UNIVERSIDADE da MADEIRA

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, 21 de maio de 2020

Índice

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.

2. 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 como 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.

3. Procedimento e implementação do código

3.1. Classes

3.1.1. 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 se implementou os métodos **toString** e **equals**. O **toString** apenas irá imprimir o nome e género da pessoa e no método **equals** dois objetos da classe Pessoas são iguais se ambos tiverem o mesmo nome e género.

3.1.2. Edicao

A classe Edicao 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 *protected*, 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 inicializa a zero cada posição do ArrayList das pontuações correspondente a esse Perito.

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.

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.

No método **equals** verificamos se duas edições são exatamente iguais, ou seja, todos os atributos são iguais.

3.1.3. 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 no método **equals** dois objetos da classe Perito são iguais se ambos tiverem o mesmo nome e género.

3.1.4. 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 no método **equals** dois objetos da classe Realizador são iguais se ambos tiverem o mesmo nome e género.

3.1.5. Filme

No festival de cinema participam vários filmes. Os filmes são caracterizados pelo seu nome, o seu género, 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) e "numeroPremios" 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 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, 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. Verifica-se se o filme já tem ator/atriz principal (dependendo do género do ator a inserir) e se o ator que se pretende inserir já participa no filme como ator secundário. Caso o filme já possua ator/atriz principal ou o ator já se encontra na lista de atores secundários, é

apresentada uma mensagem de erro. Caso contrário, procede-se à inserção do ator no filme e insere-se o filme na lista de filmes que o ator participa. Já se o parâmetro "principal" for false, significa que pretende-se adicionar um ator secundário. Verifica-se se o ator já participa no filme, como ator secundário ou principal. Estas 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 no método **equals** dois objetos da classe Filme são iguais se ambos tiverem o mesmo nome.

3.1.6. 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. Este método é extremamente útil, pois como todos os atores do festival de cinema são guardados no mesmo ArrayList ("atores"), quando criamos uma nova edição, utiliza-se este método para eliminar todos os filmes em que os atores participaram na edição anterior (uma vez que, um ator não pode participar em mais do que dois filmes em uma edição), e assim, caso algum ator que participou em outras edições participe novamente na edição atual, não é necessário criá-lo novamente, apenas temos que inserir o/s filme/s em que este participa e atribuir o seu papel. Este método também é útil para diferenciar os atores da edição atual dos demais na hora de listar os atores. Como os filmes em que os atores participaram foram apagados, os atores da edição atual são aqueles que participam em pelo menos um filme.

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 ("filmesParticipa") 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). no método **equals** dois objetos da classe Ator são iguais se ambos tiverem o mesmo nome e género.

3.1.7. 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. 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 arrayLists 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. Em caso de empate, o vencedor do desempate é aquele cujas pontuações tiverem menor desvio padrão.

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.

E por fim, o método **equals** verifica se dois objetos desta classe têm todos os mesmos atributos.

3.1.8. FestivalCinema

A classe FestivalCinema é a principal classe deste programa. É esta classe que relaciona todas as classes. Tem como atributos um 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 "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, 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 ator, é verificado se o mesmo já existe no programa. Caso ele não exista, é inserido na ArrayList de atores.

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 prémio já tem candidatos. Se o prémio não tem candidatos é feita a listagem de todos os filmes existentes na edição corrente (os possíveis candidatos). 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 lista de possíveis candidatos 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. Se isso se verificar, é feita a listagem de todos os realizadores (incluindo o nome do seu filme). Caso não haja pelo menos 4 diferentes realizadores candidatos ao prémio, informa-se o utilizador. Caso haja pelo menos 4 diferentes realizadores candidatos ao prémio, é pedido ao utilizador para inserir o nome do realizador que pretende escolher como candidato. Se o realizador escolhido é realizador de vários filmes na edição corrente, 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, nomeia-se o realizador ao prémio, pelo filme que o utilizador escolheu, e remove-se o mesmo da lista de possíveis candidatos, para o mesmo realizador não poder ser nomeado duas vezes. Se o realizador

só direciona um filme, nomeia-se logo este e o filme que ele direcionou e remove-se o realizador da lista de possíveis candidatos.

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. 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 nomeia-se o mesmo e o filme em qual este participa, removendo o ator da lista de possíveis candidatos. 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, nomeia-se o ator (ou atriz) pelo filme que o utilizador indicou e remove-se o ator da lista de possíveis candidatos, para este não poder ser nomeado pelo outro filme em que participa.

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) e a escolha de quais atores (ou atrizes) devem ser nomeados ao prémio.

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. 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 nomeado ao prémio e removido da lista de possíveis candidatos.

O método **pontuarCandidatos** recebe como parâmetro o prémio que o utilizador pretende pontuar. 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 inserir 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 acessos 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**, e carrega os dados do ficheiro "Atores.txt" na pasta EdicaoX onde X é o número da edição da qual pretendemos carregar os dados.

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 acrescenta-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.

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.

Por fim, tem-se o método **limparConsola**. Este método serve para limpar a consola onde é executado a programa (evitando assim o acúmulo de informação na consola, à medida que o programa é executado), ou seja, simula o pressionar das teclas CTRL + L. Para efetuar tal simulação, recorreu-se à classe *java.awt.Robot* que, em termos simples, fornece controle sobre o mouse e o teclado. Para implementar este método foi necessário também efetuar o *import* das classes *java.awt.AWTException* (utilizada para lançar uma exceção quando ocorre algum problema no AWT, com a mensagem de detalhe especificada) e *java.awt.event.KeyEvent*, que indica se uma tecla foi ou não pressionada.

3.1.9. 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.

4. 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.

5. Anexo A – Código (por classes)

5.1. Ator

```
package com.mycompany.festivalcinema;

import java.util.ArrayList;

public class Ator extends Pessoa {

    private int anosCarreira;
    private final ArrayList<Filme> filmesParticipa;

    protected Ator(String nome, boolean genero, int anosCarreira)
{
        super(nome, genero);
        this.anosCarreira = anosCarreira;
        this.filmesParticipa = new ArrayList<>(0);
    }

    protected void resetFilmesEdicaoAtual() {
        this.filmesParticipa.clear();
    }

    protected void incrementaAnosCarreira() {
        this.anosCarreira++;
    }

    /**
     *
     * @return true se o ator participa em menos de 2 filmes na
edição atual,
     * false caso contrario
     */
    protected boolean podeInserirFilme() {
        return this.filmesParticipa.size() < 2;
    }

    /**
     * Método que guarda um filme no catalogo de filmes do ator e
atualiza as
     * variáveis que contam os filmes
     *
     * @param filme - filme a inserir no catalogo de filmes do ator
     */
    protected void inserirFilme(Filme filme) {
        this.filmesParticipa.add(filme);
    }

    protected int getAnosCarreira() {
        return anosCarreira;
    }

    protected ArrayList<Filme> getFilmes() {
        return filmesParticipa;
    }

    protected int getnumFilmesEdiçãoAtual() {
        return this.filmesParticipa.size();
    }
```

```
        @Override
        public String toString() {
            String ator = super.toString();
            ator += "Anos de Carreira: " + anosCarreira + "\n";
            if (!this.filmesParticipa.isEmpty()) {
                ator += "Filmes em que participa: \n";
                for (Filme f : this.filmesParticipa) {
                    ator += f.getNome() + "\n";
                }
            } else {
                ator += "Não participa em nenhum filme desta edição.\n";
            }
            return ator;
        }

        @Override
        public boolean equals(Object obj) {
            if (obj == null) {
                return false;
            }
            if (obj == this) {
                return true;
            }
            if (this.getClass() != obj.getClass()) {
                return false;
            }
            Ator comparar = (Ator) obj;
            return   this.getGenero()   ==   comparar.getGenero()   &&
this.getNome().equalsIgnoreCase(comparar.getNome());
        }
    }
```

5.2. Filme

```
    package com.mycompany.festivalcinema;

    import java.util.ArrayList;

    public class Filme {

        private final String nome;
        private final String genero;
        private int numeroPremios;
        private final Realizador realizador;
        private Ator AtorPrincipal;
        private Ator AtrizPrincipal;
        private final ArrayList<Ator> atoresSecundarios;

        protected  Filme(String  nome,  String  genero,  Realizador
realizador) {
            this.nome = nome;
            this.genero = genero;
            this.realizador = realizador;
            this.AtorPrincipal = null;
            this.AtrizPrincipal = null;
            this.atoresSecundarios = new ArrayList<>(0);
            this.numeroPremios = 0;
        }

        protected String getNome() {
```

```
            return nome;
        }

        protected String getGenero() {
            return genero;
        }

        protected Ator getAtorPrincipal() {
            return AtorPrincipal;
        }

        protected Ator getAtrizPrincipal() {
            return AtrizPrincipal;
        }

        protected ArrayList<Ator> getAtoresSecundarios() {
            return atoresSecundarios;
        }

        protected Realizador getRealizador() {
            return realizador;
        }

        protected int getNumeroPremios() {
            return numeroPremios;
        }

        protected void incrementaNumeroPremios() {
            numeroPremios++;
        }

        /**
         * Método que insere um ator no filme, se é principal ou
secundário depende
         * da posição
         *
         * @param ator - ator que se vai inserir no filme, se possível
         * @param principal - indica se o ator é suposto ser principal
ou secundário
         * (true - principal, false - secundário)
         */
        protected void insereAtor(Ator ator, boolean principal) {
            if (principal) {
                for (Ator a : this.atoresSecundarios) {
                    if (ator == a) { //verifica se o ator já está na
lista de atores secundários
                        System.out.println((ator.getGenero()    ?    "O
ator" : "A atriz") + " já está no filme!");
                        return;
                    }
                }
                if (ator.getGenero() && this.AtorPrincipal == null) {
                    this.AtorPrincipal = ator;
                    ator.inserirFilme(this); //insere o filme na lista
de filmes em que o ator participa
                } else if (!ator.getGenero() && this.AtrizPrincipal ==
null) {
                    this.AtrizPrincipal = ator;
                    ator.inserirFilme(this); //insere o filme na lista
de filmes em que o atriz participa
                } else {
```

```
                    System.out.println(this.nome + " já tem" +
(ator.getGenero() ? " um ator principal!" : " uma atriz principal!"));
                }
            } else {
                if (ator == this.AtorPrincipal || ator ==
this.AtrizPrincipal) { //evita que o ator seja inserido como secundário
se já é principal
                    System.out.println((ator.getGenero() ? "O ator" :
"A atriz") + " já está no filme!");
                    return;
                }
                for (Ator a : this.atoresSecundarios) {
                    if (ator == a) { //verifica se o ator já está na
lista de atores secundários
                        System.out.println((ator.getGenero() ? "O
ator" : "A atriz") + " já está no filme!");
                        return;
                    }
                }
                this.atoresSecundarios.add(ator);
                ator.inserirFilme(this); //insere o filme na lista de
filmes em que o ator/atriz participa
            }
        }

        @Override
        public String toString() {
            String filme;
            filme = "Filme: " + nome + "\n";
            filme += "Género: " + genero + "\n";
            filme += "Realizador: " + realizador.getNome() + "\n";
            filme += "Ator Principal: " + (AtorPrincipal != null ?
AtorPrincipal.getNome() : "") + "\n";
            filme += "Atriz Principal: " + (AtrizPrincipal != null ?
AtrizPrincipal.getNome() : "") + "\n";
            filme += "Atores Secundários:\n";
            for (Ator secundario : this.atoresSecundarios) {
                filme += secundario.getNome() + "\n";
            }
            return filme;
        }

        @Override
        public boolean equals(Object obj) {
            if (obj == null) {
                return false;
            }
            if (obj == this) {
                return true;
            }
            if (this.getClass() != obj.getClass()) {
                return false;
            }
            Filme filme = (Filme) obj;
            return this.nome.equalsIgnoreCase(filme.nome);
        }
    }
```

