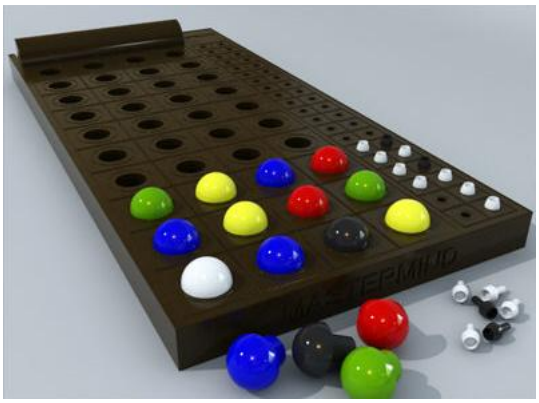
Disciplina: Programação II

Docente: Professora Lígia Ferreira

Curso: Engenharia Informática

Mastermind



Trabalho Elaborado Por:

Marlene Oliveira Nº 25999

Pedro Mateus Nº 26048

Ano Lectivo 2009/2010

Índice

Introdução

No âmbito da disciplina de Programação II, foi-nos pedida a implementação de um programa em JAVA permita jogar o popular jogo de estratégia Mastermind. Este trabalho tem como objectivo permitir que os alunos utilizem a linguagem JAVA e o paradigma da Programação Orientada a Objectos na realização do referido trabalho.

Pretende-se que o programa permita que o jogo seja do tipo humano contra computador e que possua os seguintes requisitos:

- Gerar uma chave com as cores disponíveis, que o jogador tentará adivinhar;
- Receber a tentativa, ou seja, a jogada efectuada pelo utilizador;
- Classificar a Jogada;
- Permitir a visualização das tentativas efectuadas e a respectiva classificação;
- Permitir que o utilizador insira um nome de jogador;
- Permitir que os jogadores joguem à vez contra o computador;
- Pontuar cada jogo com o número de tentativas que foram necessárias para o jogador adivinhar o padrão;
- Comparar as pontuações dos jogadores e atribuir a vitória ao jogador que obtiver a pontuação menor.

Para facilitar a elaboração do nosso trabalho, decidimos dividir o código pelas seguintes classes: Classificacao.java; Jogada.java; Jogador.java; Mastermind.java; Menu.java; Savetab.java; Sequencia.java; Tabuleiro.java e Vitoria.java. a descrição do funcionamento destas classes e dos seus métodos e variáveis de instância pode ser consultada de seguida.

1.Descrição das Classes do Mastermind

Classe Classificacao.java:

A classe Classificacao.java permitirá ao programa atribuir uma pontuação ao jogador. Esta classe recebe como variável de instância o inteiro pontos, que corresponderá ao número de pontos do jogador. Para além do construtor vazio e do construtor que recebe argumentos, esta classe possui os seguintes métodos de instância:

- getPontos();
- setPontos();
- toString().

O construtor permitirá que o inteiro pontos seja inicializado a zero e o construtor que recebe argumentos permitirá que esse mesmo inteiro, anteriormente inicializado a zero, seja igual ao argumento dado.

O método getPontos() permitirá que o programa retorne o inteiro correspondente ao número de pontos do jogador.

O método setPontos() permitirá que o programa altere, caso seja necessário, o valor inteiro correspondente ao número de pontos do jogador.

O método toString() retornará uma String semelhante à seguinte: “Pontuação: 10 ponto(s)”.

