Universidade do Minho - Mestrado em Engenharia informática
Engenharia de Sistemas de Software
Arquitecturas de Software
Trabalho 1

João Magalhães pg27762
Paulo Guterres pg24160

Conteúdo

Introdução

Este trabalho, proposto na unidade curricular de Arquitecturas de Software, do perfil de Engenharia de Sistemas de Software, tem como objectivo a implementação de um serviço denominado de Futscore, cuja função é enviar updates desportivos de um ou mais jogos de futebol para adeptos interessados.
Esta implementação foi feita com base no *design pattern Observer* em que temos um conjunto de *Observers* (adeptos), que recebem os updates dos jogos aos quais estão interessados, ou seja, os *Subjects.*

Diagrama de classes

Para este serviço, foi criado diagrama de classes presente no anexo, que explica de que maneira cada classe se associa com outras, bem como as operações e atributos de cada uma.
Antes de mais é importante referir que as classes Jogo e Jornada não têm nomes intuitivos, visto que a classe jogo se refere a uma competição e a classe Jornada se refere a um jogo.
Importante de referir também que as equipas só existem no contexto do torneio, ou seja, não é possível que uma equipa participe em dois ou mais torneios ao mesmo tempo.
Como podemos ver temos duas interfaces, uma correspondendo aos *Observers* e outra aos *Subjects.* As classes que implementam a interface Subject são observadas pelas classes que implementam Observer (neste caso os jogos são observados pelos adeptos). A interface Observer tem apenas uma operação, o update, que é substituída (overridden) pela classe que implementa esta interface (a classe Adepto). Com a interface Subject acontece o mesmo, mas desta vez a classe que a implementa é o Jogo.
A classe Equipa tem os atributos nome, treinador e os atributos referentes ao campeonato em que está inserida, ou seja, o nome do campeonato, os jogos feitos, os golos marcados e sofridos e os pontos obtidos. Como operadores, esta classe tem os gets e sets dos atributos e o construtor dela própria. Esta classe vai ser usada na classe Jornada para indicar as equipas que vão jogar o jogo e também na classe Adepto num array para indicar que equipas esse adepto está a seguir.
A classe Jornada serve para indicar que equipas participam num jogo e qual é o seu resultado. Esta classe vai ter duas equipas (casa e fora), dois inteiros correspondentes aos golos de cada equipa, um inteiro que indica qual o número da jornada no campeonato e um boolean que indica se essa jornada já foi efectuada.
A classe Evento corresponde aos eventos que existem no sistema. Estes eventos (por exemplo golos ou remates) vão existir num ArrayList em cada jogo (que corresponde a todos os eventos que podem ocorrer nesse conjunto de jogos) e num Array em cada Adepto (que corresponde aos eventos que o Adepto pretende observar, ou seja, sobre os quais o Adepto pretende ser avisado).
A classe Adepto, como já referido anteriormente, tem um nome (que corresponde ao nome do adepto), um array de Equipas (que corresponde às equipas que o adepto quer seguir) e um array de Eventos (que correspondem aos eventos que o Adepto pretende seguir).

A classe Jogo contém um arraylist de adeptos, correspondente aos adeptos que estão com este jogo escolhido para observação, um arraylist de Evento, que corresponde a todos os eventos que podem ocorrer durante esse jogo e uma Jornada à qual o jogo corresponde. A classe tem também dois inteiros e uma string. Os inteiros correspondem ao tempo de jogo actual (o time) e a que equipa (1 ou 2 correspondente a casa ou fora) um determinado evento (a string) ocorreu. Estes três atributos servem para os avisos que vão sendo enviados aos adeptos.
A classe jogo tem a operação de começar, que começa o jogo e gera eventos aleatórios que estejam dentro do array de eventos do jogo. Cada vez que um evento é gerado, todos os Observers que estejam à espera deste evento são avisados, desde que tenham pelo menos uma das equipas em jogo na sua lista de equipas a seguir. Por exemplo, se um adepto se subscrever à equipa Porto e ao evento em que existe um golo, este adepto vai ser avisado de todos os golos, a favor ou contra o Porto para todos os jogos que o Porto fizer.
A classe campeonato, que contém apenas o atributo nome, com os gets e sets respectivos, serve apenas para servir de informação para as classes Equipa (indicando a que campeonato essa equipa está inscrita) e Dados (que indica a que campeonato pertencem os dados).
Por último, a classe Dados contém três ArrayLists, que correspondem às equipas, aos eventos e às jornadas que um determinado campeonato tem.

