Tic Tae Toe
Programação Modular

Luiz Otávio Resende Vasconcelos

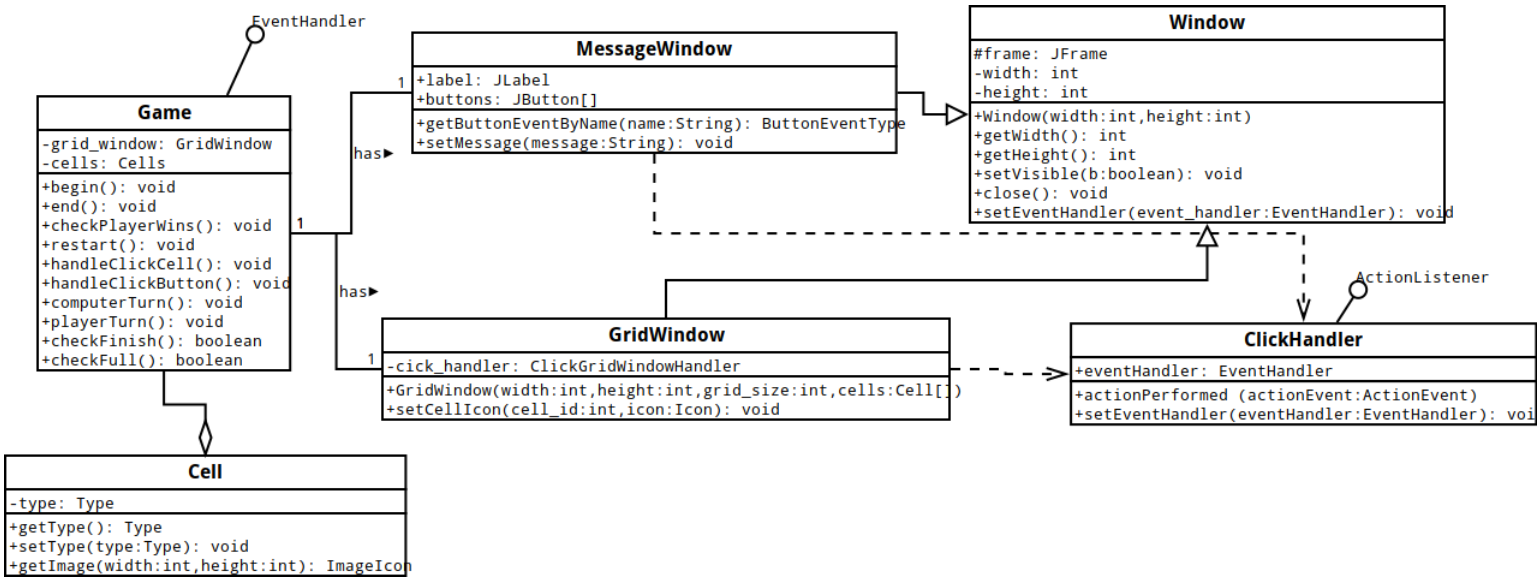
Outubro 2018

1 Introdução

O presente relatório visa descrever a resolução dos desafios propostos pela atividade 3 da disciplina de Programação Modular a qual foi solicitada a implementação de um "Jogo da Velha" (Tic Tae Toe) utilizando JFrame e conceitos aprendidos na disciplina.
O jogador é o primeiro a jogar. Em seguida, o computador escolhe uma casa aleatória não marcada. O jogo termina quando três casas seguidas são marcadas (na horizontal vertical ou diagonal) ou todas as casas são marcadas (empate), então, é exibida uma mensagem de vitória do jogador ou computador ou empate sendo exibido também dois botões: reiniciar e fechar jogo.

2 Desenvolvimento

O projeto foi desenvolvido usando a biblioteca gráfica da linguagem *Java® swing*, utilizando os componentes *JButton* e *JLabel*.
Abaixo é mostrado o diagrama de classes UML referente à modularização do projeto:
Nas próximas seções são detalhadas os detalhes de implementações e objetivos de cada módulo.



2.1 Game

Game é o módulo principal do jogo. Após instanciado, chamamos seu método publico *begin()* para iniciar o jogo. O jogo pode ser terminado com a chamada do *end()* ou por um evento do jogador. O módulo também armazena a memória do mapa atual do jogo e realiza o turno de cada jogador e trata os eventos.
Ele possui um *array* de células (os 9 quadrados do jogo) representadas pela classe *Cell* e dois tipos genéricos de janela: uma que exibe mensagens de aviso e outro para o jogo em si. Os módulos citados acima serão explicitados nas próximas seções.

2.2 Cell

Cell é o módulo responsável por encapsular os dados e operações a respeito das células do jogo (os 9 quadrados). Ele possui uma propriedade que indica de qual tipo aquela cédula é (vazia, círculo ou xis) e um método para alterar esse estado. Os tipos possíveis são descritos pelo *enum*:

```
public enum Type
{
    NONE,
    NOUGHT,
    CROSS
}
```

Possui também um método para retornar o *ImageIcon* da célula para ser exibido.

2.3 MessageWindow

MessageWindow é uma classe genérica para exibição de mensagens para o jogador. Basta a instanciarmos, setarmos a mensagem e exibir a janela.
Ela também possui botões genéricos que retornam eventos que podem ser capturados uma classe que implemente a interface *EventHandler*.