Classe Jogada.java:

A classe Jogada.java tem como principal função receber a jogada efectuada pelo utilizador, avaliá-la e retornar a jogada efectuada e a respectiva avaliação. Esta classe tem como variáveis de instância os arrays de caracteres 'jogada' e 'avalia', o array de inteiros antirepete e o Scanner 's'. O array jogada irá conter a sequência de caracteres dada pelo jogador no input. O array avalia irá conter a classificação dada pelo programa à jogada efectuada, ou seja, este array irá conter as peças pequenas brancas (no caso de a peça possuir a mesma cor de uma das peças das sequência aleatória gerada pelo computador, mas se encontrar na posição incorrecta) e/ou as peças pequenas pretas (no caso de a peça possuir a mesma cor de uma das peças da referida sequência aleatória gerada pelo computador e se encontrar na posição correcta). O array antirepete irá permitir que as peças pequenas brancas resultantes da avaliação não sejam repetidas quando há cores repetidas que se encontram nas posições erradas mas que, contudo, pertencem à sequência correcta. O Scanner s permitirá que a informação dada pelo jogador no input seja lida.

Esta classe contém os seguintes métodos de instância:

- geraJogada();
- avaliaJogada();
- toString();
- resetAvalia();
- restantes().

O método geraJogada() terá como funcionalidade principal receber o input do utilizador, colocar o referido input num array temporário e retornar o conteúdo desse array temporário no array jogada. Inicialmente, é solicitada ao jogador a introdução, peça a peça, das quatro opções que constituirão a sua jogada (as cores disponíveis são: Blue (B) , Green (G) , Lilac (L) , Red (R) , White (W) e Yellow (Y)). As peças serão introduzidas sob a forma de uma String, que posteriormente será convertida em carácter e colocada na posição correspondente do array temporário (cores). O ciclo do/while, contido no método, restringirá as opções de input (o jogador só pode inserir caracteres que correspondam às opções disponíveis). Seguidamente o programa utiliza um ciclo for para copiar os caracteres contidos no array temporário para o array jogada. Finalmente, o método retorna o array jogada, que já contém os caracteres correspondentes à jogada efectuada.

O método avaliaJogada() recebe um array de caracteres (posteriormente correspondente à sequencia aleatória gerada pelo computador), compara char a char o conteúdo do array que que contém a jogada com cada um dos caracteres que estão contidos no array que o método recebe. Para que a comparação possa ser efectuada peça a peça colocámos cada um dos elementos do array numa String, utilizando o método valueOf() da classe String do JAVA. Desta operação resultaram quatro Strings (a, b, c e d), cujo conteúdo corresponde, respectivamente, ao carácter contido em cada uma das posições do array de caracteres 'jogada' (a String a conterá o carácter da posição zero, etc…). O mesmo método foi utilizado para isolar cada um dos elementos do array que o método irá receber (correspondente à sequência aleatória gerada pelo computador). As Strings e, f, g e h contêm estes elementos. Finalmente, o método irá comparar, utilizando o método equals da classe String do JAVA, cada uma das Strings que contêm os elementos isolados do array jogada com a String correspondente à mesma posição do array que o método recebe. Caso a peça a comparar corresponda à peça que se encontra na mesma posição do array que o método recebe, o valor na posição correspondente do array antirepete seja igual a zero e a posição do array avalia possui a peça 'v' (o que indica que aquela posição do array avalia está vazia), o programa retorna 'b' na posição correspondente do array 'avalia' e preenche a posição correspondente do array antirepete com o valor inteiro 1. Caso contrário, o programa compara

a String que contém a peça a avaliar com as restantes Strings que constituem a sequência, verifica se o array avalia possui a peça 'v' naquela posição e verifica se o array antirepete tem o valor zero nessa mesma posição. Se a String for igual a uma destas Strings restantes, o array avalia tenha a peça 'v' naquela posição e o array antirepete tiver o valor zero na posição correspondente à ocupada pela String da sequência a adivinhar, o programa deve retornar 'w' na posição do array 'avalia' correspondente à peça que se encontra no local errado mas que, porém, se encontra na sequência e retornar 1 na posição correspondentes do array antirepete.