5.3. Edicao

```
package com.mycompany.festivalcinema;

import java.util.ArrayList;

public class Edicao {

    private final int numEdicao;
    private final int ano;
    private final ArrayList<Filme> filmes;
    private final ArrayList<Premio> premios;
    private final ArrayList<Perito> peritos;

    protected Edicao(int numEdicao, int ano) {

        this.numEdicao = numEdicao;
        this.ano = ano;
        this.premios = new ArrayList<>(9);
        this.filmes = new ArrayList<>();
        this.peritos = new ArrayList<>();
        inserePremios();
    }

    @Override
    public String toString() {
        String Edicao;
        Edicao = "Edição: " + numEdicao + " | Ano: " + ano;

        return Edicao;
    }

    protected ArrayList<Perito> getPeritos() {
        return this.peritos;
    }

    /**
     * Método responsável por criar a lista dos 9 prémios de cada
edição
     */
    protected final void inserePremios() {
        this.premios.add(new Premio("Melhor Ator Principal"));
        this.premios.add(new Premio("Melhor Atriz Principal"));
        this.premios.add(new Premio("Melhor Ator Secundário"));
        this.premios.add(new Premio("Melhor Atriz Secundária"));
        this.premios.add(new Premio("Melhor Filme"));
        this.premios.add(new Premio("Melhor Realizador"));
        this.premios.add(new Premio("Melhor Argumento"));
        this.premios.add(new Premio("Melhor Cinematografia"));
        this.premios.add(new Premio("Prémio Carreira"));
    }

    protected void insereFilmes(Filme filme) {
        this.filmes.add(filme);
    }

    protected void inserePerito(Perito perito) {
        this.peritos.add(perito);
        for (Premio p : this.premios) { //acede a cada prémio
            for (ArrayList<Integer> lista : p.getPontuacoes()) {
```

```
//acede à lista de pontuações de cada candidato
                    lista.add(0);   //coloca   a   pontuação   0   (não
avaliado), em cada candidato do prémio
                }
            }
        }

        protected int getNumEdicao() {
            return numEdicao;
        }

        protected ArrayList<Filme> getFilmes() {
            return this.filmes;
        }

        protected ArrayList<Premio> getPremios() {
            return this.premios;
        }

        /**
         * Método que lista todos os filmes da edição
         */
        protected void imprimeFilmes() {
            System.out.println("EDIÇÃO: " + this.numEdicao);
            if (filmes.isEmpty()) {
                System.out.println("! NÃO HÁ FILMES REGISTADOS !");
            } else {
                System.out.println("\nFILMES:\n");
                for (Filme filme : this.filmes) {
                    System.out.println(filme);
                }
            }
        }

        /**
         * Método que lista os Premios a premiar na edição atual
         */
        protected void imprimePremios() {
            System.out.println("CATEGORIAS A PREMIAR:");
            int indice = 0;
            while (indice < premios.size()) {
                System.out.println(premios.get(indice));
                indice++;
            }
        }

        /**
         * Método que lista os filmes com pelo menos um prémio
         */
        protected void listarFilmesMaisPremiados() {
            boolean  semPremiados  =  true;  //true  se  nenhum  filme  foi
premiado
            System.out.println("FILMES MAIS PREMIADOS: ");
            for (Filme filme : this.filmes) {
                if (filme.getNumeroPremios() > 0) { //se o filme tiver
pelo menos 1 prémio
                    System.out.println(filme.getNome()   +   ":   "   +
filme.getNumeroPremios());
                    semPremiados = false;
                }
            }
```

```
            if (semPremiados) {
                System.out.println("Nenhum filme foi premiado.");
            }
        }

        /**
         * Método que mostra os quatro candidatos ao prémio em cada
categoria
         */
        protected void listarCandidatos() {
            int contaPremios = 1;
            System.out.println("CANDIDATOS AOS PRÉMIOS:");
            for (Premio premio : this.premios) {
                System.out.println();
                System.out.println(premio + ":");
                try {
                    if (contaPremios <= 4) { //se o prémio está
relacionado com atores
                        for (int i = 0; i < 4; i++) {
                            System.out.printf("-    %s    em    %s\n",
premio.getAtoresCandidatos().get(i).getNome(),
premio.getFilmesCandidatos().get(i).getNome());
                        }
                    } else if (contaPremios > 4 && contaPremios != 6 &&
contaPremios != 9) { //se o prémio está relacionado com filmes
                        for (int i = 0; i < 4; i++) {
                            System.out.println("-          "          +
premio.getFilmesCandidatos().get(i).getNome());

                        }
                    } else if (contaPremios == 9) { //se é o prémio
carreira
                        for (int i = 0; i < 4; i++) {
                            System.out.println("-          "          +
premio.getAtoresCandidatos().get(i).getNome());
                        }
                    } else { //se é o prémio de melhor realizador
                        for (int i = 0; i < 4; i++) {
                            System.out.println("-          "          +
premio.getFilmesCandidatos().get(i).getRealizador().getNome() + " por "
+ premio.getFilmesCandidatos().get(i).getNome());
                        }
                    }
                } catch (NullPointerException |
IndexOutOfBoundsException e) {
                    System.out.println("- Não tem candidatos.\n");
                }
                contaPremios++;
            }
        }

        /**
         * Método que lista os vencedores dos prémios
         */
        protected void listarVencedores() {
            System.out.println("VENCEDORES:");
            for (Premio premio : this.premios) {
                premio.vencedorCategoria();
            }
        }
```

```
        /**
         * Método que, como o nome indica, lista os candidatos de cada
prémio pela
         * ordem da sua pontuação
         */
        protected void listarPontuaçõesOrdenadas() {
            System.out.println("PONTUAÇÕES: ");
            for (Premio premio : this.premios) {
                premio.imprimePontuações();
            }
        }

        @Override
        public boolean equals(Object obj) {
            if (obj == null) {
                return false;
            }
            if (obj == this) {
                return true;
            }
            if (this.getClass() != obj.getClass()) {
                return false;
            }
            Edicao edic = (Edicao) obj;
            return   this.ano   ==   edic.ano   &&   this.numEdicao   ==
edic.numEdicao       &&       this.filmes.equals(edic.filmes)       &&
this.peritos.equals(edic.peritos) && this.premios.equals(edic.premios);
        }
    }
```

