MODELO DE CLASSES

Recife, dezembro de 2010.

http://code.google.com/p/denisbattleship

SUMÁRIO

HISTÓRICO DE VERSÕES

1.0 [10/10/2010]

Definição inicial do modelo.

1.1 [18/10/2010]

Atualização do diagrama de classes de domínio.

1.2 [05/11/2010]

Início da construção do diagrama de projeto.

1.3 [27/11/2010]

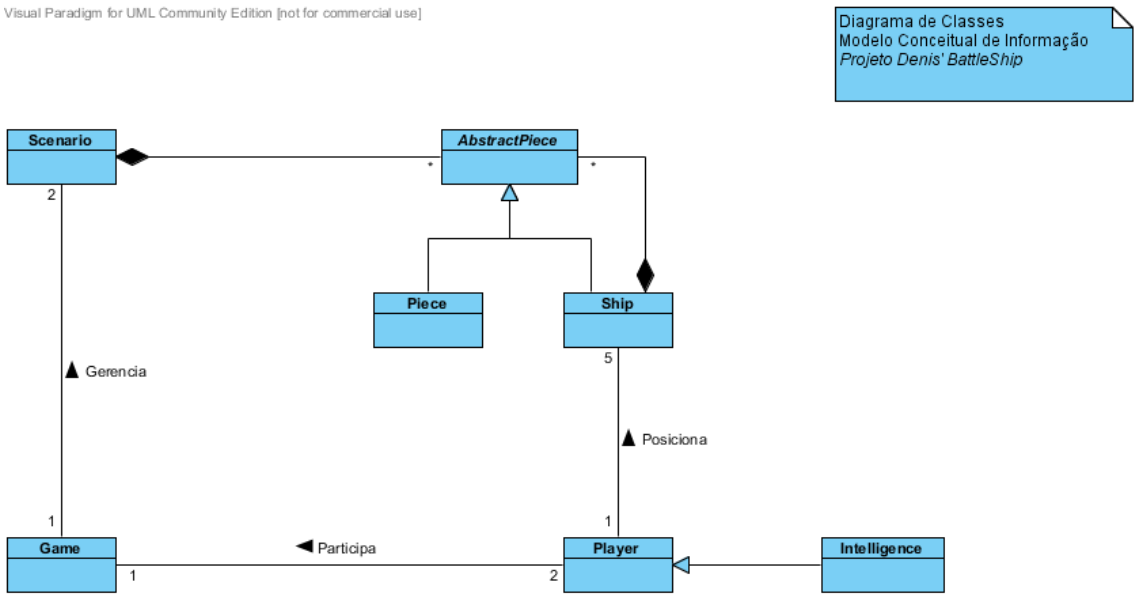
Ajustes e aplicação de padrões no diagrama de projeto.

1.4 [07/12/2010]

Atualização referente à manipulação das regras do jogo pelos controle.

1 DIAGRAMAS

1.1 Classes de Domínio *(Modelo Conceitual de Informação)*



Considerações:

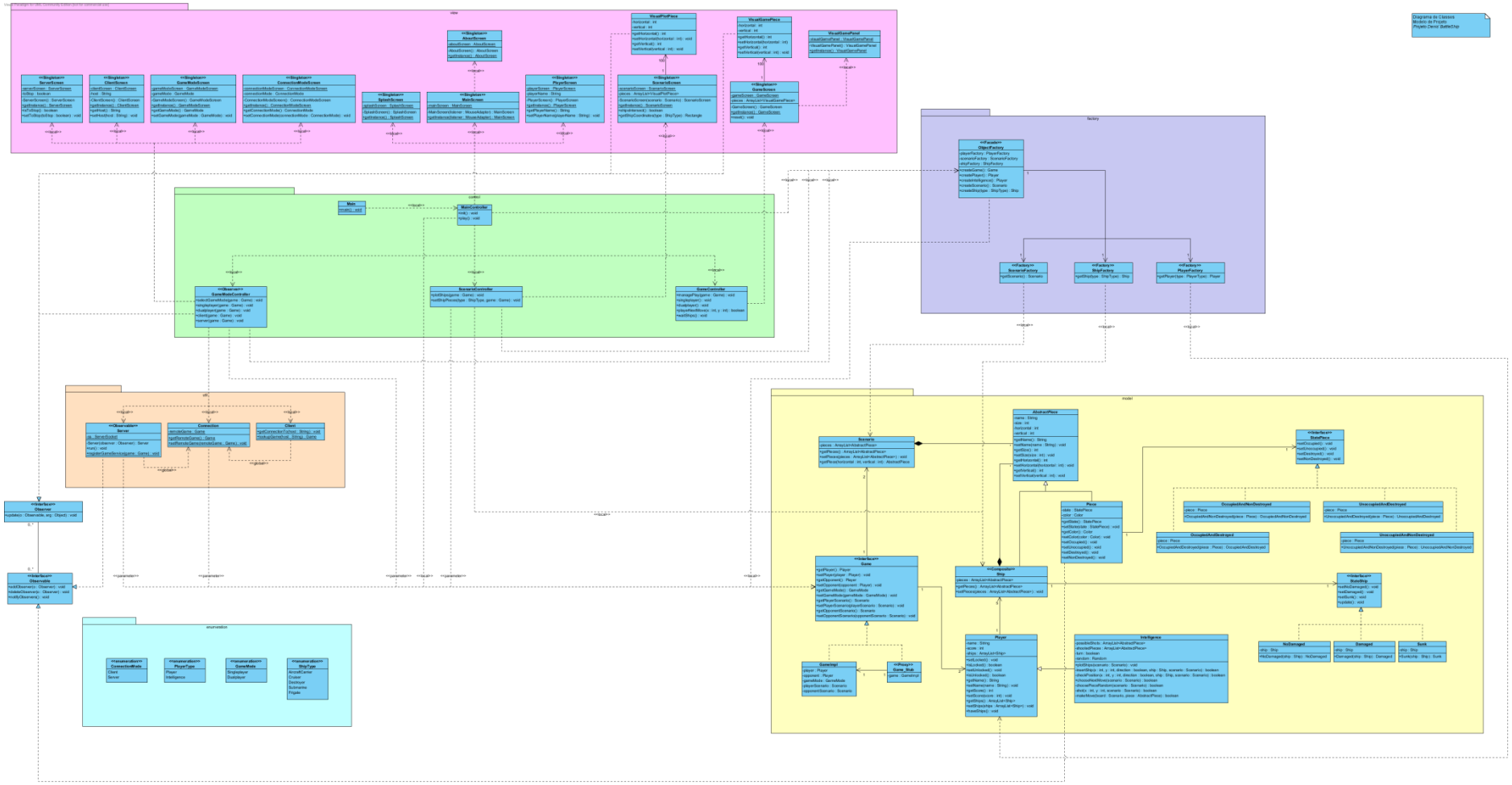
O seguinte diagrama apresenta o modelo conceitual de informação de nosso sistema, diagrama mais conhecido como classes de domínio. Nele estão listadas as entidades principais do sistema. São elas: jogo, tabuleiro, jogador, inteligência, embarcação e peças.

O jogo possui dois tabuleiros e dele participam dois jogadores. O tabuleiro é composto por peças. O jogador, por sua vez, possui 5 (cinco) embarcações posicionadas por ele próprio.

Assim como o tabuleiro, as embarcações também são compostas de peças. Por esse motivo, a solução mais viável foi a aplicação de um *composite* para solucionar a questão de recursividade aqui encontrada.

A inteligência é um jogador com comportamento diferenciado, por isso a extensão. É ela que irá implementar o processamento automático de jogadas quando o jogador desejar jogar sozinho.

1.2 Classes de Implementação *(Modelo de Projeto)*



Considerações:

O seguinte modelo segue a um padrão de responsabilidades onde o controle da aplicação é responsável pela instanciação e exibição das telas, pela captura dos dados das telas, e pelo processamento da informação capturada, gerando os estados decorrentes da execução do sistema.

O diagrama anterior está dividido em pacotes. São eles: *view*, *control*, *util*, *enumeration*, *factory* e *model*. No pacote de *view* estão todas as telas, que por sua vez implementam o padrão *singleton* para que a informação seja recuperada delas posteriormente à sua exibição. No pacote de controle estão todos os controladores e a classe Main. Esta é responsável pelo início da aplicação.

No pacote *util*, estão as classes responsáveis pela conexão a baixo nível entre servidores e clientes quando o modo de jogo é *dualplayer*. Eles manipulam um *proxy* – implementado usando RMI de Java – para obter os dados do oponente remoto.

No pacote *factory* estão todas as classes responsáveis pela instanciação de objetos complexos, além de uma fachada ObjectFactory, que simplifica uma grande chamada de métodos para criar um objeto.

Por fim, no pacote *model*, estão todos as entidades do modelo conceitual com suas especificações de implementação. Também são encontrados padrões de projeto para solucionar questões específicas, como o padrão *state* (para os estados de Ship e Piece), *composite* (para solucionar a recursividade de Ship e Piece) e *proxy* (para solucionar o problema de sincronização do Game via rede).

Neste último pacote também é necessário salientar que as peças visuais (VisualPlotPiece e VisualGamePiece) são observadores de Piece, pois se houver alguma alteração do estado de Piece durante a execução, as peças visuais vinculadas poderão repintar-se de acordo com a mudança.