A classe Jogada.java contém também o método toString() cuja principal função é retornar a jogada efectuada pelo utilizador e a respectiva avaliação efectuada no método avaliaJogada() sob a forma de uma String. Para tal, colocámos os arrays 'jogada' e 'avalia' sob a forma de duas Strings distintas, respectivamente jog e av, utilizando o método toString() da classe Arrays do package util do JAVA. O método retornará uma String semelhante à seguinte: "Jogada Efectuada: [R, G, B, Y] ---→ [b, w, b, w] ".

O método resetAvalia() tem como principal função efectuar o reset dos arrays avalia e antirepete antes de cada uma das avaliações da jogada a efectuar, ou seja, este método elimina os valores contidos nos arrays avalia e antirepete antes de ser efectuada uma nova avaliação.

Finalmente, esta classe possui o método restantes() que recebe um inteiro e retorna uma String semelhante a: "Jogadas Restantes: 2", correspondendo 2 ao inteiro que o método recebe.

Para esta classe foi fornecido um construtor vazio para que pudessem ser respeitadas as convenções.

Classe Jogador.java:

A classe Jogador.java tem como principal funcionalidade criar um perfil de jogador que irá identificar o utilizador durante o jogo. Esta classe tem como variável de instância a String 'nome'. A String nome conterá o nome do jogador. Para além do construtor e do construtor que recebe como argumento uma String, a classe jogador possui os seguintes métodos de instância:

- getNome();
- setNome();
- toString();
- clone();

O construtor permitirá que a String nome seja inicializada com "".

O construtor que recebe como argumento uma String permite que, através da referência this, a String nome contenha os mesmos elementos que a String que o método recebe, por outras palavras, o nome que o utilizador definir como seu será associado automaticamente ao conteúdo da String nome desta classe.

O método getNome() irá retornar a String nome, ou seja, irá retornar o nome do jogador.

O método setNome() recebe uma String e permitirá ao jogador alterar o nome que definiu anteriormente.

O método toString() permite que o perfil do jogador (constituído pelo nome que o jogador definiu) seja retornado sob a forma de uma String. A String retornada terá um aspecto semelhante à seguinte: "Nome do Jogador: Pedro".

O método clone() permitirá reproduzir um objecto Jogador exactamente igual a outro mas diferenciado, ou seja, dado um objecto Jogador, ao cloná-lo, queremos outro objecto distinto do original mas exactamente igual.

Classe Mastermind.java:

A classe Mastermind.java terá como principal objectivo controlar o fluxo do jogo. Esta classe tem como variáveis de instância o Scanner s, o objecto Menu m, o objecto Vitoria v, os objectos Sequencia seq1 e seq 2, os objectos Jogada jog1 e jog2, os Tabuleiros t1 e t2, as Strings nome1 e nome2 e os booleans check1 e check2. O Scanner s permitirá que a informação dada pelo jogador no input seja lida, o objecto Menu m permitirá gerar os menus que irão orientar o jogador ao longo do jogo, o objecto Vitoria v permitirá comparar as pontuações de ambos os jogadores e gerar um vencedor, os objectos Sequencia seq1 e seq2 permitem gerar as sequencias aleatória que os jogadores têm de adivinhar (cada jogador tem uma sequência aleatória única), os objectos Jogada jog1 e jog2 permitem gerar as jogadas de ambos os jogadores e avaliar ambas, os objectos Tabuleiro t1 e t2 permitem gerar a arraylist que corresponderá ao tabuleiro de jogo de cada um dos jogadores, as Strings nome1 e nome2 são inicializadas com o valor "" e os booleans check1 e check2 permitirão verificar se cada um dos jogadores adivinhou a sequência aleatória.

Para além do construtor vazio, esta classe contém os seguintes métodos de instância:

- perfilJogo1();
- perfilJogo2();
- geraJogo1();
- geraJogo2();
- gestorJogo1();
- gestorJogo2().