5.4. FestivalCinema

```
    package com.mycompany.festivalcinema;

    import java.awt.AWTException;
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    import java.util.ArrayList;
    import java.util.Scanner;
    import java.io.*;

    public class FestivalCinema {

        private final ArrayList<Edicao> edicoes;
        private final ArrayList<Ator> atores;
        private int ano;
        private int numEdicao; //numEdicao - 1 dá o índice da edição na
lista edicoes da edicao com número numEdicao
        private final Scanner scan;
        private boolean quebra;
        private String opcao;

        public FestivalCinema() {
            this.edicoes = new ArrayList<>();
            this.atores = new ArrayList<>();
            this.opcao = "";
            this.ano = 0;
            this.numEdicao = 0;
            this.scan = new Scanner(System.in, "cp1252");
            this.quebra = false;
        }
```

```
        /**
         * O método que trata do display das opções possíveis, de
receber a opção
         * que o utilizador escolheu e chamar o método que trata da
operação que o
         * utilizador escolheu
         */
        public void menu() {
            System.out.println("\t\t\tFESTIVAL CINEMA");
            System.out.print("\nIndique o ano da edição do festival:
");
            ano = recebeInteiro();
            novoOuCarregar();
            quebra = false;
            while (!quebra) {
                System.out.print("Opções:\n(c):           Criar\n(l):
Listar\n(p): Atribuir Papel\n(h): Nova Edição\n(g): Gravar Dados\n(s):
Sair\nOpção: ");
                opcao = scan.nextLine().trim().toLowerCase();
                limparConsola();
                switch (opcao.toLowerCase()) {
                    case "c":
                        System.out.print("Opções:\n(f):          Criar
Filme\n(a):  Criar  Ator/Atriz\n(p):  Criar  Perito\n(c):  Escolher
candidatos\n(s): Atribuir Pontuação\nOpção: ");
                        opcao = scan.nextLine().trim().toLowerCase();
                        limparConsola();
                        switch (opcao) {
                            case "f":
                                criarFilme();
                                break;
                            case "a":
                                criarAtor();
                                break;
                            case "p":
                                boolean parar = false;
                                while (!parar) {
                                    criarPerito();
                                    System.out.print("\nPrima  1  se
pretende criar mais um perito e outro número caso contrário\nOpção: ");
                                    int maisUm = recebeInteiro();
                                    if (maisUm != 1) {
                                        parar = true;
                                    }
                                }
                                break;
                            case "c":
                                escolherCandidatos();
                                break;
                            case "s":
                                pontuarCandidatos(escolherPremio());
                                break;
                            default:
                                System.out.println("Opção inválida.");
                        }
                        break;
                    case "l":
                        System.out.print("Opções:\n(a):          Listar
Atores\n(f):  Listar  Filmes\n(p):  Listar  Prémios\n(i):  Consultar
Edições\nOpção: ");
```

```
                        opcao = scan.nextLine().trim().toLowerCase();
                        limparConsola();
                        switch (opcao) {
                            case "a":
                                System.out.print("Opções:\n(a): Listar
da Edição Atual\n(t): Listar de todas as Edições\nOpção: ");
                                opcao                                   =
scan.nextLine().trim().toLowerCase();
                                limparConsola();
                                switch (opcao) {
                                    case "a":
                                        listarAtores(true);
                                        break;
                                    case "t":
                                        listarAtores(false);
                                        break;
                                    default:
                                        System.out.println("Opção
inválida.");
                                }
                                break;
                            case "f":
                                System.out.print("Opções:\n(f): Listar
Filmes\n(p): Listar Filmes Mais Premiados\nOpção: ");
                                opcao = scan.nextLine().trim();
                                limparConsola();
                                switch (opcao) {
                                    case "f":
                                        boolean imprimiu = false;
                                        consultarEdicoes();
                                        while (!imprimiu) {
                                            try {

System.out.print("\nInsira o número da edição.\nOpção: ");
                                                int    opcaoPremio    =
recebeInteiro();

edicoes.get(opcaoPremio - 1).imprimeFilmes();
                                                imprimiu = true;
                                            }                    catch
(IndexOutOfBoundsException e) {

System.out.print("\nEssa edição não existe.\n");
                                            }
                                        }
                                        break;
                                    case "p":
                                        edicoes.get(numEdicao        -
1).listarFilmesMaisPremiados();
                                        break;
                                    default:
                                        System.out.println("Opção
inválida.");
                                }
                                break;

                            case "p":
                                System.out.print("Opções:\n(p): Listar
Categorias\n(c): Listar Candidatos\n(o): Listar Candidatos (Ordenados
por Avaliação)\n(v): Listar Vencedores\nOpção: ");
                                opcao                                   =
```

```
scan.nextLine().trim().toLowerCase();
                                limparConsola();
                                switch (opcao) {
                                    case "p":
                                        edicoes.get(numEdicao        -
1).imprimePremios();
                                        break;
                                    case "c":
                                        edicoes.get(numEdicao        -
1).listarCandidatos();
                                        break;
                                    case "o":
                                        edicoes.get(numEdicao        -
1).listarPontuaçõesOrdenadas();
                                        break;
                                    case "v":
                                        edicoes.get(numEdicao        -
1).listarVencedores();
                                        break;
                                    default:
                                        System.out.println("Opção
inválida.");
                                }
                                break;
                            case "i":
                                consultarEdicoes();
                                break;
                            default:
                                System.out.println("Opção inválida.");
                        }
                        break;
                    case "p":
                        atribuirPapel();
                        break;
                    case "h":
                        for (Ator a : atores) {
                            a.resetFilmesEdicaoAtual();
                        }
                        for (Ator a : atores) {
                            a.incrementaAnosCarreira();
                        }
                        novoOuCarregar();
                        quebra = false;
                        break;
                    case "g":
                        System.out.print("Opções:\n(a): Gravar Atores
e Filmes\n(c): Gravar Atores, Filmes e Candidatos\n(p): Gravar
Peritos\n(t): Gravar Tudo\nOpção: ");
                        opcao = scan.nextLine().trim().toLowerCase();
                        limparConsola();
                        try {
                            switch (opcao) {
                                case "a":
                                    gravaAtores();
                                    gravaFilmes();
                                    break;
                                case "c":
                                    gravaAtores();
                                    gravaFilmes();
                                    gravaCandidatos();
                                    break;
```

```
                                case "p":
                                    gravaPeritos();
                                    break;
                                case "t":
                                    gravaAtores();
                                    gravaFilmes();
                                    gravaCandidatos();
                                    gravaPeritos();
                                    gravaPontuações();
                                    break;
                                default:
                                    System.out.println("Opção
inválida.");
                            }
                        } catch (IOException e) {
                            System.out.println("Erro ao gravar os
dados.\n");
                        }
                        break;
                    case "s":
                        quebra = true;
                        break;
                    default:
                        System.out.println("Opção inválida.");
                }
            }
        }

        /**
         * Método que cria um Filme como o nome indica
         */
        private void criarFilme() {
            System.out.print("NOVO FILME\nNome do Filme: ");
            String nome = scan.nextLine().trim();
            System.out.print("Género do Filme: ");
            String genero = scan.nextLine().trim();
            System.out.print("Nome do Realizador: ");
            String nomeRealizador = scan.nextLine().trim();
            System.out.print("Género do Realizador (M-Masculino; F-
Feminino): ");
            String generoRealizador = scan.nextLine().trim();
            while (!(generoRealizador.equalsIgnoreCase("M") ||
generoRealizador.equalsIgnoreCase("F"))) { //verifica que inseriu uma
opção válida
                System.out.print("Género Inválido. Género (M-
Masculino; F-Feminino): ");
                generoRealizador = scan.nextLine().trim();
            }
            Realizador realizador = new Realizador(nomeRealizador,
generoRealizador.equalsIgnoreCase("M"));
            for (Edicao e : edicoes) {
                for (Filme f : e.getFilmes()) { //percorrer os filmes
previamente criados para verificar se o realizador já existe
                    if (f.getRealizador().equals(realizador)) { //se o
realizador criado já existia
                        realizador = f.getRealizador(); //o realizador
do novo filme aponta para mesmo realizador anteriormente criado
                    }
                }
            }
            Filme filme = new Filme(nome, genero, realizador);
```

```
            if (!edicoes.get(numEdicao - 1).getFilmes().contains(filme)) { //se o filme criado não existir na edição
                edicoes.get(numEdicao - 1).insereFilmes(filme); //insere na lista de filmes da edição atual o filme criado
            } else {
                System.out.println("Esse filme já existe na edição.");
            }
        }

        /**
         * Método que cria um Ator como o nome indica
         */
        private void criarAtor() {
            System.out.print("NOVO ATOR\nNome do Ator/Atriz: ");
            String nome = scan.nextLine().trim();
            System.out.print("Ator ou Atriz (M-Masculino; F-Feminino): ");
            String genero = scan.nextLine().trim();
            while (!(genero.equalsIgnoreCase("M") || genero.equalsIgnoreCase("F"))) { //verifica que inseriu uma opção válida
                System.out.print("Género Inválido. Género (M-Masculino; F-Feminino): ");
                genero = scan.nextLine().trim();
            }
            System.out.print("Anos de carreira do Ator/Atriz: ");
            int anosCarreira = recebeInteiro();
            Ator ator = new Ator(nome, genero.equalsIgnoreCase("M"), anosCarreira);
            if (!atores.contains(ator)) { //verifica se o ator criado já existe
                atores.add(ator); //adiciona o ator criado no array de atores
            } else {
                System.out.println((genero.equalsIgnoreCase("M") ? "Esse ator " : "Essa atriz ") + "já existe!");
            }
        }

        /**
         * Método que cria um Perito como o nome indica
         */
        private void criarPerito() {
            System.out.print("NOVO PERITO\nNome do perito: ");
            String nome = scan.nextLine().trim();
            System.out.print("Género do Perito (M-Masculino; F-Feminino): ");
            String genero = scan.nextLine().trim();
            while (!(genero.equalsIgnoreCase("M") || genero.equalsIgnoreCase("F"))) { //verifica que inseriu uma opção válida
                System.out.print("Género Inválido. Género (M-Masculino; F-Feminino): ");
                genero = scan.nextLine().trim();
            }
            Perito perito = new Perito(nome, genero.equalsIgnoreCase("M"));
            if (edicoes.get(numEdicao - 1).getPeritos().contains(perito)) { //verifica se o perito criado já existe
                System.out.println("Esse perito já existe.");
                return;
```

```
            }
            edicoes.get(numEdicao - 1).inserePerito(perito); //insere
o perito na lista de peritos da edição, caso ele não exista
        }

        /**
         * Método que permite atribuir a um ator já criado um papel num
filme já
         * criado
         */
        private void atribuirPapel() {
            int i = 1;
            for (Ator a : atores) { //imprime os atores criados
anteriormente na execução do programa
                System.out.printf("%d. %s\n", i, a.getNome());
                i++;
            }
            if (i == 1) {
                System.out.println("Ainda não foram criados atores!");
                return;
            }
            System.out.println("Qual o nome do ator/atriz que pretende
inserir no filme?");
            String nomeAtor = scan.nextLine().trim();
            try {
                Ator mudar = atores.get(indexOfByActorName(nomeAtor,
this.atores)); //guarda o ator ao qual atribuir um papel, se possível,
caso contrário apanha a excepção e imprime uma mensagem
                if (mudar.podeInserirFilme()) { //verifica que o ator
não participa em 2 filmes na edição atual
                    i = 1;
                    for (Filme f : edicoes.get(numEdicao -
1).getFilmes()) { //imprime a lista de filmes da edição atual
                        System.out.printf("%d. %s\n", i, f.getNome());
                        i++;
                    }
                    if (i == 1) {
                        System.out.println("Ainda não foram criados
filmes na edição atual!");
                        return;
                    }
                    System.out.println("Em que filme " +
(mudar.getGenero() ? "este ator" : "esta atriz") + " participará?");
                    String nomeFilme = scan.nextLine().trim();
                    try {
                        Filme casting = edicoes.get(numEdicao -
1).getFilmes().get(indexOfByFilmName(nomeFilme, edicoes.get(numEdicao -
1).getFilmes())); //guarda o filme no qual se pretende inserir o ator
                        String papel = "";
                        while (!(papel.equalsIgnoreCase("P") ||
papel.equalsIgnoreCase("S"))) {//verifica que inseriu uma opção válida
                            System.out.println("Qual o papel do
ator/atriz (P-Principal ou S-Secundário)?");
                            papel = scan.nextLine().trim();
                            switch (papel.toLowerCase()) {
                                case "p": //se pretende-se que o
ator/atriz seja principal
                                    casting.insereAtor(mudar, true);
//ver o método insereAtor em Filmes
                                    break;
                                case "s": //se pretende-se que o
```

```
ator/atriz seja secundário/a
                                    casting.insereAtor(mudar,  false);
//ver o método insereAtor em Filmes
                                    break;
                                default:
                                    System.out.println("Opção
inválida.");
                            }
                        }
                    } catch (IndexOutOfBoundsException e) {
                        System.out.println("Esse  filme  não  existe
(possivelmente escreveu mal o nome).");
                    }
                } else {
                    System.out.println((mudar.getGenero() ? "O ator" :
"A atriz") + " já participa em 2 filmes na edição atual!");
                }
            } catch (IndexOutOfBoundsException e) { //no caso de o
utilizador quiser selecionar um ator que não exista
                System.out.println("Esse ator não existe (possivelmente
escreveu mal o nome).");
            }
        }

        /**
         * Método que permite consultar todas as edições registadas no
programa
         */
        private void consultarEdicoes() {
            int aux = 0;
            if (edicoes.isEmpty()) {
                System.out.println("Ainda  não  há  nenhuma  edição  do
festival!");
            } else {
                System.out.println("EDIÇÕES: ");
                while (aux < edicoes.size()) {
                    System.out.println(edicoes.get(aux));
                    aux++;
                }
            }
        }

        /**
         * Método que lista todos os Atores criados (quer carregados de
um ficheiro
         * quer criados pelo teclado)
         *
         * @param atual - se for false, então é para imprimir todos os
atores
         * previamente criados no programa, se for true só se imprime
os atores que
         * participam em filmes na edição corrente
         */
        private void listarAtores(boolean atual) {
            boolean existe = false; //false se não existem atores na
edição
            for (Ator ator : this.atores) {
                if (ator.getnumFilmesEdiçãoAtual() > 0 && atual) {
                    System.out.println(ator);
                    existe = true;
                } else if (!atual) {
```

```
                    System.out.println(ator);
                    existe = true;
                }
            }
            if (!existe && atual) {
                System.out.println("Não existem atores que participem
num filme desta edição.");
            } else if (!existe && !atual) {
                System.out.println("Não existem atores no festival.");
            }
        }

        /**
         * Método que permite escolher os candidatos
         */
        private void escolherCandidatos() {
            Premio premioEscolhido = escolherPremio();
            switch (premioEscolhido.getNome()) { //verifica o nome do
premio escolhido pelo utilizador
                case "Melhor Ator Principal":

escolherAtoresPrincipaisCandidatos(premioEscolhido, true);
                    break;
                case "Melhor Atriz Principal":

escolherAtoresPrincipaisCandidatos(premioEscolhido, false);
                    break;
                case "Melhor Ator Secundário":

escolherAtoresSecundariosCandidatos(premioEscolhido, true);
                    break;
                case "Melhor Atriz Secundária":

escolherAtoresSecundariosCandidatos(premioEscolhido, false);
                    break;
                case "Melhor Realizador":
                    escolherRealizadorCandidatos(premioEscolhido);
                    break;
                case "Prémio Carreira":
                    escolherPremioCarreira(premioEscolhido);
                    break;
                default: //se o prémio for algum filme
                    escolherFilmesCandidatos(premioEscolhido);
                    break;
            }
        }

        /**
         * Método que trata da escolha do prémio que o utilizador
pretende
         * ver/atribuir candidatos, etc
         *
         * @return Premio selecionado pelo utilizador
         */
        private Premio escolherPremio() {
            System.out.println("Escolha o prémio:");
            System.out.print("(1) Melhor Ator Principal\n" + "(2)
Melhor Atriz Principal\n"
                    + "(3) Melhor Ator Secundário\n" + "(4) Melhor Atriz
Secundária\n"
                    + "(5) Melhor Filme\n"
```

```
                    + "(6) Melhor Realizador\n"
                    + "(7) Melhor Argumento\n"
                    + "(8) Melhor Cinematografia\n"
                    + "(9) Prémio Carreira"
                    + "\nOpção: ");
            while (true) { //até não escolher uma opção válida
                try {
                    int opcaoPremio = recebeInteiro();
                    return edicoes.get(numEdicao - 1).getPremios().get(opcaoPremio - 1);
                } catch (IndexOutOfBoundsException e) {
                    System.out.print("Escolha uma opção válida.\nOpção: ");
                }
            }
        }

        /**
         * Método que permite o utilizador escolher os filmes candidatos para um
         * dado prémio
         *
         * @param p - prémio ao qual pretende-se nomear filmes
         */
        private void escolherFilmesCandidatos(Premio p) {
            if (p.getFilmesCandidatos().size() == 4) { //se já foram escolhidos os candidatos
                System.out.println("Os candidatos para este prémio já foram escolhidos.\n");
                return;
            }
            ArrayList<Filme> possiveisCandidatos = new ArrayList<>(); //lista com os candidatos possiveis
            int contaCandidatosPossiveis = 0;
            for (Filme filme : edicoes.get(numEdicao - 1).getFilmes()) {
                possiveisCandidatos.add(filme);
                contaCandidatosPossiveis++;
            }
            if (contaCandidatosPossiveis < 4) { //se não há pelo menos 4 candidatos
                System.out.println("\nNão há candidatos suficientes para esta categoria.");
                return; //não continua
            }
            int indiceCandidato;
            while (p.getFilmesCandidatos().size() < 4) { //Obriga o utilizador a escolher os 4 candidatos de uma só vez
                for (int j = 0; j < possiveisCandidatos.size(); j++) {
                    System.out.printf("%d. %s\n", j + 1, possiveisCandidatos.get(j).getNome());
                }
                System.out.print("Indique o filme candidato: ");
                String nome = scan.nextLine().trim();
                limparConsola();
                try {
                    indiceCandidato = indexOfByFilmName(nome, possiveisCandidatos); //indice do filme candidato, na lista dos possiveis candidatos
                    Filme candidato = possiveisCandidatos.get(indiceCandidato);
```

```
                    p.nomeiaFilme(candidato);    //insere    o    filme
candidato no prémio em questão

possiveisCandidatos.remove(possiveisCandidatos.get(indiceCandidato));
//remove o filme escolhido, da lista dos candidatos. assim, não pode ser
escolhido novamente
                } catch (IndexOutOfBoundsException e) {
                    System.out.println("Por  favor,  indique  um  nome
válido.");
                }
            }
        }

        /**
         * Método  que  permite  o  utilizador  escolher  os  melhores
realizadores
         *
         * @param p - prémio ao qual pretende-se nomear realizadores
         */
        private void escolherRealizadorCandidatos(Premio p) {
            if (p.getFilmesCandidatos().size() != 4) { //se ainda não
foram escolhidos os candidatos
                ArrayList<Realizador>   possiveisCandidatos   =   new
ArrayList<>(); //lista com os realizadores candidatos
                ArrayList<Filme>   filmesPossiveisCandidatos   =   new
ArrayList<>(); //lista com os filmes dos realizadores candidatos
                int contaCandidatosPossiveis = 0;
                for    (Filme    filme    :    edicoes.get(numEdicao    -
1).getFilmes()) {
                    if
(!possiveisCandidatos.contains(filme.getRealizador()))    {    //se    o
realizador participa em 2 ou mais filmes, só conta como 1 candidato
                        contaCandidatosPossiveis++;
                    }
                    possiveisCandidatos.add(filme.getRealizador());
                    filmesPossiveisCandidatos.add(filme);
                }
                if (contaCandidatosPossiveis < 4) { //se não há pelo
menos 4 candidatos
                    System.out.println("\nNão      há      candidatos
suficientes para esta categoria.");
                    return;
                }
                int indiceCandidato;
                int indiceFilmeCandidato;
                while (p.getFilmesCandidatos().size() < 4) { //Obriga
o utilizador a escolher os 4 candidatos de uma só vez
                    for (int j = 0; j < possiveisCandidatos.size();
j++) {
                        System.out.printf("%d.  %s  por  %s\n", j + 1,
possiveisCandidatos.get(j).getNome(),
filmesPossiveisCandidatos.get(j).getNome());
                    }
                    System.out.print("Indique  o  nome  do  realizador
nomeado: ");
                    String nome = scan.nextLine().trim();
                    try {
                        for (indiceCandidato = 0; indiceCandidato <
possiveisCandidatos.size(); indiceCandidato++) { //procura o indice do
realizador escolhido
                            if
```

```
(nome.equalsIgnoreCase(possiveisCandidatos.get(indiceCandidato).getNom
