo “numFilmesEdiçãoAtual”.

O método **incrementaAnosCarreira**, tal como o nome indica, incrementa os anos de carreira do ator, ou seja, quando é criada uma nova edição, os anos de carreira de todos os atores que participaram nas edições anteriores é incrementado.

O método **podeInserirFilme** diz-nos se podemos ou não inserir um filme na lista de filmes do ator. Como um ator só pode participar em um máximo de dois filmes em cada edição do festival, caso o ator já participe em dois filmes, este método retorna false, caso contrário retorna true.

Para inserir um filme na lista de filmes definiu-se o método **inserirFilme**, que para além de inserir o filme, incrementa os atributos “numFilmesEdiçãoAtual” e “numFilmesTotal”.

E por último, o método **toString** imprime o nome, o género, os anos de carreira do ator, e o nome dos filmes em que este participa (caso o ator participe em algum filme). E o método **equals** compara se um objeto é igual a outro desta classe.

* + 1. **Premio**
    2. **FestivalCinema**
    3. **Main**

1. **Conclusão**
2. **Anexos**