Os métodos perfilJogo1() e perfilJogo2() têm como principal função gerar um perfil de jogo para cada um dos jogadores. Inicialmente, no método perfilJogo1(), o é apresentado o menu da entrada. No método perfilJogo2() isto não ocorre. Seguidamente, ambos os métodos recebem o input do utilizador que corresponde ao nome com que os jogadores serão identificados ao longo do jogo. Seguidamente são criados os objectos Jogador de cada um dos jogadores, que permitirá gerir os dados dos jogadores. Finalmente, em ambos os métodos é retornada toda a informação de cada um dos jogadores (esta informação encontra-se contida em cada um dos objectos Jogador).

Os métodos geraJogo1() e geraJogo2() têm como principal função gerar os jogos dos jogadores. Inicialmente são efectuados os resets aos arrays jogada e avalia para que, caso esta jogada seja seguinte a outra, não haja erros na avaliação. De seguida são inicializados os booleans check1 (no método geraJogo1()) e check2 (no método gerajogo2()), sendo gerada uma jogada e efectuada a respectiva avaliação para cada um dos jogadores. São criadas duas Strings, uma para cada uma das jogadas efectuadas. Seguidamente é redefinido o tamanho do tabuleiro para que este corresponda ao valor do inteiro n da classe menu. Finalmente as jogadas são inseridas nos tabuleiros respectivos e o programa mostrará o tabuleiro do jogador, as tentativas restantes e, por fim, verifica se o jogador inseriu a sequência correspondente à combinação aleatória.

Os métodos gestorJogo1() e gestorJogo2() têm como principal função gerir os jogos de cada um dos jogadores. Inicialmente é calculada a pontuação de cada um dos jogadores. De seguida, o programa verifica se o jogador acertou na combinação correcta ou se chegou ao final do tabuleiro sem acertar na referida sequência. Caso isto se verifique, o programa sai. Caso contrário, o programa gera um novo jogo. No gestorJogo2() ocorre um processo semelhante ao descrito anteriormente, contudo, quando o jogador 2 termina o jogo, o programa compara a pontuação deste jogador com a do jogador 1 e determina qual deles é o vencedor ou se ocorre um empate. Caso o jogador não tenha acertado na sequência correcta, o programa gerará um novo jogo para o jogador 2.

No método main da classe Mastermind, o programa gerará jogos do popular jogo Mastermind e guardará os tabuleiros finais de cada um dos jogadores na arraylist Savetab salvar. O programa apresentará todos os menus necessários seja no inicio ou no final de qualquer jogo. O jogador poderá, depois de jogar o número de jogos que desejar, consultar cada jogo feito todas as vezes que assim desejar até que decida, finalmente, sair.

Classe Menu.java:

A classe Menu.java permite a criação dos menus que irão orientar o jogador ao longo do jodo. Esta classe implementa cinco métodos de instância (menuEntrada(), menuGameOver(), menuSair(), menuConsultar() e menuSizeTabuleiro()), à parte do construtor. Esta classe tem como variáveis de instância dois inteiros (opcao e n), um boolean (sair) e um Scanner (s). O inteiro 'opcao' corresponde à opção tomada pelo jogador nos menus. O inteiro 'n' corresponderá ao tamanho do tabuleiro de jogo. O boolean sair corresponderá à intenção do jogador de abandonar, ou não, o jogo. O Scanner s permitirá que a informação dada pelo jogador no input seja lida.

O construtor permitirá que o inteiro dado como opção no input corresponda ao inteiro 'opcao' desta classe e que o inteiro dado no input como tamanho do tabuleiro de jogo corresponda ao inteiro 'n' desta classe.

O método menuEntrada() apresenta um pequeno menu que permitirá ao jogador escolher se prefere sair ou continuar e iniciar um jogo. Para tal, o método recebe um valor de input que corresponderá ao inteiro opcao. A restrição garante que o utilizador não pode inserir números diferentes de zero ou de um. Se o utilizador escolher a opção zero, o programa não retorna nada e sai. Se o utilizador decidir que pretende jogar Mastermind, ou seja, se o jogador escolher a opção um, o programa avança para o menu que permite escolher o tamanho do tabuleiro.