e())) {
                                    break; //para no indice do realizador
escolhido
                                }
                            }
                            Realizador             candidato             =
possiveisCandidatos.get(indiceCandidato);
                            if             (indiceCandidato             ==
possiveisCandidatos.lastIndexOf(candidato)) { //se o realizador apenas
aparece 1 vez na lista de possiveis candidatos
                                limparConsola();

p.nomeiaFilme(filmesPossiveisCandidatos.get(indiceCandidato));
//adiciona filme do realizador

possiveisCandidatos.remove(indiceCandidato);  //remove  o  realizador
escolhido dos possiveis candidatos

filmesPossiveisCandidatos.remove(indiceCandidato); //remove o filme do
realizador escolhido dos possiveis candidatos
                            } else {
                                System.out.println("Esse        realizador
direcionou 2 ou mais filmes, a qual se refere?");
                                String nomeFilme = scan.nextLine().trim();
                                limparConsola();
                                indiceFilmeCandidato                    =
indexOfByFilmName(nomeFilme, filmesPossiveisCandidatos);
                                Filme                filme                =
filmesPossiveisCandidatos.get(indiceFilmeCandidato);
                                if (filme.getRealizador() == candidato) {
                                    p.nomeiaFilme(filme); //insere o filme
do realizador escolhido no prémio em questão

possiveisCandidatos.remove(indiceFilmeCandidato); //remove o realizador
escolhido dos possiveis candidatos

filmesPossiveisCandidatos.remove(indiceFilmeCandidato);   //remove   o
filme do realizador escolhido dos possiveis candidatos
                                    while
(possiveisCandidatos.contains(candidato)) { //até o realizador não estar
na lista dos possiveis candidatos
                                        indiceCandidato                 =
possiveisCandidatos.indexOf(candidato);

possiveisCandidatos.remove(candidato); //remove todas as ocorrências do
realizador escolhido na lista dos possiveis candidatos

filmesPossiveisCandidatos.remove(indiceCandidato);//remove  o  filme  do
realizador escolhido dos possiveis candidatos
                                    }
                                }
                            }
                        } catch (IndexOutOfBoundsException e) {
                            System.out.println("Por favor, indique um nome
válido.");
                        }
                    }
                } else {
                    System.out.println("Os candidatos para este prémio já
foram escolhidos.\n");
```

```
            }
        }

        /**
         * Método que permite o utilizador escolher os atores (ou as
atrizes)
         * principais
         *
         * @param p - prémio ao qual se pretende nomear atores
         * @param homem - indica se é para escolher atores principais
ou atrizes
         * principais (mesma lógica que em Ator, true-Ator, false-Atriz)
         */
        private void escolherAtoresPrincipaisCandidatos(Premio p,
boolean homem) {
            if (p.getAtoresCandidatos().size() != 4) { //se ainda não
foram escolhidos os candidatos
                ArrayList<Ator> possiveisCandidatos = new
ArrayList<>(); //lista com os possiveis atores candidatos
                ArrayList<Filme> filmesPossiveisCandidatos = new
ArrayList<>(); //lista com os filmes dos possiveis atores candidatos
                int contaCandidatosPossiveis = 0;
                if (homem) {
                    for (Filme filme : edicoes.get(numEdicao -
1).getFilmes()) {
                        if (filme.getAtorPrincipal() != null) {
                            if
(!possiveisCandidatos.contains(filme.getAtorPrincipal())) { //se o ator
aparece em mais do que 1 filme, como ator principal, só conta como 1
candidato
                                contaCandidatosPossiveis++;
                            }

possiveisCandidatos.add(filme.getAtorPrincipal());
                            filmesPossiveisCandidatos.add(filme);
                        }
                    }
                } else {
                    for (Filme filme : edicoes.get(numEdicao -
1).getFilmes()) {
                        if (filme.getAtrizPrincipal() != null) {
                            if
(!possiveisCandidatos.contains(filme.getAtrizPrincipal())) {//se a
atriz aparece em mais do que 1 filme, como atriz principal, só conta
como 1 candidat0
                                contaCandidatosPossiveis++;
                            }

possiveisCandidatos.add(filme.getAtrizPrincipal());
                            filmesPossiveisCandidatos.add(filme);
                        }
                    }
                }
                if (contaCandidatosPossiveis < 4) { //se há menos de 4
candidatos diferentes
                    System.out.println("\nNão há candidatos
suficientes para esta categoria.");
                    return;
                }
                int indiceCandidato;
                int indiceFilmeCandidato;
```

```
                while (p.getAtoresCandidatos().size() < 4) { //Obriga
o utilizador a escolher os 4 candidatos de uma só vez
                    for (int j = 0; j < possiveisCandidatos.size();
j++) {
                        System.out.printf("%d. %s por %s\n", j + 1,
possiveisCandidatos.get(j).getNome(),
filmesPossiveisCandidatos.get(j).getNome());
                    }
                    System.out.printf("Indique o nome do %s candidato:
", (homem ? "ator" : "atriz"));
                    String nome = scan.nextLine().trim();
                    try {
                        indiceCandidato = indexOfByActorName(nome,
possiveisCandidatos); //posição do ator/atriz candidato na lista dos
possiveis candidatos
                        Ator candidato =
possiveisCandidatos.get(indiceCandidato); //ator/atriz correspondente
ao indice
                        if (indiceCandidato ==
possiveisCandidatos.lastIndexOf(candidato)) { //se o ator/atriz só
aparece uma vez na lista de candidatos (participa apenas num filme como
principal)
                            limparConsola();
                            p.nomeiaAtor(candidato,
filmesPossiveisCandidatos.get(indiceCandidato)); //insere o ator/atriz
na lista de candidatos do prémio

possiveisCandidatos.remove(indiceCandidato); //remove o ator/atriz da
lista de candidatos possiveis

filmesPossiveisCandidatos.remove(indiceCandidato); //remove o filme do
ator/atriz da lista de candidatos possiveis
                        } else {
                            System.out.println((homem ? "Esse ator" :
"Essa atriz") + " participa em 2 filmes, a qual se refere?");
                            String nomeFilme = scan.nextLine().trim();
                            limparConsola();
                            try {
                                indiceFilmeCandidato =
indexOfByFilmName(nomeFilme, filmesPossiveisCandidatos); //posicao do
filme do ator/atriz candidato na lista de filmes candidatos
                                Filme filme =
filmesPossiveisCandidatos.get(indiceFilmeCandidato); //filme
correspondente ao indice
                                if (homem) { //se for ator principal
                                    if (filme.getAtorPrincipal() ==
candidato) { //se o ator principal desse filme for o ator principal
escolhido
                                        p.nomeiaAtor(candidato,
filme); //coloca o ator nos candidatos do prémio

possiveisCandidatos.remove(indiceFilmeCandidato); //remove o ator da
lista dos possiveis candidatos (do filme escolhido)
                                        indiceCandidato =
possiveisCandidatos.indexOf(candidato); //indice do ator da lista dos
possiveis candidatos

possiveisCandidatos.remove(candidato); //remove o ator da lista dos
possiveis candidatos

filmesPossiveisCandidatos.remove(indiceFilmeCandidato);   //remove   o
```

```
filme que foi escolhido do ator da lista de possiveis candidatos

filmesPossiveisCandidatos.remove(indiceCandidato); //remove o filme que
não foi escolhido
                                    } else {
                                        System.out.println("Esse  ator
não participa no filme que inseriu.\n");
                                    }
                                } else {
                                    if  (filme.getAtrizPrincipal()  ==
candidato) { //se a atriz principal desse filme for o ator principal
escolhido
                                        p.nomeiaAtor(candidato,
filme); //coloca a atriz nos candidatos do prémio

possiveisCandidatos.remove(indiceFilmeCandidato); //remove a atriz da
lista dos possiveis candidatos (do filme escolhido)
                                        indiceCandidato              =
possiveisCandidatos.indexOf(candidato); //indice da atriz da lista dos
possiveis candidatos

possiveisCandidatos.remove(candidato); //remove a atriz da lista dos
possiveis candidatos

filmesPossiveisCandidatos.remove(indiceFilmeCandidato);    //remove  o
filme que foi escolhido da atriz da lista de possiveis candidatos

filmesPossiveisCandidatos.remove(indiceCandidato); //remove o filme que
não foi escolhido
                                    } else {
                                        System.out.println("Essa atriz
não participa no filme que inseriu.\n");
                                    }
                                }
                            } catch (IndexOutOfBoundsException e) {
                                System.out.println("Escreveu   mal   o
nome do filme. Tente outra vez.");
                            }
                        }
                    } catch (IndexOutOfBoundsException e) {
                        System.out.printf("Por     favor,     escolha
%s.\n\n", (homem ? "um ator válido" : "uma atriz válida"));
                    }
                }
            } else {
                System.out.println("Os candidatos para este prémio já
foram escolhidos.\n");
            }
        }

        /**
         * Método que permite o utilizador escolher os atores (ou as
atrizes)
         * secundários
         *
         * @param p - prémio ao qual se pretende nomear atores
         * @param homem - indica se é para escolher atores secundários
ou atrizes
         * secundárias (mesma lógica que em Ator, true-Ator, false-
Atriz)
         */
```

```
        private void escolherAtoresSecundariosCandidatos(Premio p,
boolean homem) {
            if (p.getAtoresCandidatos().size() != 4) { //se ainda não
foram escolhidos os candidatos
                ArrayList<Ator> possiveisCandidatos = new
ArrayList<>(); //lista com os possiveis atores candidatos
                ArrayList<Filme> filmesPossiveisCandidatos = new
ArrayList<>(); //lista com os filmes dos possiveis atores candidatos
                int contaCandidatosPossiveis = 0;
                for (Filme filme : edicoes.get(numEdicao -
1).getFilmes()) {
                    for (Ator a : filme.getAtoresSecundarios()) {
                        if (a.getGenero() == homem) {
                            if (!possiveisCandidatos.contains(a)) {
//se o ator aparece em mais do que 1 filme, só conta como 1 candidato
                                contaCandidatosPossiveis++;
                            }
                            possiveisCandidatos.add(a);
                            filmesPossiveisCandidatos.add(filme);
                        }
                    }
                }
                if (contaCandidatosPossiveis < 4) { //se há menos de 4
candidatos diferentes
                    System.out.println("\nNão há candidatos
suficientes para esta categoria.");
                    return;
                }
                int indiceCandidato;
                int indiceFilmeCandidato;
                while (p.getAtoresCandidatos().size() < 4) { //Obriga
o utilizador a escolher os 4 candidatos de uma só vez
                    for (int j = 0; j < possiveisCandidatos.size();
j++) {
                        System.out.printf("%d. %s por %s\n", j + 1,
possiveisCandidatos.get(j).getNome(),
filmesPossiveisCandidatos.get(j).getNome());
                    }
                    System.out.println("Indique o nome " + (homem ? "
do ator candidato: " : "da atriz candidata: "));
                    String nome = scan.nextLine().trim();
                    try {
                        indiceCandidato = indexOfByActorName(nome,
possiveisCandidatos); //posicao do filme do ator candidato na lista de
filmes candidatos
                        Ator candidato =
possiveisCandidatos.get(indiceCandidato); //ator correspondente ao
indice
                        if (indiceCandidato ==
possiveisCandidatos.lastIndexOf(candidato)) { //se o ator/atriz só
participa num filme
                            limparConsola();
                            p.nomeiaAtor(candidato,
filmesPossiveisCandidatos.get(indiceCandidato));

possiveisCandidatos.remove(indiceCandidato); //remove o ator dos
possiveis candidatos

filmesPossiveisCandidatos.remove(indiceCandidato); //remove o filme do
ator dos filmes candidatos
                        } else {
```

```
                            System.out.println((homem ? "Esse ator" :
"Essa atriz") + " participa em 2 filmes, a qual se refere?");
                            String nomeFilme = scan.nextLine().trim();
                            limparConsola();
                            try {
                                indiceFilmeCandidato                =
indexOfByFilmName(nomeFilme,  filmesPossiveisCandidatos); //posicao  do
filme escolhido
                                Filme              filme              =
filmesPossiveisCandidatos.get(indiceFilmeCandidato);  //filme escolhido
                                if
(filme.getAtoresSecundarios().contains(candidato))   {   //se   o   ator
escolhido pertence a esse filme
                                    p.nomeiaAtor(candidato,    filme);
//insere na lista de candidatos do premio o ator escolhido

possiveisCandidatos.remove(indiceFilmeCandidato); //remove  o  ator  dos
possiveis candidatos
                                    indiceCandidato                    =
possiveisCandidatos.indexOf(candidato);

possiveisCandidatos.remove(candidato); //remove  a  outra  ocorrencia  do
ator nos possiveis candidatos (o outro filme)

filmesPossiveisCandidatos.remove(indiceFilmeCandidato);   //remove   o
filme escolhido

filmesPossiveisCandidatos.remove(indiceCandidato);  //remove  o  outro
filme do ator
                                } else {
                                    System.out.println((homem  ?  "Esse
ator" : "Essa atriz") + " não participa no filme que inseriu.\n");
                                }
                            } catch (IndexOutOfBoundsException e) {
                                System.out.println("Escreveu   mal   o
nome do filme. Tente outra vez.");
                            }
                        }
                    } catch (IndexOutOfBoundsException e) {
                        System.out.printf("Por  favor,  escolha  %s.\n",
(homem ? "um ator válido" : "uma atriz válida"));
                    }
                }
            } else {
                System.out.println("Os candidatos para este prémio já
foram escolhidos.\n");
            }
        }

        /**
         * Método que permite ao utilizador escolher os nomeados ao
Prémio Carreira
         *
         * @param p - prémio ao qual se pretende nomear os atores
         */
        private void escolherPremioCarreira(Premio p) {
            if (p.getAtoresCandidatos().size() != 4) { //se ainda não
foram escolhidos os candidatos
                ArrayList<Ator>     possiveisCandidatos     =     new
ArrayList<>();
                int contaCandidatosPossiveis = 0;
```

```
                for (int posição1 = 0; posição1 < atores.size();
posição1++) {
                    if (atores.get(posição1).getAnosCarreira() > 20) {
//se o ator tem mais de 20 anos de carreira
                        possiveisCandidatos.add(atores.get(posição1));
                        contaCandidatosPossiveis++;
                    }
                }
                int indiceCandidato;
                if (contaCandidatosPossiveis < 4) { //se há pelo menos
4 candidatos
                    System.out.println("\nNão há candidatos
suficientes para esta categoria.");
                    return;
                }
                while (p.getAtoresCandidatos().size() < 4) { //Obriga
o utilizador a escolher os 4 candidatos de uma só vez
                    for (int j = 0; j < possiveisCandidatos.size();
j++) {
                        System.out.printf("%d. %s\n", j + 1,
possiveisCandidatos.get(j).getNome());
                    }
                    System.out.print("Indique o nome do candidato: ");
                    String escolhido = scan.nextLine().trim();
                    limparConsola();
                    try {
                        indiceCandidato =
indexOfByActorName(escolhido, possiveisCandidatos); //posicao do ator
candidato na lista de candidatos
                        Ator candidato =
possiveisCandidatos.get(indiceCandidato); //ator correspondente
                        p.nomeiaAtor(candidato); //insere o ator nos
candidatos do prémio
                        possiveisCandidatos.remove(indiceCandidato);
//remove o ator dos possiveis candidatos
                    } catch (IndexOutOfBoundsException e) {
                        System.out.println("Por favor, escolha um
candidato válido.");
                    }
                }
            } else {
                System.out.println("Os candidatos para este prémio já
foram escolhidos.\n");
            }
        }

        /**
         * Método que permite pontuar os candidatos do Prémio passado
como paramêtro
         *
         * @param premio - Prémio no qual se pretende pontuar os
candidatos
         */
        private void pontuarCandidatos(Premio premio) {
            System.out.println("AVALIAÇÃO DO PRÉMIO: " +
premio.toString().toUpperCase());
            if (premio.getNome().contains("Carreira")) {
                if (premio.getMediasPontuacoes()[0] != 0 &&
!Double.isNaN(premio.getMediasPontuacoes()[0])) { //verificar se o
prémio carreira já foi pontuado (pois este não atualiza a variável
vencedor do prémio)
```

```
                    System.out.println("Os candidatos já foram
pontuados para esta categoria.\n");
                    return;
                }
            }
            if (premio.getVencedor() == null) { //se não for o prémio
carreira, verifica-se se o vencedor é diferente de nulo, o que significa
que o prémio ainda não foi pontuado
                boolean pontuou = false; //se esta variável permanecer
falsa significa que não há peritos criados para avaliar os prémios
                try {
                    if (premio.getNome().contains("Ator") ||
premio.getNome().contains("Atriz") ||
premio.getNome().contains("Carreira")) {//verificar se é um prémio para
atores
                        for (int indiceCandidato = 0; indiceCandidato
< 4; indiceCandidato++) { //percorrer a lista de candidatos
                            if (premio.getNome().contains("Carreira"))
{
                                System.out.printf("\nCANDIDATO %d:
%s\n", indiceCandidato + 1,
premio.getAtoresCandidatos().get(indiceCandidato).getNome());
                            } else {
                                System.out.printf("\nCANDIDATO %d: %s
em %s\n", indiceCandidato + 1,
premio.getAtoresCandidatos().get(indiceCandidato).getNome(),
premio.getFilmesCandidatos().get(indiceCandidato).getNome());
                            }
                            for (Perito p : edicoes.get(numEdicao -
1).getPeritos()) { //percorrer a lista de peritos para estes pontuarem
o candidato em questão
                                pontuou = true; //se entra dentro deste
ciclo então há peritos e atualiza-se esta variável para true
                                while (!p.inserePontuacao(premio,
indiceCandidato, edicoes.get(numEdicao - 1).getPeritos().indexOf(p),
scan)) { //se esta função retornar false quer dizer que o valor foi
inválido
                                    System.out.println("O valor
precisa de ser entre 1 e 10 (inclusive) e inteiro!");
                                }//quando a função inserePontuacao
retornar true significa que a pontuação foi válida
                            }
                        }
                    } else {
                        for (int indiceCandidato = 0; indiceCandidato
< 4; indiceCandidato++) { //percorrer a lista de candidatos
                            if
(premio.getNome().contains("Realizador")) { //verificar se é o prémio
de Melhor Realizador
                                System.out.printf("\nCANDIDATO %d: %s
por %s\n", indiceCandidato + 1,
premio.getFilmesCandidatos().get(indiceCandidato).getRealizador().getN
ome(), premio.getFilmesCandidatos().get(indiceCandidato).getNome());
                            } else {
                                System.out.printf("\nCANDIDATO %d:
%s\n", indiceCandidato + 1,
premio.getFilmesCandidatos().get(indiceCandidato).getNome());
                            }
                            for (Perito p : edicoes.get(numEdicao -
1).getPeritos()) { //percorrer a lista de peritos para pontuarem os
candidatos
```

```
                                pontuou = true;//se entra dentro deste
ciclo então há peritos e atualiza-se esta variável para true
                                while (!p.inserePontuacao(premio,
indiceCandidato, edicoes.get(numEdicao - 1).getPeritos().indexOf(p),
scan)) {//se esta função retornar false quer dizer que o valor foi
inválido
                                    System.out.println("O valor
precisa de ser entre 1 e 10 (inclusive) e inteiro!");
                                }//quando a função inserePontuacao
retornar true significa que a pontuação foi válida
                            }
                        }
                    }
                    limparConsola();
                    if (!pontuou) { //se pontuou é false, como já foi
dito, não há peritos para pontuar o prémio e informa-se o utilizador
                        System.out.println("Não há peritos para pontuar
esta categoria.\n");
                        return;
                    }
                    premio.ordenaPontuações();    //agora que as
pontuações foram dadas ao prémio chama-se esta função para calcular as
médias e verificar quem venceu, etc.
                } catch (IndexOutOfBoundsException e) {
                    System.out.println("Não há candidatos para esta
categoria.\n");
                }
            } else { //se o prémio já tem vencedor então os candidatos
já foram pontuados
                System.out.println("Os candidatos já foram pontuados
para esta categoria.\n");
            }
        }

        /**
         * Método que permite encontrar o índice de um Ator numa lista
de atores
         * pelo nome
         *
         * @param nome - nome do Ator a pesquisar na lista
         * @param atores - lista na qual procurar o ator
         * @return o índice do ator na lista, se o ator não está na
lista retorna-se
         * -1
         */
        private int indexOfByActorName(String nome, ArrayList<Ator>
atores) {
            for (Ator a : atores) {
                if (nome.equalsIgnoreCase(a.getNome())) {
                    return atores.indexOf(a);
                }
            }
            return -1;
        }

        /**
         * Método que pesquisa a lista de filmes pelo filme especificado
pelo
         * parametro nome
         *
         * @param nome - nome do filme que pretende-se encontrar
```

```
     * @param filmes - lista de filmes em que procurar
     * @return o índice do filme, se foi encontrado. Caso contrário,
retorna -1
     */
    private int indexOfByFilmName(String nome, ArrayList<Filme>
filmes) {
        for (Filme f : filmes) {
            if (nome.equalsIgnoreCase(f.getNome())) {
                return filmes.indexOf(f);
            }
        }
        return -1;
    }

    /**
     * Método que trata de receber inteiros e a possível exceção
associada
     *
     * @return o inteiro que o utilizador inseriu
     */
    private int recebeInteiro() {
        String aux;
        int num;
        while (true) {//não sai do while enquanto o utilizador não
inserir um número
            aux = scan.nextLine().trim();
            try {
                num = Integer.parseInt(aux);
                break;//só chega aqui se o parseInt não atirar a
NumberFormatException
            } catch (NumberFormatException e) {//entra aqui se não
é inserido um número
                System.out.println("Insira um número inteiro!");
            }
        }
        return num;
    }

    /**
     * Método responsável pelo menu de inicio após a criação de uma
edição
     */
    private void novoOuCarregar() {
        while (!quebra) {
            System.out.print("(n): Começar uma nova edição\n(c):
Carregar dados\nOpção: ");
            opcao = scan.nextLine().trim().toLowerCase();
            switch (opcao) {
                case "c":
                    quebra = true;
                    numEdicao++;
                    if (numEdicao > 1) { //para avançar para o
próximo ano (caso não seja a primeira edição)
                        ano++;
                    }
                    edicoes.add(new Edicao(numEdicao, ano));
                    System.out.print("Opções:\n(a): Carregar
Atores e Filmes\n(c): Carregar Atores, Filmes e Candidatos\n(p):
Carregar Tudo (sem pontuações)\n(t): Carregar Tudo (com
pontuações)\nOpção: ");
                    opcao = scan.nextLine().trim().toLowerCase();
```

```
                        try {
                            switch (opcao.toLowerCase()) {
                                case "a":
                                    carregaFilmes();
                                    carregaAtores();
                                    limparConsola();

                                    break;
                                case "c":
                                    carregaFilmes();
                                    carregaAtores();
                                    carregaCandidatos();
                                    limparConsola();

                                    break;
                                case "p":
