
# Programação Avançada 2015/2016

## Linha3 - 2º Fase

Turma: INF-ES-02
Ricardo José Horta Morais
Nº 140221066

# Índice

# 1 Introdução

O projeto tem como objetivo aplicar os conhecimentos adquiridos no âmbito da unidade curricular de Programação Avançada.

O trabalho consiste em criar um jogo parecido ao 3 em linha em javaFX.

Numa primeira fase vai ser definido o modelo de análise. Também ainda nesta primeira fase vão ser especificados e implementados os tipos abstratos de dados para posteriormente serem utilizados na implementação do Jogo.

# 2 Utilização de padrões

Durante o projeto foram aplicados alguns padrões de desenho. É importante focar que os padrões foram aplicados por necessidade e não à força.

- Iterator – aplicado para percorrer as TADs
- Memento – aplicado para criar pontos de restauro no Jogo e por sua vez implementar a funcionalidade de undo.
- Strategy – aplicado para a estratégia de pontuação, conjuntos de peças, comparadores, peças … etc.
- Observer – aplicado para o logger do jogo e para o gestor de jogos observar os jogos.
- Delegation – aplicado múltiplas vezes para utilizar a funcionalidade de uma classe com a mesma interface. Usado nos Nodes das TADs, nos loggers, nas peças e claro usado todas as vezes que é aplicado o padrão strategy.

