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

Índice

[1. Introdução 3](#_Toc40363254)

[2. Objetivos 3](#_Toc40363255)

[3. Procedimento e implementação do código 3](#_Toc40363256)

[3.1. Classes 3](#_Toc40363257)

[3.1.1. Pessoa 3](#_Toc40363258)

[3.1.2. Edição 4](#_Toc40363259)

[3.1.3. Perito 5](#_Toc40363260)

[3.1.4. Realizador 6](#_Toc40363261)

[3.1.5. Filme 6](#_Toc40363262)

[3.1.6. Ator 7](#_Toc40363263)

[3.1.7. Premio 8](#_Toc40363264)

[3.1.8. FestivalCinema 9](#_Toc40363265)

[3.1.9. Main 15](#_Toc40363266)

[4. Conclusão 16](#_Toc40363267)

[5. Anexos 16](#_Toc40363268)

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 numa 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. 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 “filmes” contendo os filmes (classe Filme) que participam na edição, o ArrayList “peritos” onde são guardados os peritos (classe Perito) que irão pontuar os candidatos aos prémios e o ArrayList “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).

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, se o ator ainda não participa nesse filme (como ator secundário), 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 um atributo do tipo int “anosCarreira” (anos de carreira do ator), e também possui como atributo o ArrayList “filmesParticipa” contendo os filmes (classe Filme) 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, atribuiu-se a “anosCarreira” o valor do parâmetro “anosCarreira” e inicializa-se o ArrayList dos filmes, para posteriormente serem inseridos os filmes.

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 **resetFilmesEdicaoAtual** remove todos os filmes que estavam na lista (“filmesParticipa “) de filmes que o ator participa.

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**.

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**

Logo no começo desta classe, efetuou-se uma chamada, para além da *package* “java.util.ArrayList”, à *package* java.util.Collections. Esta última *package* disponibiliza serviços para coleções de objetos e é utilizada apenas no método **swap**, essencialmente para trocar os elementos das posições especificadas na lista fornecida.

Esta classe tem como atributos: do tipo String o “nome” (nome do prémio), do tipo array “mediasPontuacoes” (que contem elementos do tipo double) e da classe Filme tem-se o “vencedor” (filme vencedor). Para além disto tem-se vários ArrayLists: o ArrayList “filmes” que contem os filmes candidatos ao prémio (classe Filme), o ArrayList “atores” que contem os atores candidatos ao prémio, e por fim o ArrayList “pontuações”, cujos elementos são também ArrayLists de números inteiros.

O construtor recebe como parâmetro apenas o nome do prémio. A partir do nome do prémio determina-se se iremos usar o ArrayList de atores e/ou de filmes. Se o nome do prémio não conter a palavra “Atriz” ou “Ator” ou “Carreira”, significa que se trata de um prémio relacionado com os filmes e é colocado o ArrayList de atores a null por questões de memória e tratamento de exceções. Já se o nome do prémio contem a palavra “Carreira” é colocado o ArrayList de filmes a null pelos mesmos motivos. Estabelece-se também que o array com as médias das pontuações terá quatro posições (uma para cada candidato ao prémio) e inicializa-se os restantes ArrayLists. O ArrayList com as pontuações também terá quatro posições, sendo que em cada posição é inicializado um ArrayList para serem posteriormente inseridas as pontuações que os peritos deram para esse candidato. E por fim, o atributo “vencedor” é inicializado a null.

O método **setPontuacao** recebe três parâmetros do tipo int: “candidato”, “perito” e “pontuação”. Para alterar a pontuação do candidato, o método coloca na posição do candidato no ArrayList “pontuações” a respetiva pontuação na posição do perito que pontuou esse candidato.

O método **nomeiaFilme** adiciona à lista de filmes do prémio o filme que este recebe como parâmetro.

Esta classe tem dois métodos **nomeiaAtor**. Um deles apenas recebe como parâmetro o ator que irá ser inserido na lista de atores candidatos. E o outro para além de receber como parâmetro o ator que irá ser inserido na lista de atores candidatos, também recebe o filme pelo qual esse ator está a ser nomeado no prémio. O último método adiciona à lista de atores candidatos o ator que se pretende inserir e chama o método **nomeiaFilme**. Foi necessário definir dois métodos distintos porque os atores que concorrem ao prémio carreira não necessitam de ter um filme associado à sua vitória, pois o que interessa são os seus anos de carreira e os passos percorridos ao longo da mesma.