                                    carregaFilmes();
                                    carregaAtores();
                                    carregaPeritos();
                                    carregaCandidatos();
                                    limparConsola();

                                    break;
                                case "t":
                                    carregaFilmes();
                                    carregaAtores();
                                    carregaPeritos();
                                    carregaCandidatos();
                                    carregaPontuacoes();
                                    for (Premio p : edicoes.get(numEdicao - 1).getPremios()) { //faz-se este ciclo para sabermos logo no início
                                        //do programa os filmes mais premiados, já que todos os dados, incluindo nomeados e as suas pontuações, foram carregados
                                        p.ordenaPontuações();
                                    }
                                    limparConsola();

                                    break;
                                default:
                                    quebra = false;
                                    edicoes.remove(numEdicao - 1); //apaga-se a edição criada pois o utilizador inseriu uma opção inválida
                                    numEdicao--;//decrementar o número de edição por causa da opção inválida
                                    if (numEdicao > 1) {
                                        ano--;
                                    }
                                    System.out.println("Opção inválida.");
                            }
                        } catch (IOException e) {
                            System.out.println("Erro ao carregar os dados.");
                        }
                        break;
                    case "n":
                        numEdicao++;
                        if (numEdicao > 1) {
                            ano++;
```

```
                        }
                        edicoes.add(new Edicao(numEdicao, ano));
                        quebra = true;
                        limparConsola();
                        break;
                    default:
                        System.out.println("Opção inválida.");
                }
            }
        }

        //-------------------------------------------------------------
-------------------------------
        /**
         * Método que permite carregar os atores de um ficheiro
         */
        private void carregaAtores() throws IOException {
            Ator ator;
            String nomeAtor;
            boolean generoAtor;
            int anosCarreiraAtor;
            boolean cria;
            String line;
            int indexFilmes = 0;
            FileReader inStream = new FileReader("Edicao" + numEdicao
+ "\\Atores.txt");
            BufferedReader lerDados = new BufferedReader(inStream);
            line = lerDados.readLine();
            while (line != null) {
                cria = true;
                switch (line) {
                    case "Ator principal:":
                        nomeAtor = lerDados.readLine().trim();//ler a
linha para saber o nome do ator
                        generoAtor                                    =
lerDados.readLine().trim().equals("M"); //ler a linha para saber o
género do ator ator
                        anosCarreiraAtor                              =
Integer.parseInt(lerDados.readLine().trim());//ler a linha para os anos
de carreira do ator ator
                        if (!nomeAtor.equals("VAZIO")) { //se quando
foi gravado o filme não tinha ator principal então no nome estará vazio
                            ator = new Ator(nomeAtor, generoAtor,
anosCarreiraAtor);
                            for (Ator a : this.atores) {//percorremos
a lista de atores para ver se o ator já foi criado
                                if (ator.equals(a)) { //se foi,
inserimos no filme o ator previamente criado e não se cria um novo
                                    ator = a;
                                    cria = false;
                                    break;
                                }
                            }
                            if (cria) { //se o ator já foi criado, não
se cria de novo
                                atores.add(ator);
                            }
                            edicoes.get(numEdicao                     -
1).getFilmes().get(indexFilmes).insereAtor(ator, true); //insere-se o
ator no filme
                        }
```

```
                        break;
                    case "Atriz principal:":
                        nomeAtor = lerDados.readLine().trim();//ler a
linha para saber o nome do atriz
                        generoAtor                                  =
lerDados.readLine().trim().equals("M");//ler a linha para saber o género
do ator atriz
                        anosCarreiraAtor                            =
Integer.parseInt(lerDados.readLine().trim());//ler a linha para os anos
de carreira da atriz
                        if (!nomeAtor.equals("VAZIO")) {//se quando foi
gravado o filme não tinha atriz principal então no nome estará vazio
                            ator  =  new  Ator(nomeAtor,  generoAtor,
anosCarreiraAtor);
                            for (Ator a : this.atores) {//percorremos
a lista de atores para ver se a atriz já foi criada
                                if    (ator.equals(a))    {//se    foi,
inserimos no filme a atriz previamente criada e não se cria uma nova
                                    ator = a;
                                    cria = false;
                                    break;
                                }
                            }
                            if (cria) {//se a atriz já foi criada, não
se cria de novo
                                atores.add(ator);
                            }
                            edicoes.get(numEdicao                       -
1).getFilmes().get(indexFilmes).insereAtor(ator,  true);//insere-se  o
ator no filme
                        }
                        break;
                    case "Atores Secundarios:":
                        while (true) {
                            nomeAtor = lerDados.readLine(); //lê-se a
linha, para ver se é o nome de um ator ou não
                            if      (nomeAtor      !=      null      &&
!nomeAtor.contains("----------")) { //verificamos se ou já chegamos ao
fim do ficheiro
                                //ou  ao  fim  da  lista  de  atores
secundários do filme
                                nomeAtor = nomeAtor.trim();
                                generoAtor                              =
lerDados.readLine().trim().equals("M");//ler a linha para saber o género
do ator atriz
                                anosCarreiraAtor                        =
Integer.parseInt(lerDados.readLine().trim());//ler a linha para os anos
de carreira do ator
                                ator = new Ator(nomeAtor, generoAtor,
anosCarreiraAtor);
                                for    (Ator    a    :    this.atores)
{//percorremos a lista de atores para ver se o ator já foi criado
                                    if  (ator.equals(a))  {//se  foi,
inserimos no filme o ator previamente criado e não se cria um novo
                                        ator = a;
                                        cria = false;
                                        break;
                                    }
                                }
                                if (cria) {//se o ator já foi criado,
não se cria de novo
```

```
                                    atores.add(ator);
                                }
                                edicoes.get(numEdicao - 1).getFilmes().get(indexFilmes).insereAtor(ator, false);
                            } else { //se a linha lida (em nomeAtor) é igual ao separador, aumenta-se o índice para guardar o próximo filme
                                indexFilmes++;
                                break; //
                            }
                        }
                    default:
                        break;
                }
                line = lerDados.readLine();
            }
            lerDados.close();
        }

        /**
         * Método que permite carregar os filmes de um ficheiro
         */
        private void carregaFilmes() throws IOException {
            String nomeFilme;
            String generoFilme;
            String nomeRealizador;
            boolean generoRealizador;
            Realizador realizador;
            String line;
            FileReader inStream = new FileReader("Edicao" + numEdicao + "\\Filmes.txt");
            BufferedReader lerDados = new BufferedReader(inStream);
            line = lerDados.readLine(); //ler a primeira linha, que é, se houver, o nome do primeiro filme
            while (line != null) {
                nomeFilme = line.trim();
                generoFilme = lerDados.readLine().trim(); //ler a linha que indica o género do filme
                nomeRealizador = lerDados.readLine().trim(); //ler a linha que indica o nome do realizador
                generoRealizador = lerDados.readLine().trim().equals("M"); //ler a linha que indica o género do realizador
                realizador = new Realizador(nomeRealizador, generoRealizador);
                for (int i = 0; i < numEdicao; i++) {
                    for (Filme f : edicoes.get(i).getFilmes()) {//percorremos a lista de filmes para ver se o realizador já foi criado
                        if (f.getRealizador().equals(realizador)) {//se foi, inserimos no filme realizador previamente criado e não um novo
                            realizador = f.getRealizador();
                            break;
                        }
                    }
                }
                Filme filme = new Filme(nomeFilme, generoFilme, realizador); //cria-se o filme
                edicoes.get(numEdicao - 1).insereFilmes(filme); //e acrescenta-se o filme à lista de filmes da edição que está a ser carregada
                line = lerDados.readLine();
            }
```

```
            lerDados.close();
        }

        /**
         * Método que permite carregar os peritos de um ficheiro
         */
        private void carregaPeritos() throws IOException {
            String nomePerito;
            boolean generoPerito;
            FileReader inStream = new FileReader("Edicao" + numEdicao
+ "\\Peritos.txt");
            BufferedReader lerDados = new BufferedReader(inStream);
            String line;
            line = lerDados.readLine();//ler a primeira linha, que é,
se houver, o nome do primeiro perito
            while (line != null) {
                nomePerito = line.trim();
                generoPerito = lerDados.readLine().trim().equals("M");
//ler a linha que indica o género do perito
                Perito perito = new Perito(nomePerito, generoPerito);
//criar o perito
                edicoes.get(numEdicao - 1).inserePerito(perito); //e
acrescentá-lo na lista de peritos da edição que está a ser carregada
                line = lerDados.readLine();
            }
            lerDados.close();

        }

        /**
         * Método que permite carregar os candidatos de um ficheiro
         */
        private void carregaCandidatos() throws IOException {
            String nomeAtor;
            String filme;
            Premio premio;
            boolean vazio;
            int indexPremios = 0;
            FileReader inStream = new FileReader("Edicao" + numEdicao
+ "\\Candidatos.txt");
            BufferedReader lerDados = new BufferedReader(inStream);
            String line;
            line = lerDados.readLine().trim();
            while (line != null && indexPremios < 9) {
                vazio = false;
                premio = edicoes.get(numEdicao -
1).getPremios().get(indexPremios); //guardar em premio o prémio dos
quais estamos a carregar os candidatos
                if (indexPremios < 4) {
                    while (premio.getAtoresCandidatos().size() < 4) {
                        nomeAtor = lerDados.readLine(); //lê-mos a
linha, que ou é o nome do ator ou o separador, se não há candidatos para
ler
                        if (nomeAtor == null || nomeAtor.contains("---
-------")) { //se não há candidatos para ler, ou já se chegou ao fim do
ficheiro, saímos do while
                            vazio = true;
                            break;
                        }
                        nomeAtor = nomeAtor.trim();
                        filme = lerDados.readLine().trim(); //ler a
```

```
linha que indica o nome do filme pelo qual o ator/atriz foi nomeado
                        for (Ator a : this.atores) { //percorrer a lista
de atores para encontrar o ator
                            if (a.getNome().equals(nomeAtor)) { //se o
nome do ator for igual ao ator na lista, nomeia-se o ator e encontra-se
o filme na lista de filmes em que o ator participa
                                premio.nomeiaAtor(a,
a.getFilmes().get(indexOfByFilmName(filme, a.getFilmes())));
                                break;
                            }
                        }
                    }
                } else if (indexPremios > 3 && indexPremios < 8) {
                    while (premio.getFilmesCandidatos().size() < 4) {
                        filme = lerDados.readLine();//lê-mos a linha,
que ou é o nome do filme ou o separador, se não há candidatos para ler
                        if (filme == null || filme.contains("---------
-")) {//se não há candidatos para ler, ou já se chegou ao fim do ficheiro,
saímos do while
                            vazio = true;
                            break;
                        }
                        filme = filme.trim();
                        for   (Filme   f   :   edicoes.get(numEdicao   -
1).getFilmes()) { //percorre-se a lista de filmes para se encontrar o
nomeado
                            if      (filme.equals(f.getNome())      ||
filme.equals(f.getRealizador().getNome())) {//se o nome do filme for
igual ao nome do filme na lista (ou o nome do realizador, no caso do
prémio Melhor Realizador), nomeia-se o filme
                                premio.nomeiaFilme(f);
                                break;
                            }
                        }
                    }
                } else if (indexPremios == 8) {
                    while (premio.getAtoresCandidatos().size() < 4) {
                        nomeAtor   =   lerDados.readLine();//lê-mos   a
linha, que ou é o nome do ator ou o separador, se não há candidatos para
ler
                        if (nomeAtor == null || nomeAtor.contains("---
-------")) {//se não há candidatos para ler, ou já se chegou ao fim do
ficheiro, saímos do while
                            vazio = true;
                            break;
                        }
                        nomeAtor = nomeAtor.trim();
                        for (Ator a : this.atores) {//percorrer a lista
de atores para encontrar o ator
                            if     (a.getNome().equals(nomeAtor)     &&
a.getAnosCarreira() > 20) { //se o nome do ator coincide com o nome do
ator na lista e o ator na lista tem mais de 20 anos de carreira, nomeia-
se o ator
                                premio.nomeiaAtor(a);
                                break;
                            }
                        }
                    }
                }
                indexPremios++; //avançar para o próximo prémio
                if (!vazio) { //se vazio é false, lê-se a linha, que
```

```
será um separador, se vazio é true já se leu o separador anteriormente
                    line = lerDados.readLine();
                }
            }
            lerDados.close();
        }

        /**
         * Método que permite carregar as pontuações de um ficheiro
         */
        private void carregaPontuacoes() throws IOException {
            int i = 0; //variável que representa o índice da lista de
pontuações do candidato
            int j = 0; //variável que representa a posição do valor da
pontuação na lista de pontuações do candidato
            final   int   NUMPERITOS   =   edicoes.get(numEdicao   -
1).getPeritos().size();
            String pontos = ""; //varíavel que guarda o valor da
pontuação que se lê
            int indexPremios = -1;
            FileReader inStream = new FileReader("Edicao" + numEdicao
+ "\\Pontuacoes.txt");
            BufferedReader lerDados = new BufferedReader(inStream);
            String line;
            line = lerDados.readLine().trim();
            while (line != null) {
                if (!line.contains("----------")) { //vemos se a linha
não é o separador
                    for (char aux : line.toCharArray()) { //percorremos
a linha
                        if (aux != '*') { //se o carater não é igual ao
separador que separa os valor das pontuações
                            pontos     +=     Character.toString(aux);
//acrescenta-se o caráter ao valor da pontuação
                        } else { //se é igual, mete-se a pontuação na
lista de pontuações do candidato
                            edicoes.get(numEdicao                   -
1).getPremios().get(indexPremios).setPontuacao(i,   j   %   NUMPERITOS,
Integer.parseInt(pontos));
                            //insere-se o valor inteiro representado
por pontos na lista de pontuações do candidato i na posição j%NUMPERITOS
                            pontos = ""; //reset da variável que guarda
a pontuação
                            if ((j + 1) % NUMPERITOS == 0) { //se
chegou-se ao fim da lista de pontuações do candidato i, avança-se para
o próximo
                                i++;
                            }
                            j++;
                        }
                    }
                } else { //se a linha é o separador avançamos para o
próximo prémio e faz-se reset das variáveis i e j
                    indexPremios++;
                    i = 0;
                    j = 0;
                }
                line = lerDados.readLine();
            }
            lerDados.close();
        }
```

```
        /**
         * Método que permite gravar os atores num ficheiro
         */
        private void gravaAtores() throws IOException {
            String separador = "--------------------------------";
            FileWriter outStream = new FileWriter("Edicao" + numEdicao
+ "\\Atores.txt");
            BufferedWriter bW = new BufferedWriter(outStream);
            PrintWriter out = new PrintWriter(bW);
            for (Filme filme : edicoes.get(numEdicao - 1).getFilmes())
{
                out.println(separador); //inserimos o separador sempre
que gravamos um filme novo
                if (filme.getAtorPrincipal() != null) {
                    //verifica-se se o filme tem ator principal e
imprime-se o separador "Ator Principal", e imprime-se o seu nome, M
(género) e os seus anos de carreira
                    out.printf("Ator         principal:\n%s\nM\n%d\n",
filme.getAtorPrincipal().getNome(),
filme.getAtorPrincipal().getAnosCarreira());
                } else {
                    //se não imprime-se o separador "Ator Principal",
VAZIO,VAZIO e 0
                    out.println("Ator principal:\nVAZIO\nVAZIO\n0");
                }
                if (filme.getAtrizPrincipal() != null) {
                    //verifica-se se o filme tem atriz principal e
imprime-se o separador "Atriz Principal", o seu nome, F (género) e os
seus anos de carreira
                    out.printf("Atriz        principal:\n%s\nF\n%d\n",
filme.getAtrizPrincipal().getNome(),
filme.getAtrizPrincipal().getAnosCarreira());
                } else {
                    //se não imprime-se o separador "Atriz Principal",
VAZIO,VAZIO e 0
                    out.println("Atriz principal:\nVAZIO\nVAZIO\n0");
                }
                out.println("Atores Secundarios:"); //imprime-se este
separador para sinalizar que daqui para a frente estão gravados os atores
secundários
                for (Ator atorSec : filme.getAtoresSecundarios()) {
//imprime-se o nome, género e os anos de carreira de atorSec
                    out.printf("%s\n%s\n%d\n",      atorSec.getNome(),
(atorSec.getGenero() ? "M" : "F"), atorSec.getAnosCarreira());
                }
            }
            out.close();
        }

        /**
         * Método que permite gravar os filmes num ficheiro
         */
        private void gravaFilmes() throws IOException {
            FileWriter outStream = new FileWriter("Edicao" + numEdicao
+ "\\Filmes.txt");
            BufferedWriter bW = new BufferedWriter(outStream);
            PrintWriter out = new PrintWriter(bW);
            for (Filme filme : edicoes.get(numEdicao - 1).getFilmes())
{ //percorremos a lista de filmes para gravá-los
                //grava-se o nome do filme, o género e o nome do
```

```
realizador e o género deste
                out.printf("%s\n%s\n%s\n%s\n",        filme.getNome(),
filme.getGenero(),                      filme.getRealizador().getNome(),
(filme.getRealizador().getGenero() ? "M" : "F"));
            }
            out.close();
        }

        /**
         * Método que permite gravar os candidatos num ficheiro
         */
        private void gravaCandidatos() throws IOException {
            String separador = "--------------------------------";
            FileWriter outStream = new FileWriter("Edicao" + numEdicao
+ "\\Candidatos.txt");
            BufferedWriter bW = new BufferedWriter(outStream);
            int indexPremio = 0;
            PrintWriter out = new PrintWriter(bW);
            for    (Premio    premio    :    edicoes.get(numEdicao    -
1).getPremios()) { //percorremos a lista de prémios
                out.println(separador); //grava-se o separador
                if (indexPremio < 4) { //se é um prémio de ator/atriz
principal ou secundário grava-se o nome do candidato
                    //e  o  filme  no  qual  ele  participa,  em  linhas
diferentes
                    for    (int    candidato    =    0;    candidato    <
premio.getAtoresCandidatos().size(); candidato++) {
                        out.printf("%s\n%s\n",
premio.getAtoresCandidatos().get(candidato).getNome(),
premio.getFilmesCandidatos().get(candidato).getNome());
                    }
                } else if (indexPremio == 5) { //se é o prémio de melhor
realizador grava-se o nome do realizador nomeado
                    for    (int    candidato    =    0;    candidato    <
premio.getFilmesCandidatos().size(); candidato++) {

out.println(premio.getFilmesCandidatos().get(candidato).getRealizador(
).getNome());
                    }
                } else if (indexPremio == 4 || (indexPremio > 5 &&
indexPremio < 8)) { //se é um prémio dado a um filme grava-se
                    //o nome do filme nomeado
                    for    (int    candidato    =    0;    candidato    <
premio.getFilmesCandidatos().size(); candidato++) {

out.println(premio.getFilmesCandidatos().get(candidato).getNome());
                    }
                } else { //por último, se é o prémio carreira só se
grava o nome do nomeado ao prémio
                    for    (int    candidato    =    0;    candidato    <
premio.getAtoresCandidatos().size(); candidato++) {

out.println(premio.getAtoresCandidatos().get(candidato).getNome());
                    }
                }
                indexPremio++; //avança-se para o próximo prémio
            }
            out.close();
        }

        /**
```

```
         * Método que permite gravar os peritos num ficheiro
         */
        private void gravaPeritos() throws IOException {
            FileWriter outStream = new FileWriter("Edicao" + numEdicao
+ "\\Peritos.txt");
            BufferedWriter bW = new BufferedWriter(outStream);
            PrintWriter out = new PrintWriter(bW);
            for    (Perito    perito    :    edicoes.get(numEdicao    -
1).getPeritos()) { //percorremos a lista de peritos para gravá-los
                //grava-se o nome e o género do perito, em linhas
diferentes
                out.printf("%s\n%s\n",              perito.getNome(),
(perito.getGenero() ? "M" : "F"));
            }
            out.close();
        }

        /**
         * Método que permite gravar as pontuações num ficheiro
         */
        private void gravaPontuações() throws IOException {
            String separador = "--------------------------------";
            FileWriter outStream = new FileWriter("Edicao" + numEdicao
+ "\\Pontuacoes.txt");
            BufferedWriter bW = new BufferedWriter(outStream);
            PrintWriter out = new PrintWriter(bW);
            for    (Premio    premio    :    edicoes.get(numEdicao    -
1).getPremios()) {
                out.println(separador); //imprime-se o separador
                for     (int     linha     =     0;     linha     <
premio.getPontuacoes().size(); linha++) {
                    //percorremos a lista de pontuações para ter as
pontuações de cada candidato
                    for     (int     coluna     =     0;     coluna     <
premio.getPontuacoes().get(linha).size(); coluna++) {
                        //percorremos a lista com as pontuações do
candidato e grava-se as pontuações, separadas pelo asterisco

out.print(premio.getPontuacoes().get(linha).get(coluna) + "*");
                    }
                    out.println(); //fazemos parágrafo
                }
            }
            out.close();
        }

        /**
         * Método responsavel pela limpeza da consola
         */
        private void limparConsola() {
            try {
                Robot limpa = new Robot();
                limpa.keyPress(KeyEvent.VK_CONTROL);
                limpa.keyPress(KeyEvent.VK_L);
                limpa.keyRelease(KeyEvent.VK_CONTROL);
                limpa.keyRelease(KeyEvent.VK_L);
                try {
                    Thread.sleep(10);
                } catch (InterruptedException ex) {
                    System.out.println("ERRO");
                }
```

```
            } catch (AWTException e) {
                System.out.print("ERRO\n");
            }
            System.out.printf("\t\t\t%dª Edição do Festival de Cinema
%d\n", numEdicao, ano);
        }
    }
```

