UNIVERSIDADE DO OESTE DE SANTA CATARINA – UNOESC

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS VINÍCIUS MAHL

MARCELO SEHNEM

JOSÉ RAFAEL PEREIRA

JHONATAN TRINDADE

JOGO DA VELHA MASTER

São Miguel do Oeste

2014MATHEUS VINÍCIUS MAHL

MARCELO SEHNEM

JOSÉ RAFAEL PEREIRA

JHONATAN TRINDADE

JOGO DA VELHA MASTER

Trabalho desenvolvido pelos alunos o curso de Ciências da Computação, na Universidade do Oeste de Santa Catarina, como parte da disciplina de Programação II, ministrada pelo professor André da Silva.

São Miguel do Oeste

2014

SUMÁRIO

INTRODUÇÃO.................................................................................................................3

1. O JOGO.................................................................................................................4
   1. Bibliotecas.............................................................................................................4
2. CLASSE “ListaJogadoresMain”..........................................................................5
3. CLASSE “Jogo”....................................................................................................6
4. Classe “Jogadores”................................................................................................6
5. CLASSE “Arquivo”..............................................................................................7

CONCLUSAO...................................................................................................................9

INTRODUÇÃO

O trabalho a seguir foi desenvolvido com o objetivo de entreter dois jogadores com o clássico jogo conhecido como “Jogo da Velha” ou “Tic-Tac-Toe”.

Com seu desenvolvimento baseado em Java, o programa permite que se escolham dois jogadores que, alternadamente, marcam o tabuleiro com uma marcação característica, sendo o primeiro jogador identificado por xis (X) e o segundo por um circulo (O).

O objetivo do jogo é que um jogador marque três vezes o seu símbolo em linha, seja vertical, horizontal ou diagonal, antes do outro jogador, assim ganhando o jogo.

1. O JOGO

Utilizando a plataforma Java SE (Standard Edition) em modo visual, o jogo foi desenvolvido utilizando dois “Form JFrames” e duas classes, utilizados para criar uma lista de jogadores, o jogo em si, gerenciar a leitura e gravação no arquivo e gerenciar o nome do jogador, respectivamente.

O jogo em si foi criado utilizando botões como forma de tabela, ordenados em 3x3, a fim de criar o ambiente do jogo. Cada botão é referenciado para ser uma opção do jogador, que pode escolher entre qualquer um dos nove a fim de marcar a escolha do jogador.

Mas antes de iniciar o jogo, uma tela vai listar os nomes dos jogadores e vai permitir adicionar, remover ou escolher os jogadores que irão jogar. Todas as opções serão vistas a seguir.

* 1. Bibliotecas

As bibliotecas utilizadas no desenvolvimento do programa foram as seguintes:

* java.util.ArrayList
* javax.swing.JOptionPane
* java.awt.Color
* java.io.BufferedWriter
* java.io.BufferedReader
* java.io.File
* java.io.FileReader
* java.io.FileWriter

O “arrayList” foi utilizado para guardar os objetos do tipo “Jogadores”, para serem salvos no arquivo e também lidos pela lista de jogadores. Seu uso foi importante para conseguirmos guardar todos os nomes adicionados na lista e busca-los no arquivo posteriormente.

A biblioteca “JOptionPane” foi utilizada para informar mensagens de erro para o usuário, e também para informar se alguém ganhou ou deu empate no jogo. Se alguém ganhar o jogo, os três símbolos que formaram a linha que ganhou serão pintados em vermelho, com o uso da biblioteca “Color”.

As bibliotecas que estão presentes em “java.io” foram todas utilizadas na classe “Arquivo”, para permitir a criação, leitura e gravação do arquivo, utilizando os métodos necessários para que o gerenciamento do arquivo fosse possível.

1. CLASSE “ListaJogadoresMain”

Essa classe é a que vai ser executada por primeiro, pois contém o método “Main”. O ambiente que os jogadores vão encontrar é de uma seleção de jogadores, contendo uma lista de jogadores que estão ou serão adicionados. Caso seja executado pela primeira vez no sistema, a lista se encontrará vazia, obrigando os jogadores a adicionar o seu nome para realizar a seleção utilizando o campo de escrita abaixo da lista e o botão “Adicionar Jogador”.