O método menuGameOver() apresenta um pequeno menu que permitirá ao jogador, quando se verifica o Game Over, escolher se pretende jogar novamente ou sair do jogo. Para tal, o método recebe um valor de input que corresponderá ao inteiro opcao. A restrição garante que o utilizador não pode inserir números diferentes de zero ou de um. Se o utilizador escolher a opção zero, o programa retorna o boolean sair igual a true e sai do jogo. Caso contrário, o programa retorna o boolean sair sem qualquer alteração (false) e avança para o menu da escolha do tamanho do tabuleiro de jogo.

O método menuSair() permite que o jogador decida se pretende abandonar o jogo ou consultar os tabuleiros de jogos anteriores. Este método recebe um inteiro, que corresponderá ao valor do inteiro opcao neste menu. Caso o jogador escolha a opção sair, o programa retorna o boolean sair com o valor true. Caso contrário, o programa retorna o boolean sair com o valor false.

O método menuConsultar() permite que o jogador escolha os tabuleiros de jogo a consultar. Para tal o programa recebe um inteiro correspondente ao inteiro opcao neste menu. O inteiro inserido não pode ser par, não pode ser menor que zero e também não pode ser maior que 99.

O método menuSizeTabuleiro() permite que o jogador defina um tamanho para o tabuleiro de jogo. Contudo, como definimos o tamanho máximo de qualquer tabuleiro de jogo como 16, o inteiro que o programa recebe (e que corresponde ao inteiro n) não pode ser maior que 16 nem, por razões óbvias, menor que zero.

Classe Savetab.java:

A classe Savetab.java tem como principal função guardar os tabuleiros finais dos jogos efectuados. Esta classe tem como variáveis de instância o inteiro maxtab, que corresponde ao tamanho máximo da arraylist, e a arraylist tabs, que irá conter os tabuleiros finais dos jogos. Para além do construtor vazio, esta classe contém os métodos de instância:

- getTab();
- insereTab();
- toString().

O construtor permite que, quando é criado um novo objecto Savetab, a arraylist tenha o tamanho máximo de 100 posições.

O método getTab() permite que o programa retorne o tabuleiro correspondente à posição dada pelo inteiro que este método recebe.

O método insereTab() permite que o programa insira um tabuleiro na arraylist.

O método toString() retorna uma String que contém todos os tabuleiros que se encontram na arraylist.

Classe Sequencia.java:

A classe Sequencia.java permitirá que o computador gere a combinação aleatória das cores disponíveis, que será posteriormente adivinhada pelo jogador. Esta classe possui como variável de instância um array de caracteres (seq), onde será guardada a sequência aleatória gerada. Para além do construtor, esta classe possui os seguintes métodos de instância:

- geraSequencia();
- toString();
- getSequencia().

O construtor permitirá que, de cada vez que seja criado um objecto do tipo Sequencia, seja criada uma nova sequência aleatória.

O método geraSequencia() permite a criação de uma sequência aleatória a partir de um array (o array 'hip') que contém os caracteres correspondentes às cores disponíveis: Blue, Green, Lilac, Red, White e Yellow. Seguidamente são gerados números inteiros aleatórios, sendo cada um correspondente a uma posição do array que contém as hipóteses para o padrão. Posteriormente o padrão resultante é copiado para o array temporário (o array 'seqtemp'), que será finalmente retornado.

O método toString() permite que o programa retorne uma String semelhante à seguinte: "Sequência Correcta: [R,G,B,Y]".

O método getSequencia() permite que o programa retorne o array de caracteres que contém a sequência aleatória.