5.5. Perito

```
    package com.mycompany.festivalcinema;

    import java.util.Scanner;

    public class Perito extends Pessoa {

        protected Perito(String nome, boolean genero) {
            super(nome, genero);
        }

        @Override
        public String toString() {
            String perito = super.toString();
            return perito;
        }

        /**
         *
         * @param premio - prémio no qual inserir a pontuação
         * @param indiceCandidato - linha na matriz das pontuações que
indica a qual
         * candidato estamos a atribuir a pontuação
         * @param indicePerito - coluna na matriz das pontuações que
indica o perito
         * que atribui a pontuação
         * @param scan - o scanner para ler a pontuação
         * @return true se a pontuação inserida é valida (entre 1 e 10
e inteira),
         * false caso contrário
         */
        protected boolean inserePontuacao(Premio premio, int
indiceCandidato, int indicePerito, Scanner scan) {
            System.out.printf((this.getGenero() ? "O perito %s " : "A
perita %s ") + "atribui ao candidato a pontuação: ", this.getNome());
            String aux;
            double pontuacao;
            while (true) { //não sai do while enquanto o utilizador não
inserir um número
                aux = scan.nextLine();
                try {
                    pontuacao = Double.parseDouble(aux);
                    break; //só chega aqui se o parseDouble não atirar
a NumberFormatException
                } catch (NumberFormatException e) { //entra aqui se não
é inserido um número
                    System.out.println("A pontuação deve ser um
número!");
                }
            }
```

```
            if (pontuacao > 0 && pontuacao <= 10 && pontuacao == (int)
pontuacao) { //verifica-se se a pontuação atribuída é inteira e se está
entre 1 e 10, inclusive
                premio.setPontuacao(indiceCandidato,     indicePerito,
(int) pontuacao);
                return true;
            }
            return false;
        }

        @Override
        public boolean equals(Object obj) {
            if (obj == null) {
                return false;
            }
            if (obj == this) {
                return true;
            }
            if (this.getClass() != obj.getClass()) {
                return false;
            }
            Perito perito = (Perito) obj;
            return   this.getNome().equalsIgnoreCase(perito.getNome())
&& this.getGenero() == perito.getGenero();
        }
    }
```

