Criando um Novo Jogo no Servidor de Jogos

Prof. Dr. Luciano Antonio Digiampietri

Escola de Artes, Ciências e Humanidades da USP

Roteiro

- Contexto Educativo
- Descrição do Jogo
- Criando as Peças
- Criando o Jogo
- Atualizando o Servidor
- Jogando Utilizando o MultiJogador

Contexto Educativo

- O objetivo desta apresentação é ensinar como criar um novo jogo no servidor de jogos.
- As atividades presentes aqui podem ser usadas para exemplificar conceitos de orientação a objetos, algoritmos e estruturas de dados.

Descrição do Jogo

- Criaremos um Jogo da Velha.
- Já existe uma versão desse jogo no servidor (*Tictactoe*), ilustraremos a criação passo a passo de um jogo do mesmo tipo, porém chamado *JogoNovo*.
- Os passos são os seguintes:
  - Criar as peças;
  - Criar o jogo;
  - Atualizar a classe do servidor de jogos (*GameServer*)

Criando as Peças

- Para se criar peças de um novo jogo, basta criar uma classe que gerencie todas as peças.
- A classe peça (no nosso caso, *NovoJogoPiece*) deve estender a classe *Piece*.
- No caso do nosso *NovoJogo* (Jogo da Velha) há dois tipos de peça, que podem ser indicados pelo campo ID da peça.
- Podemos usar, no construtor os valores “X” e “O” como valores do ID da peça.

Criando as Peças - código

- A classe *NovoJogoPiece* além de estender a classe *Piece*, deverá sobrescrever alguns métodos (também criaremos um campo type e algumas constantes):
  - Há dois métodos para retornar uma *String* com o nome da classe (um estático e um "normal"):
    - public override String getClass2()
    - new public static String getClass()
  - Pode ser necessário criar um novo construtor :
    - public NovoJogoPiece(String id, int player, String type) : base(id, player)
  - Por fim, o método que retorna a *String* de como a peça será transmitida na mensagem XML:
    - public override String getString()

Criando as Peças - código

```
public class NovoJogoPiece : Piece
  {
     public const String PECA_X = "X";
     public const String PECA_O = "O";
     private String type = PECA_X;
     public override String getClass2()
     {
        return "NovoJogoPiece";
     }
     new public static String getClass()
     {
        return "NovoJogoPiece";
     }
     public NovoJogoPiece(String id, int player, String type)  : base(id, player)
     {
        this.type = type;
         }
     public override String getString()  {
        return ID+type+ID;
     }
  }
```

Criando as Peças

- Note que na classe que acabamos de implementar, o jogo, ao criar uma peça, deve passar o ID (tipicamente “X” ou “O”) e o tipo de peça (PECA_X ou PECA_O).
- Neste caso, por só termos dois tipos de peça, os identificadores (ID) e os tipos são idênticos.

Criando o Jogo

- Para se criar um novo jogo, deve-se estender a classe *Game* e sobrescrever os métodos que sejam convenientes.
- Neste exemplo, criaremos a classe *NovoJogoGame*.
- No caso do Jogo da Velha, nosso *NovoJogo* (Jogo da Velha) há dois tipos de peça, que podem ser indicados pelo campo ID da peça.
- Podemos usar, no construtor os valores “X” e “O” como valores do ID da peça.

Criando o Jogo – a classe *Game*

- A classe *Game* possui os seguintes atributos:
  - public String GAME_NAME = "Generic Game";
  - public int ACTUAL_PLAYER = -1;
  - public Board BOARD = null;
  - public int WINNER = -1;
  - public bool GAME_ENDED = false;
  - public bool DRAWN_GAME = false;
  - public ArrayList PLAYERS = new ArrayList();
  - public int CONT_ID = -1;
  - public String GAME_TYPE = "generic game";
  - public int MAX_PLAYERS = 2;
  - public bool WAITING_PLAYERS = true;
  - public int MIN_PLAYERS = 0;
  - public bool GAME_STARTED = false;
  - public String LAST_MESSAGE = " ";

Criando o Jogo – a classe *Game*

- A classe *Game* possui os métodos (além dos construtores e *getters*):
  - **public virtual bool add(int x, int y, int Player)**
  - **public bool canAdd(int x, int y, int Player)**
  - **public virtual bool canMove(int x1, int y1, int x2, int y2, int Player)**
  - **public String currentBoard()**
  - **public virtual bool endTurn(int Player)**
  - **public bool gameDraw()**
  - **public virtual String getClass2()**
  - **new public static String getClass()**
  - **public virtual bool gameEnded()**
  - **public virtual void initializeBoard()**
  - **public bool joinGame(int playerID)**
  - **public virtual bool move(int x1, int y1, int x2, int y2, int Player)**
  - **public virtual bool rotate(int x, int y, String direction, int Player)**
  - **public virtual void startGame()**

Criando o Jogo – a classe *Game*

- Notem que estes são todos os métodos públicos possíveis para qualquer jogo que o servidor de jogos possa ter.
- Só precisaremos implementar os métodos que são convenientes ao nosso jogo (a um Jogo da Velha), incluindo alguns métodos auxiliares.