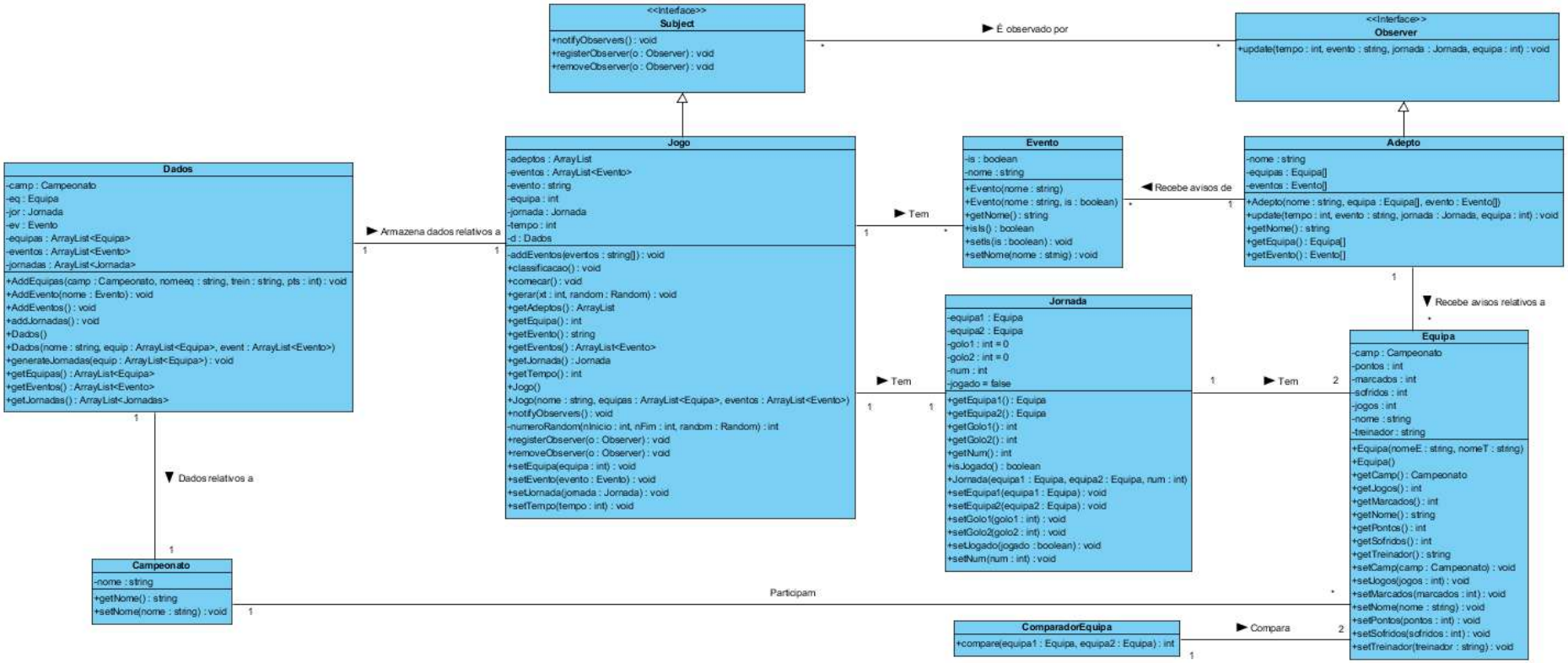
Modo de Utilização

Esta aplicação pode ser utilizada de duas maneiras:
A primeira corresponde ao modo de teste, em que as equipas, jornadas e eventos estão definidos e serve apenas, como o nome indica, para teste. Apesar de já ter a maior parte das informações, é possível adicionar Adeptos quando se corre em modo teste.
A outra maneira permite ao utilizador criar um novo campeonato, dizendo que equipas devem participar e que eventos devem ocorrer nos jogos desse campeonato. Como no modo de teste, posteriormente à criação do campeonato é possível adicionar novos Adeptos.
Em ambas as maneiras de utilização, é possível verificar a classificação actual e jogar a próxima jornada, ou seja, gerar o próximo jogo.