5.6. Premio

```
    package com.mycompany.festivalcinema;

    import java.util.ArrayList;
    import java.util.Collections;

    public class Premio {

        private final String nome;
        private final ArrayList<ArrayList<Integer>> pontuacoes;
        private ArrayList<Filme> filmes;
        private ArrayList<Ator> atores;
        private final double[] mediasPontuacoes;
        private Filme vencedor;

        protected Premio(String nome) {
            this.nome = nome;
            this.filmes = new ArrayList<>(0);
            this.atores = new ArrayList<>(0);
            this.mediasPontuacoes = new double[4];
            //se o prémio for relacionado exclusivamente com filme
(Melhor filme, melhor realizador, etc.)
            //pomos a lista dos atores a nula por questões de memória
e tratamento de exceções
            if (!(nome.contains("Ator") || nome.contains("Atriz") ||
nome.contains("Carreira"))) {
                this.atores = null;
            }
            //se o prémio tem no nome "Carreira" (indica que é o prémio
carreira)
            //pomos a lista dos filmes  a nula por questões de memória
e tratamento de exceções
```

```
            if (nome.contains("Carreira")) {
                this.filmes = null;
            }
            this.pontuacoes = new ArrayList<>(4);
            for (int i = 0; i < 4; i++) {
                this.pontuacoes.add(new ArrayList<>(0));
            }
            this.vencedor = null;
        }

        protected String getNome() {
            return this.nome;
        }

        protected ArrayList<ArrayList<Integer>> getPontuacoes() {
            return this.pontuacoes;
        }

        protected void setPontuacao(int candidato, int perito, int
pontuacao) {
            this.pontuacoes.get(candidato).set(perito, pontuacao);
        }

        protected void nomeiaFilme(Filme filme) {
            this.filmes.add(filme);
        }

        protected void nomeiaAtor(Ator ator, Filme filme) {
            this.atores.add(ator);
            nomeiaFilme(filme);
        }

        protected void nomeiaAtor(Ator ator) {
            this.atores.add(ator);
        }

        protected ArrayList<Ator> getAtoresCandidatos() {
            return atores;
        }

        protected ArrayList<Filme> getFilmesCandidatos() {
            return filmes;
        }

        protected Filme getVencedor() {
            return vencedor;
        }

        protected double[] getMediasPontuacoes() {
            return mediasPontuacoes;
        }

        @Override
        public String toString() {
            return nome;
        }

        /**
         * Método que calcula as médias das pontuações atribuídas aos
candidatos
         *
```

```
         */
        private void calcularMedias() {
            if (mediasPontuacoes[0] == 0 ||
Double.isNaN(mediasPontuacoes[0])) {
                int tam = pontuacoes.get(0).size(); //vê quantas
possíveis pontuações foram atribuídas ao candidato
                for (int linha = 0; linha < 4; linha++) {
                    double somaPontuaçõesCandidato = 0;
                    int numPontuacoes = 0;
                    for (int coluna = 0; coluna < tam; coluna++) {
                        somaPontuaçõesCandidato +=
pontuacoes.get(linha).get(coluna);
                        if (pontuacoes.get(linha).get(coluna) != 0) {
//certificar que foi pontuado (se é 0 significa que não foi pontuado)
                            numPontuacoes++;
                        }
                    }
                    mediasPontuacoes[linha] = somaPontuaçõesCandidato
/ numPontuacoes;
                }
            }
        }

        /**
         * Método que ordena todas as listas relativamente à média de
pontuações que
         * os candidatos obtiveram, através de um bubble sort.
         *
         * @throws IndexOutOfBoundsException - se não houver candidatos
atira esta
         * exceção tratada noutro lugar
         */
        protected void ordenaPontuações() throws
IndexOutOfBoundsException {
            calcularMedias();
            int n = mediasPontuacoes.length;
            for (int i = 0; i < n - 1; i++) {
                for (int j = 0; j < n - i - 1; j++) {
                    if (mediasPontuacoes[j] <= mediasPontuacoes[j + 1])
{
                        swap(j, j + 1);
                    }
                }
            }
            empateVencedores();
            determinaVencedor();
        }

        /**
         * Verifica se houve empates entre os candidatos, o desempate
será feito
         * calculando os desvios padrões e verificando que tem o menor
desvio padrão
         * nas suas pontuações
         *
         */
        private void empateVencedores() {
            double[] desviosPadrao = {-1, -1, -1, -1}; //preenchemos o
array assim por questões de verificação
            for (int i = 0; i < mediasPontuacoes.length - 1; i++)
{//percorremos o vetor das médias (já ordenado) a verificar se houve
```

```
empates
                if (mediasPontuacoes[i] == mediasPontuacoes[i + 1]) {
//se há empate, será entre posições consecutivas e calcula-se o desvio
padrão dos candidatos
                    if (desviosPadrao[i] == -1) { //para certificar que
não se calcula o desvio padrão demasiadas vezes
                        desviosPadrao[i] = desvioPadrao(i);
                    }
                    if (desviosPadrao[i + 1] == -1) { //para certificar
que não se calcula o desvio padrão demasiadas vezes
                        desviosPadrao[i + 1] = desvioPadrao(i + 1);
                    }
                }
            }
            //Bubble sort para organizar segundo os desvios padrão, se
houver empate, quem tem menor desvio padrão "ganha"
            for (int i = 0; i < mediasPontuacoes.length; i++) {
                for (int j = 0; j < mediasPontuacoes.length - i - 1;
j++) {
                    if (mediasPontuacoes[j] == mediasPontuacoes[j + 1]
&& desviosPadrao[j] > desviosPadrao[j + 1]) {
                        swap(j, j + 1);
                        double aux = desviosPadrao[j]; //trocar o
desvio padrão da posição j com j+1
                        desviosPadrao[j] = desviosPadrao[j + 1]; //se
não a ordem dos candidatos ficaria incorreta
                        desviosPadrao[j + 1] = aux;
                    }
                }
            }
        }

        /**
         * Este método guarda em vencedor o filme que tem o candidato
que ganhou o
         * prémio
         */
        private void determinaVencedor() {
            if (!Double.isNaN(mediasPontuacoes[0])) {//verifica se no
vetor das médias das pontuações, na posição 0 aparece NaN (Not a Number)
                //pois se aparece então as pontuações ainda não foram
atribuídas
                if (filmes != null && vencedor == null) { //certifica
que o vencedor só é definido uma vez para não se aumentar
                    //o número de prémios em cada vez que esta função
é chamada
                    this.vencedor = filmes.get(0);
                    this.vencedor.incrementaNumeroPremios();
                }
            }
        }

        /**
         * Este método é um auxiliar aos bubble sort's que usamos, troca
toda a
         * informação do candidato na posição i com a informação do
candidato na
         * posição j
         *
         */
        private void swap(int i, int j) throws
```

```
IndexOutOfBoundsException {
            double aux = mediasPontuacoes[i];
            mediasPontuacoes[i] = mediasPontuacoes[j];
            mediasPontuacoes[j] = aux;
            Collections.swap(this.pontuacoes, i, j);
            if (this.atores != null) {
                Collections.swap(this.atores, i, j);
            }
            if (this.filmes != null) {
                Collections.swap(this.filmes, i, j);
            }
        }

        /**
         * Este método vai calcular o desvio padrão das pontuações
atribuídas ao
         * candidato especificado por posCandidato
         *
         * @param posCandidato - a posição do candidato do qual calcular
o desvio
         * padrão
         * @return o valor de desvio padrão
         */
        private double desvioPadrao(int posCandidato) {
            double soma = 0;
            int tam = pontuacoes.get(posCandidato).size();
            for (int i = 0; i < tam; i++) {
                soma                +=                Math.pow((double)
pontuacoes.get(posCandidato).get(i)  -  mediasPontuacoes[posCandidato],
2);
            }
            return Math.sqrt(soma / mediasPontuacoes.length);
        }

        /**
         * Este método imprime as pontuações e indica se ainda não foram
atribuídas
         *  as  pontuações  aos  candidatos  ou  se  o  prémio  não  tem
candidatos
         */
        protected void imprimePontuações() {
            System.out.println("\n- " + nome + ": ");
            try {
                ordenaPontuações();  //se  não  houver  candidatos,  é
atirada a IndexOutOfBoundsException e apanhada aqui
                if (!Double.isNaN(mediasPontuacoes[0])) { //verifica se
no vetor das médias das pontuações, na posição do vencedor aparece NaN
(Not a Number)
                    //pois  se  aparece  então  as  pontuações  ainda  não
foram atribuídas
                    for (int i = 0; i < mediasPontuacoes.length; ++i)
{
                        if (atores != null) {//se o prémio for para um
ator/atriz
                            if    (!this.nome.contains("Carreira"))    {
//se  não  for  o  prémio  Carreira,  indica-se  por  qual  filme  o  ator
participou

System.out.print(atores.get(i).getNome()      +      "      em      "      +
filmes.get(i).getNome());
                            } else { //se for o prémio carreira só se
```

```
imprime o nome do ator

System.out.print(atores.get(i).getNome());
                                }
                            } else {//se o prémio for para um filme
                                if   (this.nome.contains("Realizador"))   {
//se for o prémio de Melhor Realizador, indica-se este e o filme que ele
direcionou

System.out.print(filmes.get(i).getRealizador().getNome()  +  "  por  "  +
filmes.get(i).getNome());
                                } else { //se não, só se imprime o nome do
filme

System.out.print(filmes.get(i).getNome());
                                }
                            }
                            System.out.printf(":                    %.2f\n",
mediasPontuacoes[i]); //imprime pontuação
                        }
                    } else {
                        System.out.println("Pontuações não atribuídas");
                    }
                } catch (IndexOutOfBoundsException e) { //se ocorre
IndexOutOfBoundsException é que ainda não foram definidos os candidatos
                    System.out.println("Não tem candidatos.");
                }
            }

            /**
             * Este método imprime o vencedor da categoria e indica ao
utilizador se
             * ainda não é possível determinar um vencedor (ou não há
candidatos ou as
             * pontuações ainda não foram atribuídas)
             */
            protected void vencedorCategoria() {
                System.out.print(nome + ": ");
                try {
                    ordenaPontuações(); //se não houver candidatos, é
atirada a IndexOutOfBoundsException e apanhada aqui
                    if (!Double.isNaN(mediasPontuacoes[0])) { //verifica se
no vetor das médias das pontuações, na posição do vencedor aparece NaN
(Not a Number)
                        //pois se aparece então as pontuações ainda não
foram atribuídas
                        if (atores != null) {//se o prémio for para um
ator/atriz
                            if (!this.nome.contains("Carreira")) {//se não
for o prémio Carreira, indica-se por qual filme o vencedor participou
                                System.out.println(atores.get(0).getNome()
+ " em " + vencedor.getNome() + "\n");
                            } else {//se for o prémio carreira só se imprime
o nome do ator
                                System.out.println(atores.get(0).getNome()
+ "\n");
                            }
                        } else {//se o prémio for para um filme
                            if   (this.nome.contains("Realizador"))   {//se
for o prémio de Melhor Realizador, indica-se este e o filme que ele
direcionou
```

```
System.out.println(vencedor.getRealizador().getNome()  +  "  por  "  +
vencedor.getNome() + "\n");
                            } else {//se não, só se imprime o nome do filme
                                System.out.println(vencedor.getNome()    +
"\n");
                            }
                        }
                    } else {
                        System.out.println("Ainda sem vencedor.\n");
                    }
                } catch (NullPointerException | IndexOutOfBoundsException
e) {//se ocorre uma destas exceções é que ainda não foram definidos os
candidatos
                    System.out.println("Ainda sem vencedor.\n");
                }
            }

            @Override
            public boolean equals(Object obj) {
                if (obj == null) {
                    return false;
                }
                if (obj == this) {
                    return true;
                }
                if (this.getClass() != obj.getClass()) {
                    return false;
                }
                Premio pre = (Premio) obj;
                return       this.nome.equalsIgnoreCase(pre.nome)       &&
this.vencedor.equals(pre.vencedor) && this.filmes.equals(pre.filmes) &&
this.atores.equals(pre.atores) && this.pontuacoes == pre.pontuacoes;
            }
        }
```

