Universidade Estadual de Maringá

Departamento de Informática

Curso: Ciência da Computação

Disciplina: 6888 – Paradigma de Programação Imperativa e Orientada a Objetos

Professora Drª. Valéria Delisandra Feltrim

Kick-Katch

Orientação a Objetos

Alessandra Harumi Iriguti RA: 88741

Juliano Cézar Chagas Tavares RA: 87940

Maringá

2015

1. Passo-a-passo para compilação e execução do programa

- Linguagem: Java
- IDE: NetBeans IDE 8.1
- Tipo de Projeto: Aplicação Java
- Versão Java: Java 8
- Deselvolvido em: Windows 10
- Testado em: Windows 8.1 e Windows 10

2. Conceitos de orientação a objetos

2.1. Encapsulamento

Na classe Jogador é possível observar o encapsulamento. Os atributos “numCamisa”, “nome”, “time” e “perfil” são privados (*private*). Isso significa que somente a classe em que foram declarados, no caso Jogador, estes atributos são visíveis. Para que as subclasses Batedor e Goleiro acessem as propriedades de Jogador, utiliza-se o termo *super* em seus construtores para acessar o construtor (que é público – *public*) da superclasse.

2.2. Herança

Nas classes Jogador, Batedor e Goleiro é possível observar também o conceito de herança. Batedor e Goleiro são filhos (subclasses) de Jogador, como pode-se observar em suas declarações o termo *extends* Jogador. Batedor e Goleiro possuem os mesmos atributos de Jogador. A diferença entre eles está nos métodos. O Batedor possui o método chutar, enquanto o Goleiro possui o método defender.

2.3. Polimorfismo

Não foi implementado no projeto.

3. Decisões de projeto

3.1. Criando as classes

Na classe Jogador, ao invés de ser declarado o método direcionar() : Ponto, foi declarado apenas os métodos chutar(Ponto ponto) e defender(Ponto ponto) nas classes Batedor e Goleiro, respectivamente.

Na classe Torcida, foram dados os atributos id e apelidoTime. Nessa classe é declarado um vetor de torcedores do tipo Torcedor. Além disso, há o método público geraTorcedores(int i) em que i é o número de torcedores que comporão a Torcida. A classe Torcedor possui o atributo id. Ambas as classes, Torcida e Torcedor, possuem métodos com o objetivo de apoiar ou vaiar seu time.

Na enumeração Ponto, não foi declarado o método errar(). Este método foi implementado nas classes Batedor e Goleiro. Na classe Time tem-se o atributo torcida do tipo Torcida além de nome. Há, também, um vetor de Jogador.

Na classe GerenciaPenalti inicializamos o jFrameInicial, tornando-o visível. É na classe GerenciaPenalti que tem-se métodos públicos e estáticos do tipo vetor de Jogador que geram os dados dos times. Por exemplo, para o time de Santos, o método *public static* Jogador[] inicializaObjetosSantos() cria-se um vetor de Jogador com onze posições que armazenará os jogadores do time. Inicializa-se, também, o *Random* que será usado para sortear a qualidade e a confiança dos jogadores. Em seguida, cria-se a torcida do time, o apelido do time, no caso "Peixe", e é gerado, para a torcida, cem torcedores. Depois da torcida criada, estabelecemos de fato Santos como Time do programa. Concebemos de forma aleatória, usando o *Random*, os perfis de cada jogador para depois geramos os jogadores que comporão o time. Guarda-se, então, os jogadores no vetor de jogadores.

Para criar as interfaces do jogo para facilitar a programação e a jogabilidade, foi utilizado a classe JFrame do pacote *javax.swing*. As telas a seguir descritas são filhas de javax.swing.JFrame.

A primeira tela que temos é a JFrameInicial . Nela temos o nome do jogo e o plano de fundo como imagem e encontramos dois botões: Jogar e Ajuda. Ao clicar no botão Ajuda, abre-se uma nova janela,JFrameHelp, ao lado da atual explicando o funcionamento do jogo. Ao clicar em Jogar, troca-se para a JFrameSelecionaTime, em que o usuário irá escolher com qual time irá jogar.