Criando o Jogo - métodos

- Para o Jogo da Velha, não precisaremos de atributos auxiliares e o único tipo de ação do usuário em relação ao tabuleiro é adicionar peças:
  - **Métodos sobrescritos:**
    - **public override bool add(int x, int y, int Player)**
    - **public bool canAdd(int x, int y, Piece P)**
    - **public override bool gameEnded()**
    - **new public static String getClass()**
    - **public override String getClass2()**
    - **public override void initializeBoard()**
    - **public override void startGame()**

  - **Métodos auxiliares (e construtores):**
    - **public NovoJogoGame(String name) : base(name)**
    - **public NovoJogoGame(String name, int player1) : base(name, player1)**
    - **public bool player1Won()**
    - **public bool player2Won()**
    - **public bool won(int Player)**

Criando o Jogo – métodos (1)

- A seguir, cada método será descrito:
  - **public override bool add(int x, int y, int Player)**
    - **Utiliza o método canAdd para verifica se o jogador Player pode adicionar uma peça na posição x, y. Se sim, cria uma nova peça nessa posição do tabuleiro. public override bool add(int x, int y, int Player)**
  - **public bool canAdd(int x, int y, Piece P)**
    - **Verifica se o jogador Player pode adicionar uma peça na posição x, y. Isto envolve algumas condições, como: o jogo não acabou, a posição está livre e estamos no turno do jogador Player.**

Criando o Jogo – métodos (2)

- **public override bool gameEnded()**
  - **Verifica se o jogo está encerrado: há três possibilidades o jogador1 ganhou, o jogador 2 ganhou ou houve um empate (“velha”).**

- **public override void initializeBoard()**
  - **Cria uma instância de um tabuleiro (*Board*), do tamanho 3x3.**

- **public override void startGame()**
  - **Verifica se há dois jogadores no jogo e acerta o valor de algumas variáveis (por exempo, as que indicam que o jogo não está encerrado e não está aguardando jogadores)**

Criando o Jogo – métodos (3)

- **public NovoJogoGame(String name) : base(name)**
  - **Construtor que definirá algumas variáveis do novo jogo (tipo, número mínimo e máximo de jogadores, etc).**

- **public NovoJogoGame(String name, int player1) : base(name, player1)**
  - **Construtor que além de definir algumas variáveis do novo jogo, também incluirá o jogador *player1* neste jogo.**

Criando o Jogo – métodos (3)

- **public bool player1Won()**
  - **Método que verifica se o jogador 1 venceu.**

- **public bool player1Won()**
  - **Método que verifica se o jogador 2 venceu.**

- **public bool won(int Player)**
  - **Método auxiliar que, dado o identificador do jogador, verifica se ele venceu.**

Atualizando o Servidor

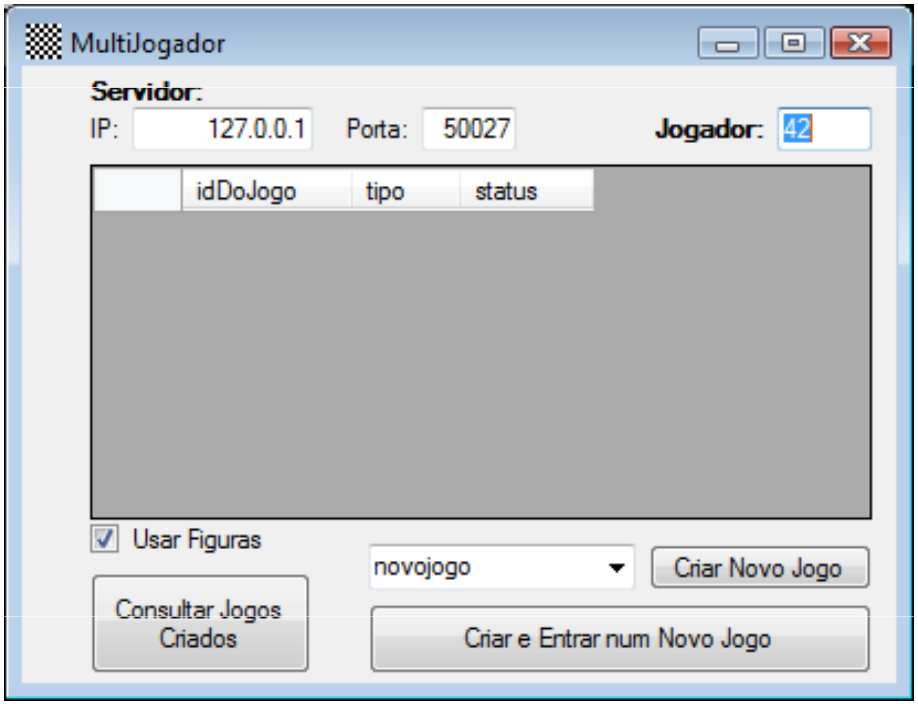
- A classe do servidor de jogos (*GameServer*) precisa ser atualizada para permitir a criação do nosso Novo Jogo.
- Para isto, basta inserir uma condição dentro do método *AnswerMessage*, onde ele verifica qual o tipo do jogo:

```
else if (game_type.Equals("novojogo", StringComparison.CurrentCultureIgnoreCase))
        {
           if (player1 < 0)
           {
              NEW_GAME = new manager.NovoJogoGame(game_name);
           }
           else
           {
              NEW_GAME = new manager.NovoJogoGame(game_name, player1);
           }
        }
```