O método **toString** apenas imprime o nome do prémio.

O método **calcularMedias**, tal como o nome indica, calcula as médias das pontuações atribuídas aos candidatos e coloca-as no array “mediasPontuacoes”.

O método **ordenaPontuações** é responsável por ordenar as listas consoante as médias das pontuações dos candidatos, usando um bubble sort, e depois de ordenados, verificamos se houve empates, com o método empateVencedores, e atualiza-se a variável vencedor com determinaVencedor.

O método **swap** é auxiliar ao bubble sort e troca os valores das médias, troca os arrays de inteiros na lista das pontuações, troca os atores nomeados, se a lista não for nula e troca os filmes nomeados, se esta lista não for nula. Assim, tendo uma posição i, nessa posição i está toda a informação do candidato.

O método **empateVencedores** é responsável por resolver os empates, se houver, com recurso aos desvios padrões das pontuações dos candidatos, usando outra vez um bubble sort.

O método **determinaVencedor** é responsável por simplesmente definir o vencedor do prémio e incrementar o número de prémios do filme que venceu.

O método **imprimePontuações** é responsável por imprimir as médias das pontuações dos nomeados ao prémio. Se os nomeados ao prémio ainda não foram definidos, o utilizador é informado e se estes já foram definidos mas as pontuações ainda não foram atribuídas, o utilizador também é informado.

O método **vencedorCategoria** é semelhante ao anterior, só que este imprime o vencedor do prémio, se houver candidatos ao prémio e as pontuações já tenham sido atribuídas

* + 1. **FestivalCinema**

A classe FestivalCinema é a principal classe deste programa. É esta classe que relaciona todas as classes. Tem como atributos uma ArrayList “edicoes”, onde são guardadas todas a edições do festival (classe Edicao), um ArrayList “atores” onde são inseridos todos os atores que pertencem ao festival, um inteiro “ano” que corresponde ao ano em que a edição do festival está a decorrer, um inteiro “numEdição” que corresponde ao número da edição a decorrer, um “scan” da classe Scanner, importada da package java.util.Scanner, que permite fazer a leitura dos dados inseridos pelo utilizador, um boolean “quebra” responsável por manter o programa em funcionamento (enquanto for false) e, uma String “opcao” que é utilizada para guardar as opções do utilizador em quase todo o programa.

O construtor desta classe inicializa os ArrayLists mencionados anteriormente, inicializa o “ano” e “numEdicao” a “0”, cria um objeto da classe Scanner, e coloca o “quebra” a false.

O método **menu** é responsável pela interface do programa, é ele que permite o utilizador visualizar os diferentes menus durante a execução do programa, fazer leitura de dados e realizar a chamada de quase todos os métodos principais do mesmo. Inicialmente, é pedido ao utilizador para inserir o ano da primeira edição do festival de cinema e o atributo “quebra” é colocado a “false”. Se ele for false, o programa estará constantemente a ser executado até o utilizador decidir sair do programa (sendo neste caso, colocada a “true”). Após o utilizador inserir o ano da primeira edição, sempre que for criada uma nova edição, o ano será incrementado uma unidade, dando a ideia de que a nova edição está a ocorrer no ano seguinte.

O método **criarFilme** permite ao utilizador criar um novo filme (objeto da classe Filme) na edição a decorrer no festival. Para isso, é pedido ao utilizador para inserir o nome, género e, o nome do realizador e género do mesmo. Para não existirem objetos distintos da classe Realizador idênticos, após a criação do realizador, é verificado se o mesmo já existe no programa, ou seja, é o realizador de outro filme que já participa no festival, sendo nesse caso, passada a referência do Realizador que já existia no programa. Após a criação do filme, este é inserido na ArrayList de filmes da edição corrente.

O método criarAtor permite ao utilizador criar um novo ator/atriz (objeto da classe Ator). Para isso, é pedido ao utilizado para inserir o nome do ator/atriz, o género e os anos de carreira do mesmo. Após a criação do utilizador, este é inserido na ArrayList de atores do FestivalCinema.