Classe Tabuleiro.java:

A classe Tabuleiro.java tem como principal objectivo criar o tabuleiro de jogo. Esta classe tornará também possível adicionar elementos (jogadas) ao tabuleiro, retornar esses mesmos elementos ou todo o tabuleiro de jogo (sob a forma de uma String), verificar se o tabuleiro de jogo está cheio, verificar o número de espaços vazios no tabuleiro, alterar o tamanho máximo e retornar o tamanho do referido tabuleiro de jogo. Esta classe tem como variáveis de instância o inteiro maxsize e a arraylist tabuleiro. O inteiro maxsize corresponderá ao tamanho máximo que o tabuleiro de jogo poderá ter e a arraylist corresponderá ao array

dinâmico que será, efectivamente, o tabuleiro de jogo. Para além do construtor que recebe argumentos, esta classe possui os seguintes métodos de instância:

- getSize();
- setMaxsize();
- toString();
- espaçoVazio();
- getPlay();
- isFull();
- insere().

O construtor que recebe argumentos tem como função permitir que, de cada vez que é criado um objecto Tabuleiro, o inteiro maxsize corresponda ao inteiro dado pelo utilizador como argumento e permitir a criação de um tabuleiro com o tamanho máximo correspondente a esse mesmo argumento.

O método getSize() tem como principal função retornar o tamanho do tabuleiro.

O método setMaxsize() permite alterar o valor correspondente ao tamanho máximo do tabuleiro.

O método toString() permitirá ao programa retornar o tabuleiro de jogo sob a forma de uma String.

O método espaçoVazio() tem como principal funcionalidade retornar o número de espaços vazios do tabuleiro.

O método getPlay() tem como principal função retornar a jogada correspondente à posição dada pelo inteiro que o método recebe.

O método isFull() permite verificar se o tabuleiro está cheio.

O método insere() recebe um objecto do mesmo tipo que é aceite pela arraylist e coloca-o no tabuleiro.

Classe Vitoria.java:

A classe Vitoria.java tem como principal objectivo a comparação das pontuações dos jogadores e indicar um vencedor ou, a ocorrer, um possível empate. Esta classe não possui variáveis de instância. Esta classe possui os seguintes métodos de instância:

- vitoria();
- checkVitoria().

O método vitoria() recebe como argumentos duas Strings, dois inteiros e três booleans. As Strings corresponderão nome de cada um dos jogadores, os inteiros corresponderão às pontuações dos jogadores e os booleans indicarão se os jogadores acertaram, ou não, na combinação certa (caso isto se verifique, os booleans conterão o valor true). Este método efectua a comparação das pontuações para poder ser escolhido um vencedor. Note-se que, no Mastermind, o jogador com menor pontuação é o vencedor. Caso não seja escolhido um vencedor através desta comparação, o programa verifica se que ocorre um empate.

O método checkVitoria() recebe como argumento um array de caracteres. Este método tem como principal função verificar se todas as posições do array que recebe estão preenchidas com o carácter ‘b’. Caso isto se verifique, o método retornará o boolean check como true.

Para esta classe foi fornecido um construtor vazio para que se pudessem cumprir as convenções.

Conclusão

Tendo em conta o objectivo do trabalho, podemos concluir que o programa cumpre a maioria dos requisitos pretendidos e funciona de uma forma algo simples e, de certo modo, funcional. Este trabalho permitiu-nos aprofundar os nossos conhecimentos no que respeita ao paradigma da Programação Orientada a Objectos.

Conclui-se portanto que este trabalho foi proveitoso para a nossa aprendizagem no que respeita à Programação Orientada a Objectos.

Bibliografia

- Slides da cadeira de Programação II por Lígia Ferreira, Ano Lectivo 2009-2010;
- Marques, F. Mário; *Java 5 e Programação por Objectos*; FCA – Editora de Informática, Lda; Lisboa; 2008.
- http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html#lastIndexOf%28int%29
- http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Character.html
- http://java.sun.com/j2se/1.4.2/docs/api/java/util/Arrays.html
- http://java.sun.com/j2se/1.4.2/docs/api/java/util/ArrayList.html