Conclusão

A utilização do design pattern Observer neste caso permitiu que a implementação fosse fácil e rápida, o que mostra mais uma vez que a utilização de padrões na implementação do código pode evitar muito trabalho aos implementadores, dando também uma garantia de que, se as regras do padrão são cumpridas, provavelmente a solução vai ser satisfatória.

Anexo


<<Interface>>
Subject
+notifyObservers() : void
+registerObserver(o : Observer) : void
+removeObserver(o : Observer) : void
É observado por
<<Interface>>
Observer
+update(tempo : int, evento : string, jornada : Jornada, equipa : int) : void
Dados
-camp : Campeonato
-eq : Equipa
-jor : Jornada
-ev : Evento
-equipas : ArrayList<Equipa>
-eventos : ArrayList<Evento>
-jornadas : ArrayList<Jornada>
+AddEquipas(camp : Campeonato, nomeeq : string, trein : string, pts : int) : void
+AddEvento(nome : Evento) : void
+AddEventos() : void
+addJornadas() : void
+Dados()
+Dados(nome : string, equip : ArrayList<Equipa>, event : ArrayList<Evento>)
+generateJornadas(equip : ArrayList<Equipa>) : void
+getEquipas() : ArrayList<Equipa>
+getEventos() : ArrayList<Evento>
+getJornadas() : ArrayList<Jornadas>
Armazena dados relativos a
Jogo
-adeptos : ArrayList
-eventos : ArrayList<Evento>
-evento : string
-equipa : int
-jornada : Jornada
-tempo : int
-d : Dados
-addEventos(eventos : string[]) : void
+classificacao() : void
+comecar() : void
+gerar(xt : int, random : Random) : void
+getAdeptos() : ArrayList
+getEquipa() : int
+getEvento() : string
+getEventos() : ArrayList<Evento>
+getJornada() : Jornada
+getTempo() : int
+Jogo()
+Jogo(nome : string, equipas : ArrayList<Equipa>, eventos : ArrayList<Evento>)
+notifyObservers() : void
-numeroRandom(nInicio : int, nFim : int, random : Random) : int
+registerObserver(o : Observer) : void
+removeObserver(o : Observer) : void
+setEquipa(equipa : int) : void
+setEvento(evento : Evento) : void
+setJornada(jornada : Jornada) : void
+setTempo(tempo : int) : void
Tem
Evento
-is : boolean
-nome : string
+Evento(nome : string)
+Evento(nome : string, is : boolean)
+getNome() : string
+isIs() : boolean
+setIs(is : boolean) : void
+setNome(nome : stnig) : void
Recebe avisos de
Adepto
-nome : string
-equipas : Equipa[]
-eventos : Evento[]
+Adepto(nome : string, equipa : Equipa[], evento : Evento[])
+update(tempo : int, evento : string, jornada : Jornada, equipa : int) : void
+getNome() : string
+getEquipa() : Equipa[]
+getEvento() : Evento[]
Recebe avisos relativos a
Tem
Jornada
-equipa1 : Equipa
-equipa2 : Equipa
-golo1 : int = 0
-golo2 : int = 0
-num : int
-jogado = false
+getEquipa1() : Equipa
+getEquipa2() : Equipa
+getGolo1() : int
+getGolo2() : int
+getNum() : int
+isJogado() : boolean
+Jornada(equipa1 : Equipa, equipa2 : Equipa, num : int)
+setEquipa1(equipa1 : Equipa) : void
+setEquipa2(equipa2 : Equipa) : void
+setGolo1(golo1 : int) : void
+setGolo2(golo2 : int) : void
+setJogado(jogado : boolean) : void
+setNum(num : int) : void
Tem
Equipa
-camp : Campeonato
-pontos : int
-marcados : int
-sofridos : int
-jogos : int
-nome : string
-treinador : string
+Equipa(nomeE : string, nomeT : string)
+Equipa()
+getCamp() : Campeonato
+getJogos() : int
+getMarcados() : int
+getNome() : string
+getPontos() : int
+getSofridos() : int
+getTreinador() : string
+setCamp(camp : Campeonato) : void
+setJogos(jogos : int) : void
+setMarcados(marcados : int) : void
+setNome(nome : string) : void
+setPontos(pontos : int) : void
+setSofridos(sofridos : int) : void
+setTreinador(treinador : string) : void
Dados relativos a
Campeonato
-nome : string
+getNome() : string
+setNome(nome : string) : void
Participam
ComparadorEquipa
+compare(equipa1 : Equipa, equipa2 : Equipa) : int
Compara