O método **criarPerito** é semelhante ao criarAtor, sendo neste caso, criado um perito (objeto da classe Perito), que como já foi referido, é responsável pela atribuição das pontuações aos candidatos aos prémios da edição. Para criar um perito é pedido ao utilizador para inserir o nome e género do mesmo. Após a criação do perito, é verificado se o mesmo já existe no programa. Caso ele não exista, é inserido na ArrayList de peritos da edição.

O método **atribuirPapel** permite ao utilizador atribuir um papel num filme a um dado ator/atriz existente no festival, caso haja pelo menos um ator/atriz no festival. Inicialmente é mostrada a lista de todos os atores criados e, de seguida, é pedido ao utilizador para inserir o nome do ator/atriz que pretende inserir num filme. Após a escolha, é verificado se o ator/atriz escolhido não participa em 2 ou mais filmes da edição corrente. Se ele participar em 2 ou mais filmes, o utilizador é informado e regressa-se ao menu inicial do programa. Caso contrário, são listados todos os filmes da edição e é pedido ao utilizador para escolher o filme em que deseja inserir o ator/atriz. Se não existir nenhum filme nessa edição, regressa-se ao menu inicial. Se o filme escolhido existir, é pedido ao utilizador para atribuir o papel de principal ou secundário. O ator/atriz só será inserido no filme se for permitido pelo método insereAtor da classe Filme.

O método **consultarEdicoes** permite listar todas as edições criadas no festival de cinema, sendo mostrado o número e ano de cada uma. Caso não existam edições no festival, é mostrada uma informação sobre isso.

O método **listarAtores** recebe como parâmetro um boolean “atual” que quando é true, permite ser listado os atores que participam em filmes da edição corrente, se for false, permite ser listado todos os atores existentes no festival (incluindo os que não participam em nenhum filme). Se não existirem atores no programa ou nenhum ator da edição corrente participar nalgum filme, o utilizador é informado.

O método **escolherCandidatos** permite verificar o prémio cujo utilizador deseja escolher os candidatos. Para isso, é verificado o nome do prémio escolhido e é feita a chamada do método correspondente à escolha dos candidatos do prémio escolhido.

O método **escolherPremio** pede ao utilizador para escolher um dos 9 prémios existentes na edição e retorna o objeto da classe Premio correspondente ao prémio escolhido.

O método **escolherFilmesCandidatos** recebe como parâmetro o prémio (Melhor Filme, Melhor Argumento ou Melhor Cinematografia) cujo utilizador pretende escolher os candidatos. Antes de efetuar a escolha, é verificado se o tamanho da ArrayList com os 4 filmes candidatos ao prémio é 4, ou seja, o prémio já tem candidatos. Se não for 4, significa que ainda não foi realizada a escolha dos candidatos para o dado prémio, sendo assim feita a listagem de todos os filmes existentes na edição corrente. À medida que é feita a listagem, cada filme é colocado numa ArrayList “possiveisCandidatos” onde são colocados todos os possíveis candidatos ao prémio. Se existirem menos de 4 possíveis candidatos ao prémio, o utilizador é informado e não é feita escolha. Caso contrário, é pedido ao utilizador para indicar o nome do filme que pretende escolher como candidato. Ao ser escolhido um determinado filme, ele é removido da ArrayList “possiveisCandidatos” de modo a evitar que possa voltar a ser escolhido. O utilizador só poderá continuar após escolher os 4 candidatos ao prémio.

O método **escolherRealizadorCandidatos** recebe como parâmetro o prémio (Melhor Realizador) cujo utilizador pretende escolher os realizadores candidatos. Antes de efetuar a escolha, é verificado se o prémio ainda não tem candidatos (tamanho da ArrayList de filmes do prémio é diferente de 4). Se isso se verificar, é feita a listagem de todos os realizadores (incluindo o nome do seu filme). À medida que é feita a listagem, o realizador do filme é colocado numa ArrayList de “possíveisCandidatos” e o filme correspondente a esse realizador é colocado na Arraylist de “filmesPossiveisCandidatos”. Como é feita a contagem do número de candidatos possíveis, só é contabilizado como um possível candidato um realizador que ainda não esteja nos “possiveisCandidatos”, de modo a evitar que existam menos de 4 candidatos diferentes e a verificação seguinte (de não haver pelo menos 4 candidatos diferentes) não se verificar. Caso se verifique, é pedido ao utilizador para inserir o nome do realizador que pretende escolher como candidato. Como o mesmo realizador pode participar em vários filmes, procura-se a posição do mesmo nos “possiveisCandidatos”. De seguida, verifica-se se o realizador escolhido apenas aparece uma vez. Se isso acontecer, o filme a qual o realizador escolhido está associado é inserido como candidato ao prémio e, o realizador é removido dos “possiveisCandidatos” e o seu filme dos “filmesPossiveisCandidatos” (para evitar que possa voltar a ser escolhido). Se o realizador escolhido aparece mais do que uma vez (realizador de vários filmes), o utilizador é informado e é pedido para ele inserir o nome do filme a que se refere. Se o realizador do filme escolhido foi igual ao realizador escolhido anteriormente, o filme é inserido como candidato ao prémio e é feita a remoção de todas as ocorrências desse realizador nos “possiveisCandidatos” e de todos os filmes relacionados com ele nos “filmesPossiveisCandidatos”, evitando assim que seja escolhido novamente.