Na JFrameHelp, temos a descrição do funcionamento do jogo em uma JTextArea e um botão Fechar para simplesmente fechar essa janela de Ajuda.

Na JFrameSelecionaTime, tem-se seis times (Corinthians, Cruzeiro, Santos, Atlético Mineiro, Flamengo e Fluminense) para o usuário escolher. Nessa janela o jogador pode voltar para a tela inicial (clicando no botão Votlar) ou pode clicar em um time. Ao clicar em um time, mudar-se-á para uma nova janela JFrameExibeJogadores. Nessa nova janela, é possível vizualizar a imagem do escudo do time selecionado, o nome do time e uma lista com os nomes dos onze jogadores e sua função. O usuário poderá confirmar o time para jogar com ele ou, caso queira ver outro time, voltar para selecionar um novo time.

Depois de selecionado o time, o usuário irá escolher os cinco jogadores que irão bater o pênalti. Isso é feito no JFrameMeuTime. Nessa classe são declaradas as variáveis nomeTime1, que salvará o nome do time escolhido pelo usuário, e o nomeTimePC, que guardará o nome do time do computador, ambos do tipo *String*. Declara-se vetores do tipo Jogador que armazenarão os jogadores que irão para o jogo. Esse vetor de jogadores possui seis posições, a primeira será para o goleiro que irá automaticamente para o jogo. Nas outras cinco posições serão preenchidas pelos jogadores escolhidos pelo usuário. Nessa tela, aparecem dez botões com os nomes dos jogadores de função batedor. Na parte de baixo um campo que incialmente terá apenas o goleiro. Ao clicar no nome de um jogador, o botão dele ficará desabilitado e o texto dele irá para o campo logo abaixo. Teríamos agora o nome do goleiro e o nome do primeiro jogador que irá bater o pênalti. Após selecionar um jogador, não é possível removê-lo, nem mesmo mudar a ordem. O botão Confirmar só irá aparecer depois de escolhidos os cinco batedores – isso é controlado pela variável de inteiro contador – e os botões com os nomes do jogadores restantes ficam todos desabilitados.

É nessa tela que o jogo irá gerar qual será o time adversário. O computador irá escolher seu time, diferente do escolhido pelo usuário, aleatoriamente. Para isso, usamos novamente o *Random*. Sortea-se um número de 1 a 6 e este determinará o time do computador. Esse sorteio irá se

repetir enquanto o nome do time do computador for igual ao do usuário. Em seguida, gera-se o vetor de jogadores e os batedores do time do computador são determinados conforme as esoclhas de batedores feitas pelo usuário. Por exemplo, se o primeiro batedor escolhido pelo usuário foi o décimo primeiro jogador do vetor de jogadores, então o primeiro batedor do computador será o décimo primeiro jogador localizado em seu vetor de jogadores.

Confirmado os batedores, seguimos para a tela do jogo – JFrameJogo. Na tela temos uma imagem de um gol com seis botões representando as posições em que o usuário irá chutar ou defender. No canto superior direito tem-se o placar. Abaixo do gol, o JTextArea exibe as mensagens da situação do jogo. Mensagens instrucionais como "Escolha onde quer chutar/defender com <nome do jogador>", mensagens mostrando o resultado da batida do pênalti: "Chute para fora!", "É Gooooooooool!!!", "O goleiro defendeu" e a reação das torcidas, por exemplo, "Sua Torcida/Torcida adversária: O <apelido do time> é o melhor!!/Já é campeão !!!/Vamos <apelido do time> !!!/Vai <apelido do time> !!!/ Aaaaaaah.....". Essas mensagens se originam da classe Torcida. As quatro primeiras são aleatorizadas no método comemorar(), enquanto que a última é definida no método lamentar(). Após as cinco batidas de cada lado, se permanecer o empate, acontece então as cobranças alternadas. A mesma ordem de batedores irá se repetir até que, depois de uma batida de cada lado, algum time tenha vantagem no placar.