Após a primeira execução, os jogadores permanecerão na lista que foi salva utilizando a classe Arquivo, que será explicada mais adiante. A tela ainda permite aos jogadores adicionar mais nomes ou então apagar algum que já esteja na lista, selecionando o nome na lista e apertando o botão “Deletar Jogador”.

Dois botões de seleção de jogadores vão pegar o jogador selecionado na lista e adicionar ao lado dos campos “Jogador 1(X):” e “Jogador 2(O):”, identificando os dois jogadores que irão jogar. Depois de selecionado os dois jogadores, o jogo poderá ser iniciado apertando o botão “Jogar”.

Algumas limitações foram implantadas para manter o controle de execução sem erros do programa e com painéis (JOptionPane) que informam o usuário qual erro está cometendo, permitindo a ele corrigir em seguida sem encerrar o programa. Para adicionar um jogador, o campo de escrita deverá ter um nome de pelo menos quatro caracteres e no máximo doze caracteres, podendo ser alfanumérico e/ou símbolos (Ex.: asd123@#$!). Para adicionar um jogador, ele deve estar selecionado na lista, e para poder jogar, os dois jogadores devem estar selecionados e não podem ser iguais, lembrando que os testes diferem letras maiúsculas de minúsculas, sendo “João” diferente de “joão”.

1. CLASSE “Jogo”

Após apertar o botão “Jogar” na classe “ListaJogadoresMain”, o ambiente de seleção de jogadores irá fechar e abrirá o ambiente de jogo, contendo a tabela de botões, o nome dos jogadores, a informação de quem será a vez de jogar e um ranking, utilizado para contar as vitórias de cada jogador e também os empates. O botão “Limpar Ranking” é utilizado para zerar a contagem das vitórias e empates, lembrando que elas não serão salvas após o encerramento do programa.

Os jogadores, em sua respectiva vez, irão apertar no botão de sua escolha, o qual irá marcar o símbolo do respectivo jogador e bloquear o botão, para que não possa ser modificado, sendo assim, apertos acidentais não poderão ser corrigidos. Quando algum jogador conseguir marcar uma linha com três símbolos seus, os botões são bloqueados e essa linha tem seus símbolos marcados em vermelho, significando a vitória desse jogador, junto com o aparecimento de uma janela de mensagem (JOptionPane) indicando o jogador que ganhou ou se acabou em empate. Quando ocorrer a vitória ou empate, o botão “Novo Jogo” servirá para limpar o tabuleiro e reiniciar a vez do jogador.

Nesse ambiente, não foi preciso criar limitações de erros por parte do usuário, além do bloqueio dos botões, pois o ambiente de jogo não utilizará nenhuma entrada de texto.

1. Classe “Jogadores”

Essa classe foi criada para que um objeto guarde um atributo do tipo “String” chamado “nomeJogador”, o qual está em proteção privada, e será acessado por meio de métodos públicos “Getter” e “Setter”, chamados de “getJogador” e “setJogador” e utilizados para buscar ou informar, respectivamente, o atributo “nomeJogador”.

Essa classe é utilizada na classe “ListaJogadoresMain”, dentro da função de ação dos botões de “Adicionar Jogador”, “Selecionar Jogador 1(X):” e “Selecionar Jogador 2(O):”, para criar um objeto que guardará o nome do jogador que foi selecionado, para poder guardar esse nome em um arrayList de objetos, o qual é utilizado pela lista de nomes e pela classe “Arquivo”, para poder salvar os nomes adicionados na lista. Ele também é utilizado pela classe “Arquivo” para gravar e receber os nomes que estão salvos no arquivo, também usando um arrayList de objetos.

1. CLASSE “Arquivo”

Essa classe foi criada para ler e gravar os nomes que foram adicionados na lista de jogadores. Ela possui três métodos, dois atributos e dois objetos, utilizados para gerenciar o processo de criação, gravação e leitura do arquivo.