O método **escolherAtoresPrincipaisCandidatos** recebe o prémio (Melhor Ator ou Melhor Atriz) cujo utilizador pretende escolher os candidatos, e um boolean “homem” que permite identificar se será feita a escolha dos melhores atores candidatos (“homem” a true) ou a escolha das melhores atrizes candidatas (“homem” a false). Tal como no método anterior, é feita a verificação de que ainda não foi feita a escolha dos candidatos a esse prémio. De seguida, é feita a listagem dos atores (ou atrizes) principais existentes na edição corrente, sendo colocados na lista de “possiveisCandidatos” os atores e na lista de “filmesPossiveisCandidatos” os filmes em que eles participam. Como um mesmo ator pode participar em dois filmes, é feita a contagem dos atores (ou atrizes) diferentes, contabilizando-o assim apenas uma vez, caso isso aconteça. Se não houver 4 possíveis candidatos diferentes, não é realizada a escolha. Caso contrário, é pedido ao utilizador para escolher o nome do ator (ou da atriz) que pretende escolher como candidato. Se esse ator só participa num filme como personagem principal (só aparece uma vez nos “possiveisCandidatos”), é inserido como candidato ao prémio e é feita a remoção do ator dos “possivesCandidatos” e do filme em que participa dos “filmesPossivesCandidatos”. Se o ator (ou atriz) participar em dois filmes como principal, é pedido ao utilizador para indicar o nome do filme a que se refere. Sendo depois feita a verificação de se o ator (ou atriz) principal do filme escolhido é o mesmo ao escolhido. Se isso acontecer, o ator (ou atriz) e o filme correspondente são inseridos como candidatos ao prémio, e é feita a remoção de todas as ocorrências do mesmo ator e de todos os filmes em que ele participa como principal dos “possiveisCandidatos” e “filmesPossiveisCandidatos”, respetivamente, evitando que possa ser escolhido novamente.

O método **escolherAtoresSecundariosCandidatos** recebe como parâmetro o prémio (Melhor Ator Secundário ou Melhor Atriz Principal) cujo utilizador pretende escolher os candidatos e um boolean “homem” para identificar se será feita a escolha dos melhores atores secundários (“homem” a true), ou melhores atrizes secundárias (“homem” a false). Este método funciona de forma idêntica ao método escolherAtoresPrincipaisCandidatos, a única diferença é que é feita a listagem de todos os atores secundários (ou atrizes secundárias), sendo estes colocados na ArrayList “possivesCandidatos”.

O método **escolherPremioCarreira** recebe como parâmetro o prémio (Prémio Carreira) cujo utilizador pretende escolher os candidatos. É verificado, também, se os candidatos já foram escolhidos. Caso isso não acontecer, é feita a listagem de todos os atores existentes no festival que têm mais do que 20 anos de carreira, sendo cada um inserido nos “possiveisCandidatos”. Se não houver mais do que 4 candidatos diferentes, não é realizada a escolha. Caso contrário, é pedido ao utilizador para inserir o nome do ator que pretende escolher como candidato ao prémio. Após a escolha, o ator é inserido como candidato ao prémio e é removido dos “possiveisCandidatos” para evitar que seja escolhido novamente.