5.7. Pessoa

```
    package com.mycompany.festivalcinema;

    public abstract class Pessoa {

        private final String nome;
        private final boolean genero; //true->Homem false->Mulher

        protected Pessoa(String nome, boolean genero) {
            this.nome = nome;
            this.genero = genero;
        }

        protected String getNome() {
            return nome;
        }

        protected boolean getGenero() {
            return genero;
        }

        @Override
        public String toString() {
            String pessoa;
```

```
            pessoa = "Nome: " + getNome() + "\n";
            pessoa += "Género: " + (getGenero() ? "Masculino" :
"Feminino") + "\n";
            return pessoa;
        }

        @Override
        public boolean equals(Object obj) {
            if (obj == null) {
                return false;
            }
            if (obj == this) {
                return true;
            }
            if (this.getClass() != obj.getClass()) {
                return false;
            }
            Pessoa pes = (Pessoa) obj;
            return       this.genero       ==       pes.genero       &&
this.nome.equalsIgnoreCase(pes.nome);
        }
    }
```

5.8. Realizador

```
    package com.mycompany.festivalcinema;

    public class Realizador extends Pessoa {

        protected Realizador(String nome, boolean genero) {
            super(nome, genero);
        }

        @Override
        public String toString() {
            return super.toString();
        }

        @Override
        public boolean equals(Object obj) {
            if (obj == null) {
                return false;
            }
            if (obj == this) {
                return true;
            }
            if (this.getClass() != obj.getClass()) {
                return false;
            }
            Realizador reali = (Realizador) obj;
            return    this.getGenero()    ==    reali.getGenero()    &&
this.getNome().equalsIgnoreCase(reali.getNome());
        }
    }
```

5.9. Main

```
    import com.mycompany.festivalcinema.FestivalCinema;

    public class Main {
```

```
    public static void main(String[] args) {
        FestivalCinema festivalCinema = new FestivalCinema();
        festivalCinema.menu();
    }

}
```

6. Anexo B – Diagrama UML