# 3 Diagrama de classes do modelo de desenho

## 3.1 Funcionalidade logger

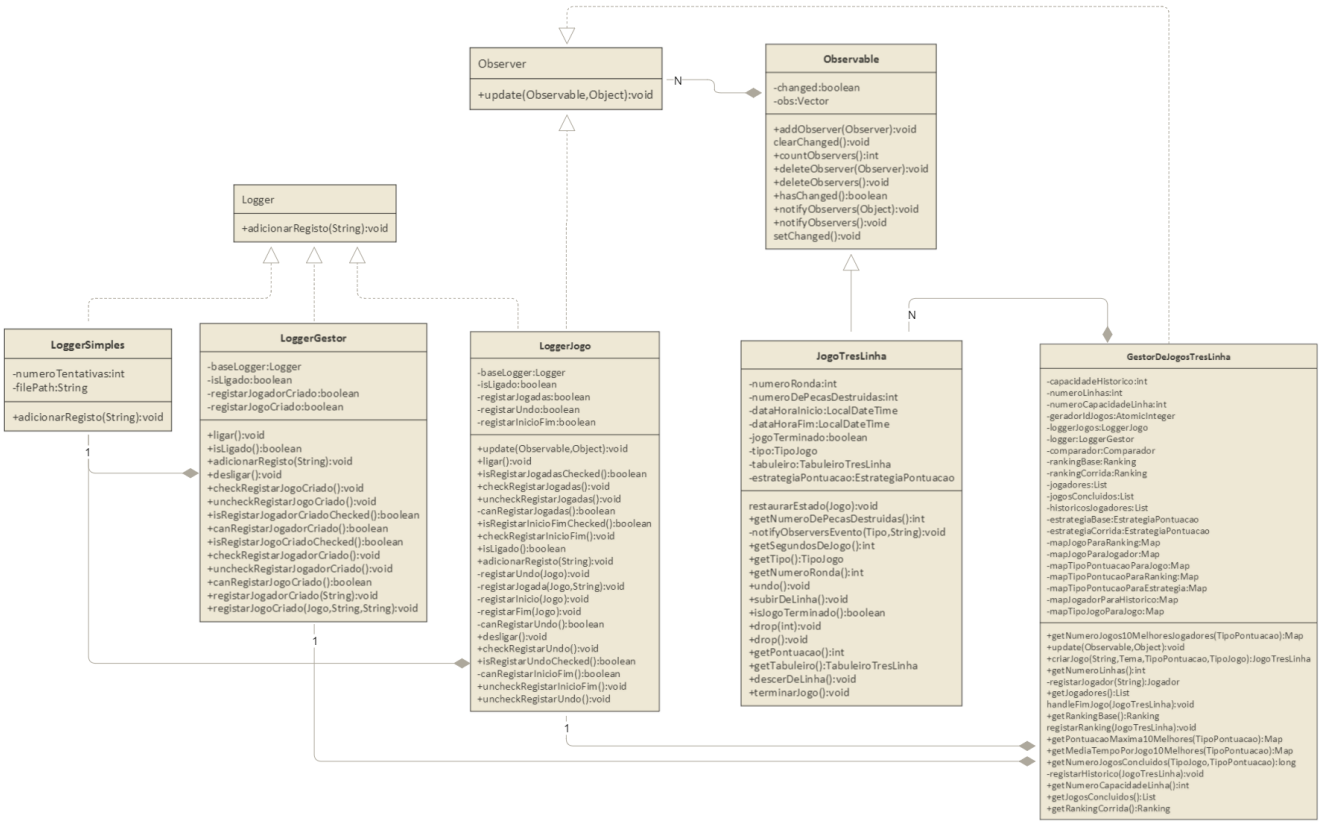


**Figura 1 - Modelo classes de desenho da funcionalidade logger**

## 3.2 Descrição funcionalidade logger

A Figura 1 representa o diagrama de classes relativo à funcionalidade de logger. No projeto existe dois loggers, o logger do gestor e o logger do jogo. Ambos são criados no gestor. O logger do gestor é usado pelo gestor sempre que é necessário registar algo sobre o mesmo e está fortemente acoplado a este. O logger jogo pode estar associado a vários jogos, este logger usa o padrão observer para receber eventos dos jogos e para tratar de registar as informações relativas aos mesmos. Assim o jogo não sabe o que está a ser registado no logger e este logger consegue guardar informação de vários jogos ao em simultâneo. Ambos os loggers usam o padrão Delegate para aproveitar a funcionalidade de um LoggerSimples. Assim parte da funcionalidade é reaproveitada e não é estendida nenhuma classe.

## 3.3 Funcionalidade Undo

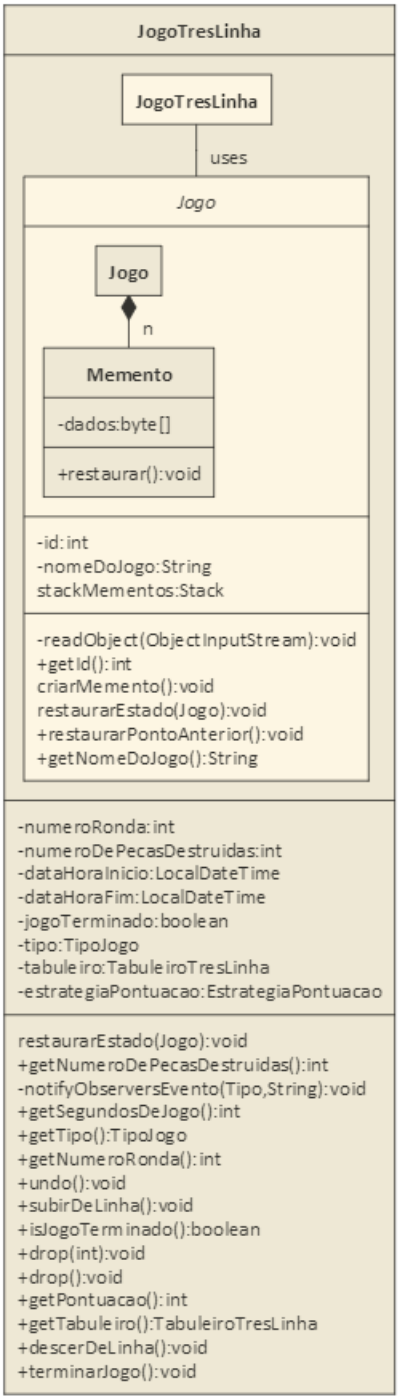


**Figura 2 - Modelo classes de desenho da funcionalidade de undo**

## 3.4 Descrição funcionalidade undo

A Figura 2 representa o diagrama de classes relativo à funcionalidade undo. Para a implementação desta funcionalidade, foi utilizado o padrão memento. Todo o processo de criação, restauro e manutenção de mementos está encapsulado na classe Jogo. Ao ser criado um memento, a própria instância do jogo é serializada e transformada em bytes, essa serie de bytes é então guardada no memento. Para restaurar um memento, os bytes são deserializados e são substituídos todos os atributos do objeto. Os mementos são guardados numa stack. Para que esta funcionalidade se propague para as subclasses é necessário fazer um override ao método restaurarEstado() que recebe um jogo com todo o estado do memento, basta fazer o cast do objeto à subclasse e extrair todos os valores dos atributos.