O método **pontuarCandidatos** recebe como parâmetro o prémio que o utilizador pretende que os candidatos sejam avaliados de 1 a 10. Inicialmente, é verificado se o prémio correspondente já não tem vencedor, se isso acontecer, é realizada a avaliação dos candidatos. A variável boolean “pontuou”, inicialmente a “false”, só será “true” se houver pelo menos um perito criado. Se não houver nenhum, não é realizada a avaliação do prémio pelo facto de não haver peritos. Se houver, é verificado se o nome do prémio em questão contém a palavra “Ator”, “Atriz” ou “Carreira” (o prémio está relacionado com atores). Se isso se verificar, todos os peritos avaliam um candidato de cada vez, inserindo um valor entre 1 a 10. Se o nome do prémio em questão não contém nenhuma dessas palavras, significa que o prémio está relacionado com filmes ou realizadores. Se o prémio conter a palavra “Realizador”, é mostrado o nome do realizador e do seu filme, caso contrário é mostrado apenas o nome do filme candidato. Todos os peritos avaliam um candidato de cada vez, da mesma forma.

O método **novoOuCarregar** é responsável pelas opções de carregar dados (através de ficheiros de texto) ou da criação de um novo programa. Caso o utilizador pretenda carregar dados é criada uma nova edição (edição seguinte), é pedido para escolher que tipo de dados pretende carregar. Caso o utilizador escolha “Carregar Tudo”, é feita a leitura de todos os ficheiros de texto e também é feita a chamada do método ordenadaPontuações a cada um dos prémios da edição, de modo a ser feita a determinação do vencedor de cada prémio. Após o carregamento de todos os dados, é criada a nova edição criada é inserida no festival. Se o utilizador decidir criar um novo programa, é criada uma nova edição (edição seguinte) sem nenhuns dados e adicionada ao festival.

Os métodos **indexOfByActorName** e **indexOfByFilmName** são dois métodos auxiliares que retornam o índice do ator ou filme na lista passada como parâmetro cujo nome iguala aquele passado como parâmetro. Se o dito ator ou filme não existir na lista, retorna -1.

O método **recebeInteiro** é outro método auxiliar que retorna o valor inteiro que o utilizador pretende, e se o utilizador não meter um valor inteiro, informa o utilizador e espera que o utilizador insira um valor inteiro.

Em termos de carregar os dados do ficheiro, dividiu-se este processo em 5 métodos, visto que os dados estão gravados em 5 ficheiros diferentes, um ficheiro para os atores que participam nos filmes da edição, um ficheiro para os filmes da edição, outro para os nomeados aos prémios, outro para os peritos da edição e um último para as pontuações atribuídas aos nomeados.

Todos os métodos que carregam dados fazem isto recorrendo a objetos da classe FileReader, para abrir a stream, e BufferedReader, este último para aumentar a eficiência da leitura do ficheiro, reduzindo o número de acessoas a este e para se poder ler os ficheiros linha a linha, usando o método *readLine()*. Quando se chega ao fim do ficheiro que estamos a ler fecha-se a stream.

O método que carrega os dados relativos aos atores chama-se **carregaAtores**, carrega os dados do ficheiro “Atores.txt” na pasta EdicaoX onde X é o número da edição da qual os dados pretendemos carregar.

O método lê o ficheiro três linhas de cada vez, para registarmos o nome, género e os anos de carreira, cria-se um objeto Ator para percorrer-se a lista de atores e verificar- se este já foi criado (recorrendo ao método equals) e, se ainda não está na lista, adiciona-se o ator à lista, e por fim acresenta-se o ator ao filme.

O método que carrega os dados relativos aos filmes chama-se **carregaFilmes** e é extremamente semelhante ao anterior, só que desta vez relativamente ao ficheiro “Filmes.txt”, presente na mesma pasta mencionada anteriormente.

Lê-se o ficheiro 4 linhas de cada vez, uma para cada atributo, cria-se o objeto realizador para comparar-se este com os realizadores previamente criados, de filmes previamente carregados e/ou criados ao longo do programa, de modo a não se criar realizadores “iguais” (com o mesmo nome e género), cria-se o filme e acrescenta-se este à lista de filmes da edição que estamos a carregar.