Os atributos foram criados apenas para utilização dentro da classe, com proteção “private” e “final”, não podendo ser modificados ou utilizados por outras classes. O atributo “ARQUIVO” é do tipo “String” e guarda o diretório com o nome do arquivo (C:/JogoDaVelha/Jogadores.txt) o qual guardará todos os nomes adicionados na lista de jogadores. O atributo “DIRETORIO” também é do tipo “String” e guarda apenas o diretório que se encontrará o arquivo (C:/JogoDaVelha) com o objetivo futuro de criar esse diretório para poder salvar o arquivo dentro dele, caso ele não exista.

Os dois objetos criados nessa classe são do tipo File, nomeados de “file” e “arquivo”, onde o primeiro será utilizado com o atributo “DIRETORIO” para gerenciar a pasta, que será chamada de “JogoDaVelha”, e o segundo será utilizado com o atributo “ARQUIVO”, para gerenciar o arquivo(chamado de “Jogadores.txt”) onde serão salvos os nomes.

O primeiro método é o “criarDiretorio”, o qual criará uma pasta utilizando o método “mkdir()” no objeto “file”, resultando na criação da pasta no local descrito pelo atributo “DIRETORIO”. Em seguida ele testa se já existe um arquivo de nome “Jogadores.txt”, e caso não exista, ele irá cria-lo. Esse método possui um tratamento de erro caso não consiga criar o diretório ou o arquivo, enviando uma mensagem de erro caso isso ocorra.

O segundo método é chamado de “gravarArquivo”, e o objetivo dele é gravar o arrayList (denominado “jogador”) que ele recebe por parâmetro, no arquivo especificado por “ARQUIVO”, utilizando as classes “BufferedWriter” e “FileWriter”. Utilizando os métodos dessas duas classes de gravação, cada nome do arrayList “jogador” recebido é gravado em uma nova linha do arquivo. Caso o arquivo já tenha sido escrito, esse método de gravação irá apagar tudo e escrever o que foi mandado pelo parâmetro. Ele também tem um tratamento de erros caso não consiga gravar no arquivo, e manda uma mensagem de erro caso isso ocorra.

O terceiro método é chamado de “lerArquivo”, e possui um retorno de arrayList de “Jogadores”, o qual será lido do arquivo “Jogadores.txt”. Ele utiliza as classes “BufferedReader” e “FileReader”, que disponibilizam métodos para a leitura dos nomes dentro do arquivo e salvam dentro de um arrayList de nome “listaLida”, que será o retorno do método. Esse método é utilizado para buscar os nomes já salvos no arquivo de volta para a lista de jogadores que podem ser selecionados para jogar. Ele tem um tratamento de erro que informa se não foi possível ler do arquivo.

1. CONCLUSÃO

O jogo foi desenvolvido pensando em ser uma ferramenta de entretenimento, com a interação de dois jogadores competindo para ganhar. É um jogo baseado em lógica simples, e por isso o desenvolvimento do jogo foi simples. Como não houve muita dificuldade em criar o jogo em si, foi adicionada a tela de jogadores, que traz a possibilidade de escolher o nome do jogador e salvá-lo para ser utilizado outra vez.

A maior dificuldade foi de gerenciar o arquivo, utilizando as bibliotecas “java.io.\*”, além da utilização do “arrayList” para guardar os nomes para utilizar os métodos de gravação para o arquivo. Nessa parte foi preciso buscar resoluções com o professor ou na internet, pois muitos erros eram encontrados, mas resolvidos logo depois.

O desenvolvimento desse trabalho foi importante para descobrirmos as várias utilidades e ferramentas da linguagem Java, e com ele descobrimos que varias modificações podem ser feitas para modelar o programa do jeito que você deseja. Com esse projeto foi possível agregar ao conhecimento as várias utilidades das bibliotecas e dos conceitos de orientação a objetos, como classes, métodos e atributos.