2.4 GridWindow

Módulo que abstrai as operações com a janela do jogo, a qual jogamos de fato. Ela é uma classe genérica para instanciar janelas "tabelas" (ou grids). Podemos adicionar ícones à tabela, no caso, xis e círculos.

2.5 Window

Módulo que interage com a interface de janelas do *swing*. É extendida pelas classes *GridWindow* e *MessageWindow*.

2.6 ClickHandler

Classe que implementa a *ActionListener* (interface necessária ser implementada para utilizarmos os eventos dos componentes do *swing*).

3 Resultados

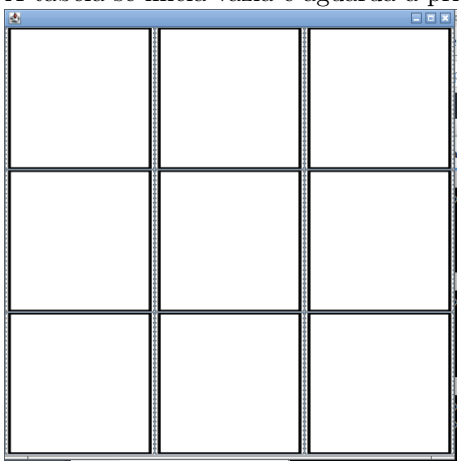
Para executarmos o jogo é necessário instanciá-lo e chamarmos o método *begin()* (como descrito nas seções anteriores). Para tal, criamos uma classe auxiliar *Main*:

```
public class Main
{
    public static void main(String[] args)
    {
        Game game = new Game();
        game.begin();
    }
}
```

Nas próximas seções são mostrados alguns resultados da jogabilidade e iteração do jogo.

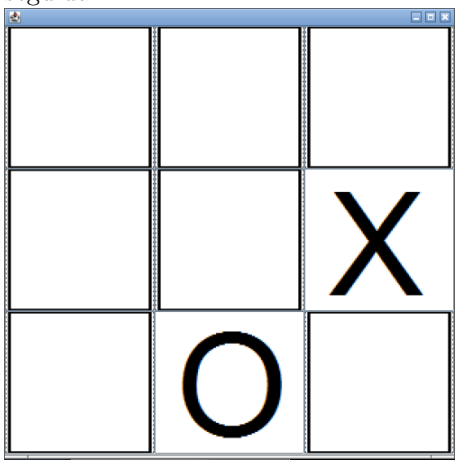
3.1 Primeira jogada

A tabela se inicia vazia e aguarda a primeira escolha do jogador:



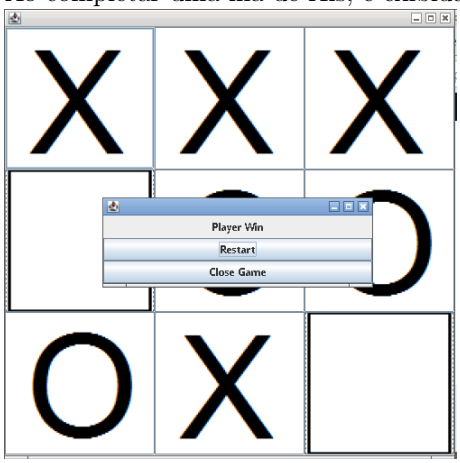
3.2 Turno do Jogador/Computador

Após o jogador escolher o primeiro quadro, o computador escolherá o dele em seguida:



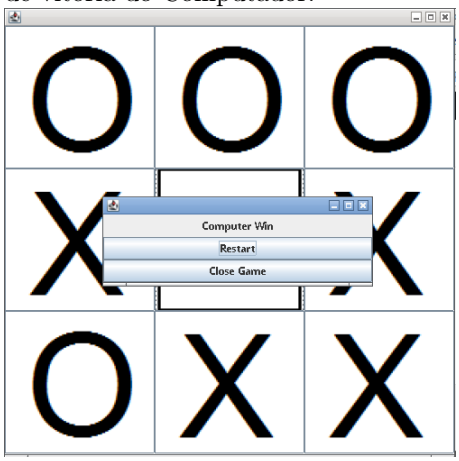
3.3 Vitória do Jogador

Ao completar uma fila de Xis, é exibida a mensagem de vitória do jogador:



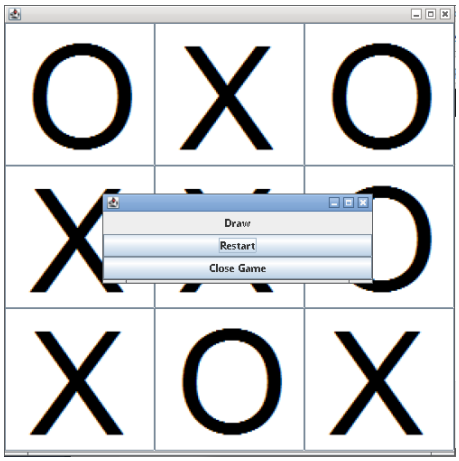
3.4 Vitória do Computador

Da mesma maneira, ao completar uma fila de Círculos, é exibida a mensagem de vitória do Computador:



3.5 Empate

Uma terceira possibilidade é o empate. Caso ele ocorra é exibido uma mensagem de empate e as possibilidades de reiniciar e fechar o jogo:


O X O
X X O
X O X
Draw
Restart
Close Game