O método que carrega os candidatos aos filmes chama-se **carregaCandidatos** e é semelhante aos anteriores, o nome do ficheiro é “Candidatos.txt” e encontra-se na mesma pasta mencionada anteriormente, cada linha tem o nome do nomeado ao prémio, e se for um dos prémios relativos aos atores, a linha seguinte tem o nome do filme pelo qual o ator foi nomeado. Lê-se os dados dos nomeados e percorre-se os dados previamente carregados (dos filmes e/ou atores) e guarda-se na lista de nomeados relevante do prémio o ator e/ou filme mencionado no ficheiro.

O método que carrega os peritos chama-se **carregaPeritos**, o nome do ficheiro é “Peritos.txt” e encontra-se na pasta mencionada anteriormente, e cada par de linhas é os dados de um perito, temos o nome deste e na linha seguinte o seu género. Lê-se os dados, criamos o perito e acrescentamos este na lista de peritos da edição.

Por fim, o método que carrega as pontuações chama-se **carregaPontuações**, o nome do ficheiro que contém estes dados chama-se “Pontuacoes.txt” e encontra-se na mesma pasta que os outros ficheiros, e contém as pontuações dos nomeados de todos os prémios. Cada linha lida representa as pontuações atribuídas a um nomeado, onde as pontuações encontram-se separadas por asteriscos. Depois de lida a pontuação (chegou-se a um asterisco) acrescenta-se a mesma à lista de pontuações do nomeado e continua-se.

Para gravar os dados de uma edição dividiu-se a tarefa em 5 métodos, como os métodos que carregam os dados. Todos os métodos recorrem a um FileWriter, para abrir a stream, um BufferedWriter, para aumentar a eficiência, e a um PrintWriter, para escrever os dados. Em todos os métodos que gravam dados, os dados são guardados no ficheiro no mesmo formato em que são lidos nos métodos que carregam os dados, para facilitar o processo de carregar dados previamente gravados. Quando já se gravou todos os dados, fecha-se a stream.

O método **gravaAtores** é responsável por gravar os dados dos atores presentes nos filmes da edição cujos dados estamos a gravar. Percorre-se a lista de filmes da edição que estamos a guardar e guarda-se os dados dos atores presentes nos filmes.

O método **gravaFilmes** é responsável por gravar os dados dos filmes (nome, género e dados do realizador) da edição a guardar. Percorre-se a lista de filmes da edição e guarda-se os dados mencionados.

O método **gravaCandidatos** é responsável por gravar os dados dos nomeados aos prémios. Percorre-se a lista de prémios e por cada prémio percorre-se a lista de candidatos relevante e guarda-se os dados dos nomeados.

O método **gravaPeritos** é responsável por gravar os dados dos peritos. Percorre-se a lista de peritos da edição e grava-se o nome e género dos peritos.

Por fim, o método **gravaPontuacoes** é responsável por gravar as pontuações dos nomeados aos prémios. Percorre-se a lista dos prémios da edição e por cada prémio percorre-se a lista das pontuações dos candidatos e gravam-se as pontuações de cada nomeado em cada linha, separadas por asteriscos.

* + 1. **Main**

O Main é a classe principal do programa, uma vez que é ela que contêm todo o código que se pretende executar. Esta classe não pode ser instanciada por outras classes, e é a primeira classe a ser invocada pelo intérprete do Java quando se executa o programa.

Esta classe apenas possui o método **main**. Neste método apenas cria-se uma nova instância da classe FestivalCinema e chama-se o método **menu** da mesma, que controla todo o funcionamento do programa.

É de realçar também que a classe Main é a única classe da *<default package>*,sendo todos as outras classes da *package* com.mycompany.festivalcinema. Decidiu-se separar as classes desta maneira para impedir que classes de outras *packages* acedam ao código do programa. Desta forma para poder-se chamar a classe FestivalCinema foi necessário efetuar um *import* da respetiva *package* juntamente com o nome dessa classe.

1. **Conclusão**

Concluindo, este projeto demonstrou a utilidade de elaborar um diagrama UML de modo a facilitar a visualização das relações entre as classes do programa.

Demonstrou também a utilidade de classes de modo a organizar o código e facilitar a localização de certos elementos de um programa.

E por fim, a elaboração do programa não se provou difícil. Graças à metodologia de elaborar este por partes e ir testando ditas partes individualmente, não se encontrou problemas graves. Este trabalho ajudou a reforçar os conhecimentos sobre a programação em linguagem Java e a programação orientada por objetos.

1. **Anexos**