## 3.5 Funcionalidade de estatísticas, histórico e ranking

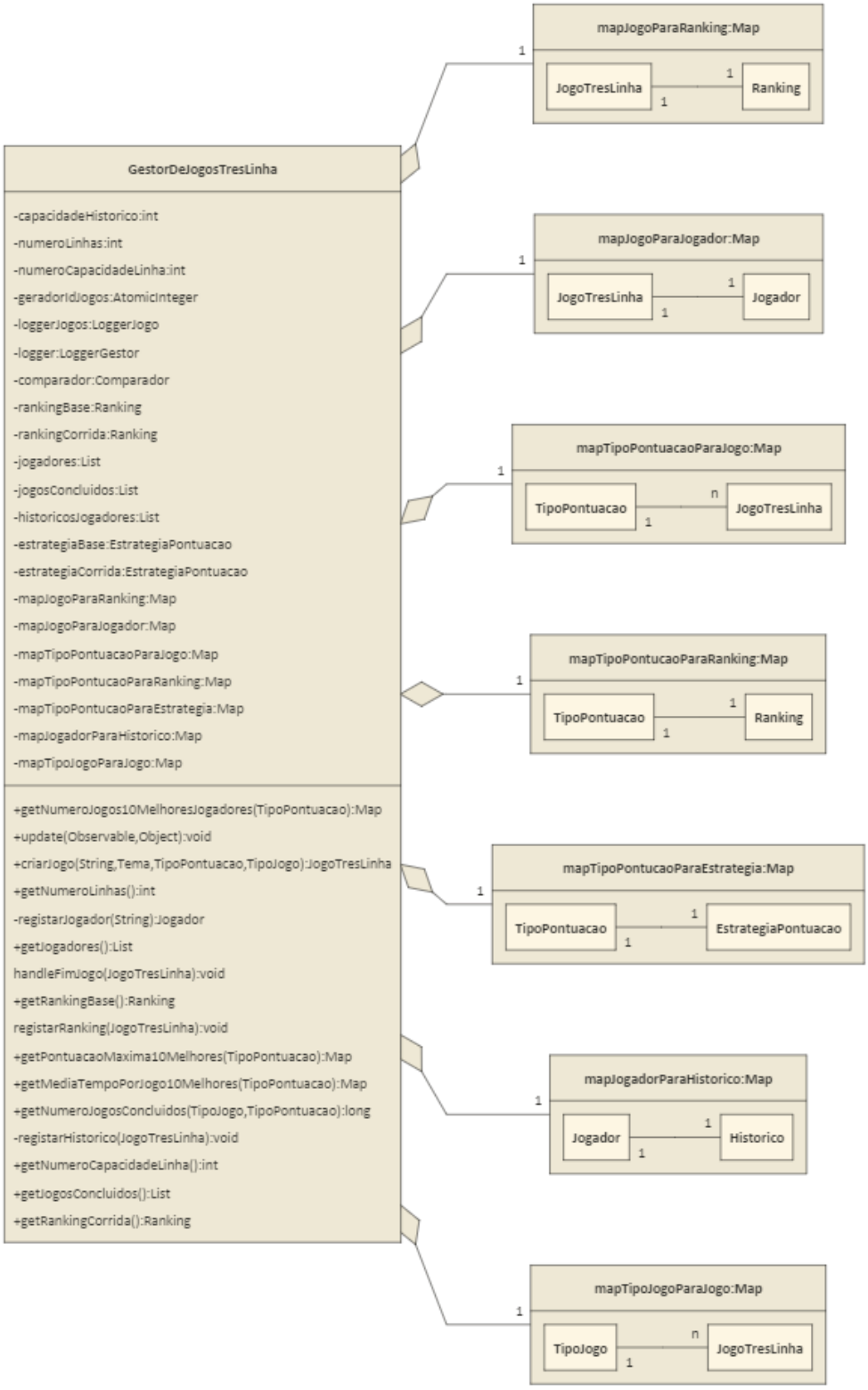


**Figura 3 - Modelo classes de desenho da funcionalidade de estatística, histórico e ranking**

## 3.6 Descrição funcionalidade de estatísticas, histórico e ranking

A Figura 3 apresenta o diagrama de classes das associações de mapas. O diagrama mostra que todas as associações estão centralizadas na classe GestorDeJogosTresLinha. Aqui através de mapas são guardadas todas as associações criadas. Isto ajuda a minimizar o acoplamento entre classes. Os jogadores, jogos, históricos, rankings…etc. não têm conhecimento destas associações e portanto não existe uma dependência entre as classes. Assim é possível criar um jogoTresLinha sem que este tenha histórico, ranking, jogador e/ou interface gráfica. Os mapas são também úteis para calcular estatísticas, os mapas podem ser interpretados como tabelas de relação de uma base de dados. Para obtermos a pontuação máxima dos 10 melhores jogadores basta usar o mapTipoPontuacaoParaRanking com a chave "tipo de pontuação" para obtermos o ranking associado e em seguida retirar os 10 primeiros elementos do ranking. Cada jogador também tem um histórico associado através dos mapas.