Jogando Utilizando o MultiJogador

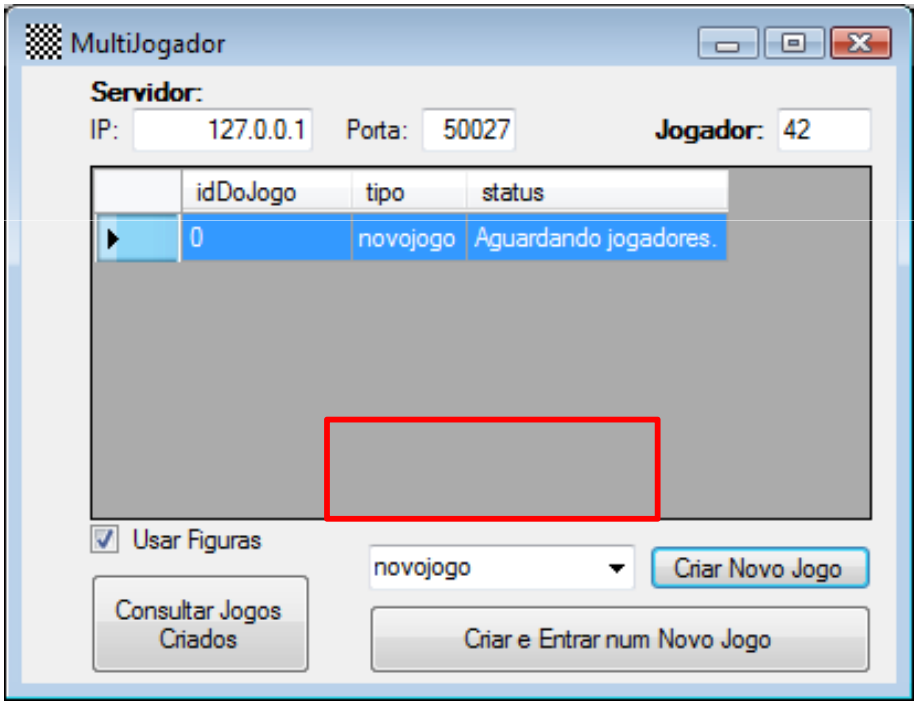
- O MultiJogador está preparado para jogar qualquer jogo do servidor de Jogos.
- Porém, se o jogo não for previamente cadastrado (o jogo e/ou as peças) não será possível mostrar figuras para as peças e o nome do jogo não aparecerá no ComboBox da tela de entrada.
- Ou você atualiza o MultiJogador ou, simplesmente, joga no modo sem figuras.

Jogando Utilizando o MultiJogador



Jogando Utilizando o MultiJogador

- Para jogo o *NovoJogo* basta criar uma instância deste jogo. Preecha (manualmente) o valor do ComboBox para a criação de novos jogos e clique em “Criar Novo Jogo”:



Jogando Utilizando o MultiJogador

- Dê duplo clique no jogo para entrar no *NovoJogo* recém criado e aguarde outros jogadores:

Jogando Utilizando o MultiJogador

- Depois disso é só jogar (observe que o “nome” das peças é aquele passado pelo método *getString* da classe *NovoJogoPiece).*

Dicas

- Veja o código completo do NovoJogo no projeto ServidorSimples.
  - Para se criar uma nova peça, dentro da classe *NovoJogoGame*, você pode utilizar:
    - **Piece P = new NovoJogoPiece("X", PLAYER1,NovoJogoPiece.PECA_X);**
    - **Piece P = new NovoJogoPiece("O", PLAYER2,NovoJogoPiece.PECA_O);**
  - Para verificar se o jogo acabou empatado, basta contar quantas peças diferentes de peças nulas (*NullPiece*) existem, se houver 9, o jogo acabou, e, se ninguém ganhou, o jogo acabou empatado.

http://www.uspleste.usp.br/digiampietri/jogos/