UNIVERSIDADE DE ÉVORA

Domino

Relatório do Trabalho Prático de Programação II

Docentes: Lígia Ferreira

João Coelho

Ricardo Fusco, 29263

Bruno Santos, 29451

Engenharia Informática – 2012/2013

Introdução

O objectivo principal deste trabalho é criar, em java, um programa que nos permita executar um jogo de Dominó entre o utilizador e o computador sendo assim possível demonstrar e meter em prática toda a matéria dada até agora na cadeira de Programação II.

Foram criadas 3 classes e um interface para o bom funcionamento do software, uma para a implementação das pedras de Dominó, outra para iniciar um jogo entre o computador e jogador e outra onde se encontra o loop principal do jogo e onde se processam as jogadas. Foi também criado um interface Pedras que será implementado na classe PedraDomino com os métodos obrigatórios, definindo assim as características que uma pedra de Dominó terá que ter.

Desenvolvimento

Classes Utilizadas:

Começámos por criar a seguinte classe:

PedraDomino - Esta classe contem as informações relativas a uma pedra de Dómino, sendo elas o lado esquerdo e direito da peça. Contém também uma arraylist que ira conter todas as pecas do jogo após chamada a função todas(int n). Esta classe implementa o interface Pedras.

Variáveis da classe:

```
private static int ladoDir, ladoEsq;
private static ArrayList<PedraDomino> ladoDir = new ArrayList<PedraDomino>();
```

Métodos da classe:

public PedraDomino(int lE, int lD) - Construtor que recebe como argumentos o lado esquerdo e direito da pedra para que possa ser criado o objecto do tipo PedraDomino com tais parâmetros.

public void setLadoEsq(int lE) - Método modificador para alterar o valor do lado esquerdo da peça.

public void setLadoDir(int lD) - Método modificador para alterar o valor do lado direito da peça.

public int getLadoEsq() - Método selector que retorna o valor do lado esquerdo da peça.

public int getLadoDir() - Método selector que retorna o valor do lado direito da peça.

public int getTotalPontos() - Método que retorna o numero de pontos total dessa peça(ladoEsq+ladoDir).

public boolean eDuplo() - Método que retorna um valor booleano. Retorna true se a peça for um "duplo" (ladoEsq = ladoDir).

public String pedraToString() - Método que retorna uma representação em String da pedra de Domino.

public static ArrayList<PedraDomino> todas(int n) - Método que adiciona todas as peças do jogo duplo-n de acordo com n e retorna a ArrayList com todas essas peças.

De seguida foi criada a classe onde irá ser iniciado o jogo entre o utilizador e o computador.

JogoDeDomino - Esta classe contém todas as colecções e métodos necessários para iniciar um jogo entre o computador e o utilizador e as funções respectivas para executar jogadas dos mesmos.

Variáveis da classe:

```
private int inc[1];
public Random aleatorio = new Random();
private Scanner input;
public int randomInt;
public int numP[2];
public int player[3] = 0;
public Iterator<PedraDomino>  itJ;
public Iterator<PedraDomino>  itC;
public Iterator<PedraDomino>  itM;
public ArrayList<PedraDomino>  maoJ = new ArrayList<PedraDomino> () ;
public ArrayList<PedraDomino>  maoC = new ArrayList<PedraDomino> () ;
public ArrayList<PedraDomino>  mesa = new ArrayList<PedraDomino> () ;
public ArrayList<PedraDomino>  banca = new ArrayList<PedraDomino> () ;
```

Os iteradores foram utilizados porque nos deparamos com alguns problemas a adicionar ou a remover elementos duma arraylist utilizando ciclos for-each e for 's normais. O modulo Random foi necessário nos casos em que é necessário retirar uma peça da banca ao acaso e quando e necessário decidir qual a primeira mão a ser criada (computador ou jogador). Quanto ás colecções necessárias usamos apenas 4 arraylists.

[1] Esta variável, como está descrito nos comentários do código, é usada para incrementar quantas vezes o computador foi a banca no sentido de as limitar a 1.
[2] Número de peças.
[3] Variável que determina quem vai jogar (0 - Comp 1 - Jogador)

Métodos da classe:

public JogoDeDomino(String jog, int n) - Construtor que recebe como argumentos uma string(nome do jogador) e um inteiro n que irá definir o tipo de jogo a criar e quantas peças irão ser disponibilizadas. Este construtor inicia um jogo duplo-n entre o computador e o utilizador, ou seja, cria a banca e remove um número de peças variável consoante n (n+1 peças) da banca para a mão de cada um. A criação das mãos é também aleatória para que não seja sempre a mesma mão a ser criada primeiro. Por fim é chamada a função para iniciar a mesa.

public void iniciaMesa(ArrayList<PedraDomino>mj, ArrayList<PedraDomino>mc, String jog, int n) - Método que recebe como argumentos uma String(nome do jogador), um inteiro n e duas arraylists de pedras (mão do comp e do jog) e verifica qual o jogador que tem a peça dupla mais alta e coloca-a na mesa, caso não tenham nenhum duplo será retirada uma peça aleatoriamente da banca.