## 3.7 O jogo

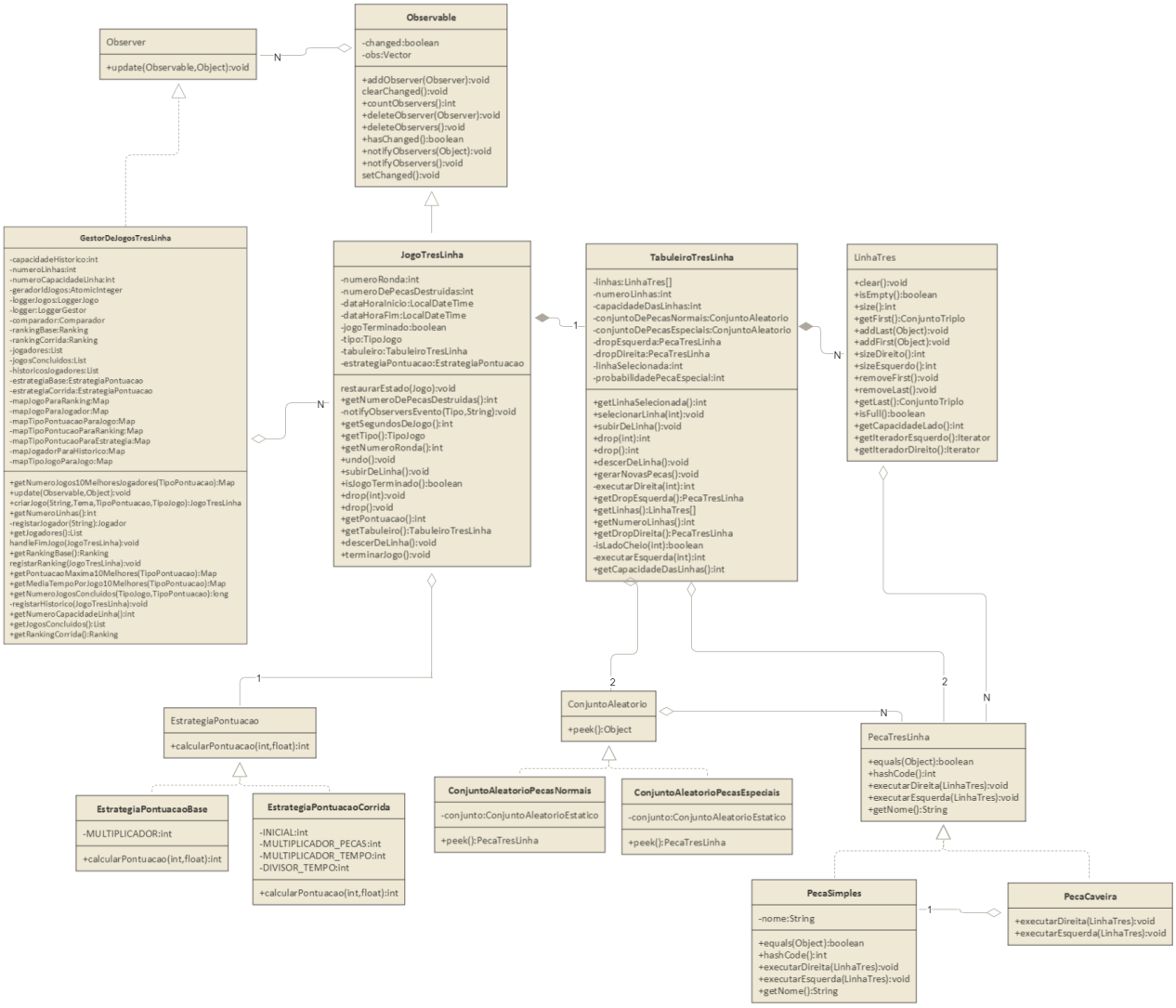


**Figura 4 - Modelo classes de desenho da funcionalidade de jogar**

## 3.8 Descrição do jogo

A Figura 4 apresenta o diagrama de classes do jogo. O GestorDeJogosTresLinha trata de disponibilizar um método de criação de Jogos. Ao serem criados jogos através do gestor, o gestor regista-se como observador dos mesmos, assim pode observar todos os eventos dos jogos. Ao terminarem os jogos, este deixam de ser observados e são arquivados pelo gestor. Um jogo tem sempre um tabuleiro, este tabuleiro guarda toda a mecânica do tabuleiro. O jogo guarda informações relativas ao que vai acontecendo no jogo como o número de peças destruídas, número de ronda, etc. Uma estratégia de pontuação é passada ao Jogo para que este possa calcular a pontuação sem saber qual o cálculo que está a ser feito.