É declarado na classe JFrameJogo dois vetores de seis posições, um para guardar o goleiro e os batedores do usuário e outra para guardar o goleiro e os batedores do computador – estes que foram determinados na tela anterior. Também são declarados duas variáveis do tipo inteiro, pontosJog e pontosPC, que comporão o placar do jogo. Em seguida, declara-se duas variáveis do tipo Ponto, escolhaJogador e escolhaPC, que armazenarão as escolhas de chute/defesa do jogador e do computador, respectivamente. Para controlar a quantidade de cobranças, criamos uma variável inteira i e uma booleana acrescimo. Por fim, declaramos o estadoJogo do tipo EstadoJogo (uma enumeração declarada nessa classe mesmo) que inicialmente recebe o valor de CHUTANDO. O método public Ponto geraPosPC() irá aleatorizar a escolha do chute/defesa do computador.

O controle do jogo foi pensado como uma máquina de estados, que será representada pela enumeração EstadoJogo. Os estados possíveis dessa máquina são CHUTANDO e DEFENDENDO, e eles, por sua vez, representam qual interação o jogador está executando naquele momento no jogo. No início o estado é dado por CHUTANDO, já que o usuário que inicia fazendo a sua jogada.

O usuário clica então na posição que ele deseja chutar/defender. Ao clicar, o evento é acionado e a variável escolhaJogador recebe o valor que o botão clicado representa. É chamado, a partir desse evento, o método jogoChute() que realiza toda a lógica do jogo que envolve a opção de chutar. Nesse método está feita a programação referente à tabela de possibilidades de interação entre batedor e goleiro, as reações das torcidas e a atualização das características dos jogadores. Para que o computador faça a defesa, dentro desse método é chamado o método geraPosPC() que retorna uma posição aleatória para a defesa realizada pelo computador.

Após o usuário chutar, então é a vez do computador fazê-lo para que o usuário possa defender. Para tal é chamado o método jogo defesa, que possui lógica semelhante ao método jogoChute() explicado anteriormente. Uma posição aleatória é gerada para o chute do computador e é solicitada uma escolha do usuário para que a defesa seja feita. Quando o usuário clicar sobre o

botão, gera o evento e a subrotina entra em execução. Tanto no método jogoChute() quanto no jogoDefesa(), em caso de gol o placar é atualizado imediatamente e todas as interações e respostas da torcida também.

Após 4 turnos, chute e defesa, uma variável booleana é acionada para que inicie a verificação de ganhador através do método confereResultado() após cada rodada. Portanto quando o último jogador de cada um dos times jogar, será verificado se houve algum ganhador. Em caso afirmativo o jogo informa ao usuário o resultado e oferece, por meio de um JButton, a opção de jogar novamente. Em caso negativo as jogadas são retomadas, mas conferidas a cada turno até que haja um vencedor. Outros métodos auxiliares foram criados também, como o método estadoBotoes() que desativa a opção de clicar nos botões quando eles são acionados e é chamado novamente para reativá-los quanto necessário. Isso garante que o usuário não gere muitos eventos que podem comprometer o funcionamento do programa.

3.2. Organização das classes em pacotes

As subclasses de javax.swing.JFrame estão no pacote “apresentacao”, as imagens utilizadas no pacote “imagens” e o restante das classes no pacote “gerenciapenalti”. Todos esses pacotes estão agrupados em Pacotes de Código Fonte. O pacote “apresentação” contém todas as classes filhas de JFrame, ou seja, todas as classes visuais que foram criadas. Já no pacote “gerenciapenalti” estão agrupadas todas as classes descritas no documento e necessárias para que o jogo funcione, como Jogador, Batedor, Torcida, GerenciaPenalti e outras. No pacote “imagens” estão todos os arquivos *.png* que foram usados.