public int calcNumP(int n) - Método que recebe como argumento um inteiro n e devolve um inteiro. Calcula o número de peças de acordo com n.
public String mostraBanca() - Método que devolve uma representação em String da banca.
public String mostraMesa() - Método que devolve uma representação em String da mesa.
public String mostraPedras() - Método que devolve uma representação em String da mão do Jogador.
public String mostraMaoC() - Método que devolve uma representação em String da mão do computador.
public void fimJogo() - Método onde é verificado se há condições para terminar o jogo. Vai verificar se a banca esta vazia. Caso esteja é feita a contagem dos pontos de cada jogador e o que tiver menos pontos ganha o jogo.
public void jogaJogador(PedraDomino x) - Método que recebe como argumento uma pedra de dominó e joga-a. Antes de ser chamado este método é pedido ao utilizador para especificar qual a pedra da mão que deseja jogar e é essa a pedra que será passada como argumento. Primeiro é pedido ao utilizador em que extremidade deseja colocar a pedra e é de seguida verificado se a peça escolhida pode ser de facto jogada nessa extremidade.

public void jogaComputador() - Método que faz a jogada do computador. Este método funciona de maneira semelhante ao jogaJogador(), primeiro é contado o número de peças da mão que o computador pode jogar e consoante esse número são contados os pontos de cada peça que pode ser jogada e aquela que tiver mais pontos é a que será jogada. Caso não seja possível jogar alguma peça iria ser removida uma da banca e é então chamada esta função recursivamente para verificar se pode

jogar essa peça ou se passa a jogada para o jogador.

Por fim temos a classe Domino onde se encontra o loop principal que é onde se processa o jogo utilizando os métodos das outras 2 classes.

Domino - Esta classe contém apenas um método onde irão ser utilizados todos os métodos que pertencem ás classes JogoDeDomino e PedraDomino. Foi também utilizado um módulo chamado JOptionPane, tanto nesta classe como na classe JogoDeDomino, para mostrar uma janela com uma mensagem ou a pedir um input. Este módulo foi usado para as mensagens de boas vindas, para pedir o nome e o tipo de jogo (duplo-n) e para mensagens de erro.

Variáveis da classe:

Nesta classe não há variáveis globais apenas variáveis locais:

```
int n[4];
int ind[5];
int podeJog[6];
int in;
int fim;
```

Métodos da classe:

public void jogoD() - Método onde se encontra o loop principal do jogo. Primeiro são dadas as boas vindas ao jogador e é-lhe pedido para inserir o seu nome e o tipo de jogo que quer iniciar (duplo-n). De seguida é iniciado o loop principal que continuará a correr enquanto a banca não estiver vazia. É aqui que é verificado se o jogador pode jogar alguma peça ou não. De seguida com o auxilio da variável player irá alternando entre jogador e computador mudando o seu valor no fim de cada jogada.

[4] Valor que determina o numero de peças existentes em jogo (duplo-n).
[5] Variável que irá conter o input do utilizador relativamente a pedra que deseja remover e que irá ser o índice dessa pedra.
[6] Variável que irá ser incrementada para a contagem do numero de peças que se pode jogar.

Interfaces:

Pedras - Como já foi referido anteriormente foi implementado este interface na classe PedraDomino para que fiquem definidos quais os métodos que a classe tem que ter.

Métodos implementados:

```
public void setLadoEsq(int lE);
public void setLadoDir(int lD);
public int getLadoEsq();
public int getLadoDir();
public int getTotalPontos();
public boolean eDuplo();
public String pedraToString();
```

Conclusão

Concluindo e resumindo conseguimos com este trabalho ultrapassar algumas dificuldades que tivemos ao início, nomeadamente na manipulação dos dados nas arraylists. Obtivemos vários problemas com a remoção e adição de pedras nas arraylists enquanto percorríamos as mesmas como por exemplo excepções de dois tipos “concurrentModificationException” e excepções relativamente a “índex out of range”, por vezes adicionava peças na mesa mas não removia da mão, mas conseguimos resolver esse problema. Inicialmente tínhamos o computador a jogar ao calhas, ou seja a primeira peça que ele encontrava que podia jogar jogava-a. No fim do trabalho pensámos num pequeno algoritmo muito simples, para que o computador tivesse um minimo de inteligência no jogo, que no fundo o que faz é contar quantas peças podem ser jogadas e é feita a contagem dos pontos de cada uma dessas peças, no fim a peça com mais pontos que o computador puder jogar é jogada.

Em geral pode-se afirmar que o balanço do trabalho foi positivo, mas temos a noção que poderíamos ter introduzido mais algumas partes da matéria dada nas aulas e ter feito o código do jogo mais geral e mais organizado.